Contribution to wide angle visual perception for mobile robots and self driving cars

Bogdan Khomutenko

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

CENTRALE NANTES

THESE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Automatique, Productique et Robotique*

Par

Bogdan KHOMUTENKO

Contribution à la Perception Visuelle Basée Caméras Grand Angle pour la Robotique Mobile et Les Véhicules Autonomes

Thèse présentée et soutenue à Nantes, le 15/06/2018
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Patrick Rives	Directeur de recherche, INRIA Sophia Antipolis
Pascal Vasseur	Professeur, IUT de Rouen

Composition du Jury :

Président :	Eric Marchand	Professeur, Université de Rennes 1
Examinateurs :	Claire Dune	Maître de conférence, Université Sud-Toulon
Dir. de thèse :	Philippe Martinet	Professeur, Ecole Centrale de Nantes
Co-encadrant :	Gaëtan Garcia	Maître de Conférences, Ecole Centrale de Nantes

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

CENTRALE NANTES

Titre : Contribution à la perception visuelle basée caméras grand angle pour la robotique mobile et les véhicules autonomes

Mots clés : fisheye, étalonnage, stereo, reconstruction 3D, localisation directe

Résumé Ce travail de thèse présente un nouveau modèle de projection pour les cameras fisheye, qui est mathématiquement simple et pourtant montre une haute précision une fois appliqué aux caméras réelles. Les propriétés géométriques de ce modèle ont été analysées en utilisant le concept de surface de projection, introduit dans ce travail. En particulier, une inverse analytique de ce modèle a été établie, ainsi qu'une équation implicite des projections de lignes droites. Cette dernière nous a permis de développer une méthode de reconstruction 3D directe basée vision pour les caméras fisheye sans rectifier les images. Cela a été fait grâce à une algorithme de rastérisation de courbes implicites. Cet algorithme de reconstruction nous permet d'employer le Semi-Global Matching pour obtenir une reconstruction 3D précise. Tous ces éléments ont été employés dans un système de localisation visuelle directe avec deux méthodes de recalage d'images : minimisation d'erreur photométrique et maximisation d'information mutuelle. L'étalonnage intrinsèque et extrinsèque d'un robot mobile équipé de caméras fisheye a été considéré et une toolbox d'étalonnage a été développée.

Title : Contribution to wide angle visual perception for mobile robots and self driving cars

Keywords : fisheye, calibration, stereo, 3D reconstruction, direct localization

Abstract : This thesis presents a novel projection model for fisheye cameras, which is mathematically simple and yet shows a high precision when applied to real cameras. Geometric properties of the model have been analyzed using the concept of projection surface, introduced in this work. In particular, a closed-form inverse mapping and an implicit equation for straight line projection have been found. This fact has been used to develop a method of direct stereo correspondence on raw fisheye images via rasterization of implicit curve. This correspondence algorithm allows us to apply the Semi-Global Matching algorithm to get an accurate 3D reconstruction using fisheye stereo systems. All these elements have been shown to be applicable to a direct visual localization system with two different methods of image registration: direct photometric error minimization and mutual information maximization. Intrinsic and extrinsic calibration of a mobile robot with fisheye cameras has been considered and a toolbox for such a calibration has been developed

Contribution to Wide-Angle Visual Perception for Mobile Robots and Self-Driving Cars

Contribution à la Perception Visuelle Basée Cameras Grand Angle pour La Robotique Mobile et Les Véhicules Autonomes

Bogdan Khomutenko

December 26, 2018

Abstract

This work lies in the domain of computer vision and autonomous navigation for mobile robots. The principal subject is the use of fisheye cameras for visual perception.

The first research question covered by this work is geometric modeling of fisheye cameras. The goal is to make fisheye cameras as easy to use as classical pinhole cameras. Existing models are either not precise enough or too complicated for analytical analysis of their geometric properties. We propose a novel projection model based on the Unified Camera Model, also known as the Spherical Model. Adding one more projection parameter increases the model expressiveness and makes additional distortion mappings unnecessary. As experiments show, this model accurately approximates a large variety of different fisheye lenses. The notion of projection surface, proposed in this work, allowed us to find an analytic inverse of the model and implicit equations of straight line projections.

An efficient and flexible calibration toolbox for multi-camera systems has been developed. It has a novel fully-automated subpixel detector of calibration boards, which is significantly faster than that of OpenCV. A method of calculating optimal trajectories for extrinsic calibration of mobile robots has been proposed and tested. Its effectiveness to reduce the noise impact has been demonstrated on simulated data. It has been applied to calibrate a complete system Camera-Odometry, which has been used for localization experiments.

The next research question is how to compute direct stereo correspondence between two fisheye images. The main goal was to avoid additional filtering such as undistortion and rectification mappings. It was possible thanks to epipolar geometry of fisheye stereo systems provided by the developed model. After computing epipolar curve equations, we sample image pixelwise along them in order to compute the correspondence cost. This approach allows us to apply the Semi-Global Matching algorithm to regularize the computed disparity map and get a more accurate reconstruction. We proposed several techniques of computing the correspondence cost, which improve the correspondence quality. The proposed stereo algorithm has been tested on both synthetic and real data, using the ground truth for validation purposes. Experiments show that the reconstruction of planar textured objects is accurate and precise, which validates the geometric model behind.

The last research question is visual localization. A novel closed-form triangulation method allows us to significantly reduce the number of unknowns in the visual odometry problem. Fisheye cameras have been combined with direct visual localization and additional sources of localization information, such as IMU and wheel odometry. This method was tested on simulated data and showed high precision. Using Mutual Information as similarity measure for image registration allowed the system to relocalize itself using real data in a map constructed a few days earlier with some changes in the environment and in lighting conditions. The method demonstrated robustness with respect to moving objects, such as cars and pedestrians.

Keywords: fisheye, calibration, stereo, 3D reconstruction, direct localization

Résumé

Ce travail de thèse s'inscrit dans le domaine de la vision par ordinateur et de la navigation autonome pour les robot mobiles. Le sujet principal est l'utilisation d'optiques grand angle pour la perception visuelle.

La première problématique traitée est la modélisation géométrique des cameras fisheye. Le but est de rendre leur utilisation aussi simple que celle des cameras classiques trou d'épingle. Les modèles existants manquent de précision ou bien sont trop compliqués pour que l'on puisse analyser leurs propriétés géométriques analytiquement. Nous proposons un nouveau modèle de projection basé sur le Modèle Unifié, aussi connu comme Modèle Sphérique. En rajoutant un paramètre intrinsèque, on augmente l'expressivité du modèle et évite le recours à une fonction de distorsion supplémentaire. Les expériences effectuées ont démontré la capacité du modèle à approcher, avec une grande précision, une large gamme d'objectifs fisheye différents. Le concept de surface de projection, proposé dans ce travail, nous a permis de trouver une inverse analytique de ce modèle ainsi que d'établir les équations de projections de droites.

Une boîte à outils d'étalonnage, flexible et efficace, conçue pour les systèmes multi-cameras, a été développée. Elle contient un nouveau détecteur de mire d'étalonnage, qui est automatique, a une précision sub-pixelique, et qui est plus rapide que le détecteur fourni avec OpenCV. Une méthode de calcul de trajectoires optimales pour l'étalonnage extrinsèque de robots mobiles a été développée et testée. Son aptitude à réduire l'impact du bruit sur la précision a été démontré sur des données synthétiques. Elle a été utilisée pour étalonner un système complet Camera-Odométrie, qui a été employé pour tester les algorithmes de localisation.

La problématique suivante était de calculer la correspondance stéréo directement dans l'espace d'images fisheye, tout en évitant un filtrage additionnel et la rectification d'images. Cela a été possible grâce à la géométrie épipolaire des systèmes stéréos fisheye, donnée par le modèle proposé. Une fois les équations des courbes épipolaires calculées, l'image est échantillonnée, pixel par pixel, le long de celles-ci afin de trouver le coût de correspondance. Cette approche nous permet d'employer l'algorithme de *Semi-Global Matching* et d'obtenir une reconstruction 3D précise. Un certain nombre de techniques, qui améliorent la qualité de correspondance, ont été proposées. De nombreux essais avec des données synthétiques ainsi que réelles ont montré que l'algorithme est capable de reconstruire des objets planaires texturés avec une grande précision.

La localisation visuelle est la dernière problématique traitée dans notre travail. Une nouvelle méthode de triangulation analytique nous permet de réduire le nombre de paramètres d'optimisation dans le problème d'odométrie visuelle d'une façon significative. Une méthode de localisation visuelle directe basée caméras fisheye, qui emploie aussi d'autres sources d'information de localisation, comme une centrale inertielle ou l'odométrie des roues, a été développée. Des essais avec des données synthétiques montrent sa précision. L'utilisation de l'Information Mutuelle en tant que mesure de similarité permet au système de se relocaliser avec des données réelles dans une carte construite quelques jours auparavant malgré des changements dans l'environnement et l'éclairage. Cette méthode se montre robuste par rapport aux objets mobiles, tels que voitures en marche et piétons.

Mots-clés : fisheye, étalonnage, stéréo, reconstruction 3D, localisation directe

Contents

Greek Letters and Their Use

small	capital	name	possible use
α	A	alpha	intrinsic parameters
β	B	beta	intrinsic parameters
γ	Γ	gamma	intermediate variable
δ	Δ	delta	variable change, small rotations
ε	E	epsilon	small quantity
ζ	Z	zeta	odometry increment
η	H	eta	projection model denominator
θ ϑ	Θ	theta	rotation angles, thresholds
ι	I	iota	—
κ	K	kappa	intermediate variable
λ	Λ	lambda	various coefficients and quantities
μ	M	mu	mean values
ν	N	nu	—
ξ	Ξ	xi	6 DoF transformation, intrinsic parameters
o	O	omicron	—
π	Π	pi	3.14159...
ρ	P	rho	distance origin-point
σ	Σ	sigma	standard deviation
τ	T	tau	—
υ	Υ	upsilon	—
ϕ φ	Φ	phi	angle, scalar function
χ	X	chi	intermediate variable
ψ	Ψ	psi	scalar function
ω	Ω	omega	angular velocity, image domain

List of Notations

$\Omega \subset \mathbb{R}^2$	domain of an image or its part definition
$I : \Omega \to \mathbb{R}$	image
$\mathcal{D} : \Omega \to \mathbb{R}$	depth map
$\boldsymbol{p} = (u, v)^T$	2D point, element of $\mathbb{R}^2$
$\boldsymbol{X} = (x, y, z)^T$	3D point, element of $\mathbb{R}^3$
M	normal plane defined by $z = 1$
$\boldsymbol{m} \in M$	normalized point
O_i	origin of frame i, used to refer to the frame
$\xi \in \mathrm{SE}(3)$	6 DoF transformation $[t_x, t_y, t_z, r_x, r_y, r_z]$,
$\boldsymbol{t} \in \mathbb{R}^3$	translation vector
$\boldsymbol{r} = \boldsymbol{u}\vartheta$	angle-axis rotation
${}^i\xi_j$	defines O_j with respect to O_i
${}^i\xi_j({}^j\boldsymbol{X})$	transforming a 3D point $\boldsymbol{X}$ from O_j to O_i.
$\xi_a \circ \xi_b$	composition of two transformations
$R \in \mathrm{SO}(3)$	3D rotation matrix
$T \in \mathrm{SE}(3)$	homogeneous transformation matrix
$\mathfrak{f}$	camera projection model
$\mathfrak{f}^{-1}$	inverse projection model
$\boldsymbol{\alpha}$	camera intrinsic parameters
in particular	
f or f_u, f_v	focal length
u_0, v_0	coordinates of the image center
K	projection matrix, contains intrinsic parameters
F	fundamental matrix
E	essential matrix
$a \propto b$	a is proportional to b
$\boldsymbol{a} \times \boldsymbol{b} \quad \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$	cross product of two vectors
$[\cdot]_\times$	cross-product skew-symmetric matrix
$\boldsymbol{p} \times \boldsymbol{q} \quad \boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^2$	is the same as det $\begin{bmatrix}\boldsymbol{p} & \boldsymbol{q}\end{bmatrix}$, or z-component of the cross product

Chapter 1

Introduction

The problem we are addressing in this thesis is how to build a flexible, precise and widely-applicable wide-angle visual perception system for autonomous mobile robots. The main purpose of such a system would be to provide the robot with all the needed information to accomplish the navigation task.

In the following introduction we provide an overview of robotic perception, different modalities, navigation problem description, and also review important bibliographical references on different localization techniques. Using the presented information, advantages and disadvantages of different hardware and methods, we justify the key choices we have made during the work.

1 Robotics and Perception

The purpose of a robot is to efficiently interact with an environment and achieve given goals. In order to do that, it must be able to build certain environmental representation, develop a sequence of actions, which leads to the desired result. Once it is done, the robot has to use its actuators to interact with the physical world, that is, to change its own state or states of other objects around. This Perception-Planning-Acting can be represented as in Fig. 1.1.

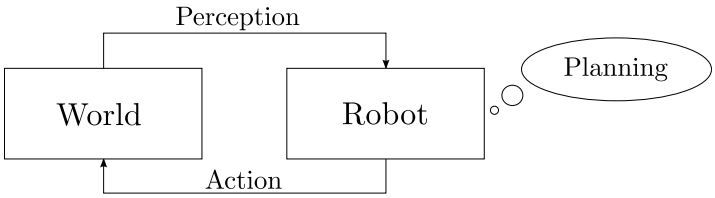


Figure 1.1: Perception-Planning-Action.

We can take a chess robot as an example (surprisingly there are no many of them around). Let's think of it as a manipulator with one or more cameras. We want it to be able to play against humans or other chess robots using a wide spectrum of reasonable chess sets. Let's analyze its task using this paradigm.

1. Perception. First, the robot needs to detect the chessboard, then detect and recognize all the pieces on the board. Then, this information must be represented under two forms. First, it needs a symbolic description of the board state to hand it to the planner. Second, it needs a 3D representation usable for the manipulation purposes.

2. Planning. In this case the required discrete action is a move. So to compute it, a chess engine able to process the symbolic description of the board state and to find the needed move.

3. Action. Once the desired move is compute, a simple pick-and-place operation is required.

Strictly speaking, the control scheme usually includes another planner, which prepares a trajectory, and the real boundary between planning and control in this case is fuzzy. What is important is that the perception is needed for symbolic planner (let's call it this way), trajectory planner, and the motor control loop. The symbolic planning part is currently solved. There are a few challenges in the action part, like grasping objects of different size and shape. But the most difficult and the least solved problem is perception.

Robotic chess players exist. But they are not playing fairly by using a special chess set where the chessboard is a sensor, and the chess pieces are marked with radio chips and whose shapes are precisely known. Once the position of such a board is known in the robot coordinate system, the task becomes almost trivial, because the robot can do everything while being completely "blind". This is not what we want because here the chess set itself is rather a part of the robot, hence there is no real manipulation with external objects.

Of course, we can facilitate the task by sticking a QR-code on every piece, or mark them with infrared reflectors. At least we can use the same chess set every time. There is a continuum of different levels of the chess-play tasks for robots. But intuitively, the majority of us would agree that what we are looking for is a robot which is as capable of using a real chess set as humans.

Speaking more generally, the problem of visual perception is not yet solved. All the contemporary computer vision systems are specific and have a narrow application range with particular environmental setting and limited functionality. A great advance in computer vision would be a system capable of solving a large variety of visual perception problems with little or no changes of its architecture.

Perception System The perception system is composed of different blocks, hardware and software:

- Sensor — a hardware device, meant to transform physical quantities into electronic digital codes. For example, a thermometer measures temperature and outputs its value in binary codes. In the previous example the sensor was a camera, which measures light brightness on the surface of its retina, where an image of the environment is projected by a system of lenses. Even though the data acquired by the sensors is treated numerically regardless of its physical origin, we shall not forget that sensors are the crucial part of any perception pipeline.

- Low-level data processing — a software block which works with raw data values to enhance their quality. For instance, in the case of images, it improves the dynamic range, reduces noise. During this step the data are prepared for further processing.

- High-level processing — at this stage, the data is interpreted, features are detected, the data is usually represented differently, for instance via symbolic information. Additional knowledge about geometric constraints, scene setting, priors, symbolic rules is employed.

This division is given as an example. Depending on the system, low-level processing can be integrated in the sensor, or on the contrary, high-level processing might be combined with low-level denoising algorithms. But the important point is that the perception system contains two important blocks: data acquisition block, or a sensor, and processing block, which actually accomplishes the given task. In the following sections, different sensors are reviewed, their advantages and disadvantages are discussed. Then the navigation problem, as the case of the most interest for us, is considered.

2 Different Sensors and Modalities

Through the evolution process of robotics, various sensors have been developed. None of them are perfect and able to cover all aspects of environment sensing, and each of them has its own "blind spots". Moreover, we have to assume a number of properties of the environment in order to use a certain sensor.

Since this thesis belongs to the domain of mobile robotics, the sensors and, ultimately, the perception systems are considered first of all as localization and navigation tools. And the question we ask regarding each sensor is how to use it for these purposes.

2.1 Proprioceptive Sensors

Proprioception, from Latin *proprius*, meaning "one's own", "individual", and *capio*, *capere*, to take or grasp, is the sense of the relative position of one's own parts of the body and strength of effort being employed in movement, says the Mosby's Dictionary of Medicine, Nursing & Health Professions.

Joint Encoders are the most important sensors for industrial robots since the robot's mechanical state is fully defined by joint angles and velocities. The so-called *direct geometric model* can be applied to compute the 3D pose of all the robot's links and its end effector. For most industrial robots it is the only sensing modality, which is sufficient for their tasks in a fully controlled environment. Encoders give a high angular precision which allows robotic arms to have an excellent repeatability.

Odometry The word *odometry* comes from Ancient Greek ὁδός (hodós, "road, path, way") + μέτρον (métron, "measure"). It is a particular type of localization which is based on integration of motion increments. Classically, it is based on measuring the rotation rate of robot's wheels and computing the instantaneous motion using kinematic constraints of the wheel-ground contact, that is non-skidding and non-slipping. The particularity of this sensing modality is that the localization it gives is consistent only locally. Since every increment has an error, integration over the whole path leads to error accumulation and divergence between the real trajectory and the estimated one.

IMU is usually an integrated circuit, sensitive to gravity, accelerations, and angular velocity. By integrating the accelerations a velocity estimation can be obtained. To get a position estimation another integration must be applied. Any bias in the acceleration estimation causes a drastic divergence of the position estimation, while the orientation is estimated using only one integration and thus is more accurate.

2.2 Exteroceptive Sensors

Exteroception, on its turn, stands for the environment perception. Such sensors give the output as a function not only of the robot state, but of the environmental state as well. Exteroceptive sensors are crucial, since robots are supposed to execute tasks related to the environment, navigate through the environment and interact with it. Without this kind of sensors, robots would be blind and not capable of doing anything in an uncontrolled environment.

The two main categories are *passive* and *active* sensors. Passive sensors work only in the reception mode (for example, cameras, microphones, radio-wave receivers). Among the passive sensors, cameras are of the most importance for the localization and navigation purposes.

The active sensors probe the environment with an emitter and receive the "echoes", or the reflections of the emitted waves. The examples are lidars, ultrasonic range finder, structured-light systems (for example, Microsoft Kinect), time-of-flight cameras. Usually, active sensors directly give us geometric information about the environment. On the other hand, they have limited range and some of them are sensitive to external radiation sources, like powerful spotlights and the sun.

Ultrasonic Range Finder It is the cheapest among active sensors, and is capable of measuring distances within a few meters. It is not precise as the ultrasound beam is not focused, but rather shaped as a cone. These sensors are used to detect that there is an obstacle nearby rather than to reconstruct the environment. Its weak point is that some surfaces are invisible for them because they absorb too much ultrasound, while other surfaces (mostly flat) reflect the beam completely instead of scattering in all directions, and hence are also invisible.

Lidar Laser range finders or lidars (LIght Detection And Ranging) emit infra-red laser pulses and measure the time until the scattered signal is received. Originally lidars measure distance in only one directions. By combining it with a rotating mirror which changes the beam direction, we obtain laser scanners, which can be referred to as lidars as well and which measure the distance to the objects around in multiple directions (usually with 1° or 0.5° angular step) within a single plane. There are models in which the mirror rotates about two axes to get quasi-two-dimensional sweep (see Fig. 1.2); There are also models where more than one beam is used at a time, and so, multiple planes are scanned (Fig. 1.3).

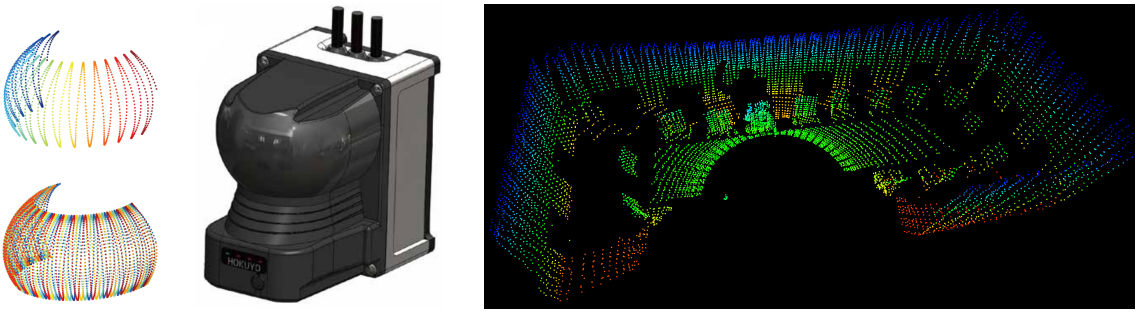
Pros:

- The output is directly geometric data which are generally easy-to-use.
- The precision of distance measures is about 5-10 cm which is relatively high and almost does not depend on the distance.
- Lidars provide a high resolution in the sweep direction;
- Since lidars are active sensors, they don't depend on external signal sources.

Cons:

- Lidars have mechanical parts, such as a rotating mirror, which are fragile and expensive.
- The range of lidars is limited. Usually it is about 50 m.
- Some surfaces are invisible because they reflect the beam instead of scattering it.
- As it is a sweep-based sensor, different parts of a scan belong to different time instants, so if the robot moves fast or there are moving objects in the environment, the motion distortion effect must be taken into account.

Structured-Light Sensors Another possible way of perceiving the depth information is via projecting a certain pattern in the visible part of the spectrum or in its infrared part. One of the most famous and most influential sensors is Microsoft Kinect V1. Microsoft released a low-cost active 3D sensor, which gives an image along with a depth map (so-called RGB-D camera). The 3D reconstruction is based on projecting an infrared pseudo-random dot pattern on the environment. An infrared camera together with the emitter form a stereo pair. Pseudo-randomness of the pattern makes the matching process efficient and robust, so



(a) (b)

Figure 1.2: (a) Hokuyo YVT-X002 and its sweep patterns. (b) A laser scan taken with this model.

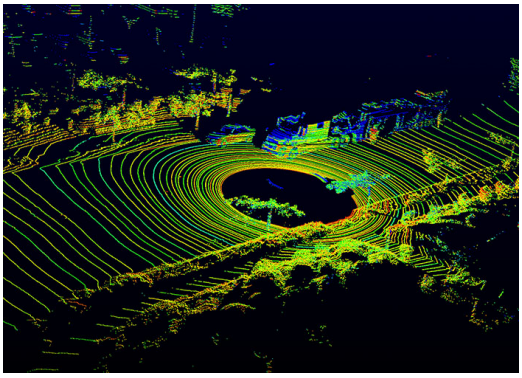


Figure 1.3: A laser scan taken with Velodyne HDL-64E

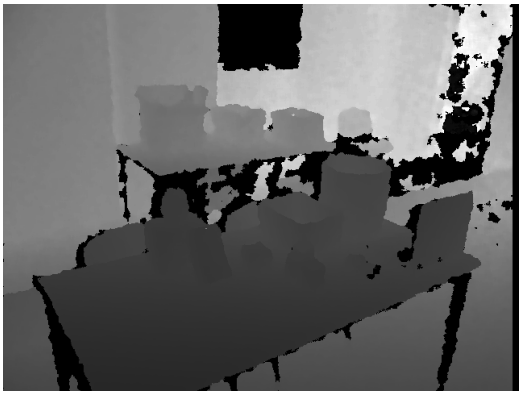


Figure 1.4: A depth map generated with Microsoft Kinect.

generally the reconstruction quality is good. Unfortunately this sensor is sensitive to sunlight as it shadows the projected pattern, so the sensor is usable only indoor. One should keep in mind that this sensor has been developed for gaming purposes, and thus is not perfectly adapted for scientific research. A depth map example is given in Fig. 1.4.

The concept of structured light can be applied to omnidirectional sensors. For example, Orghidan *et al.* (2006) describes a 3D sensor based on a conic-mirror laser projector and a catadioptric visual sensor, which are aligned with each other. This sensor is able to capture omnidirectional depth images.

Time-Of-Flight Camera Another type of active visual sensors uses a principle similar to lidars. It emits a light pulse and measures the time between the emission and reception, or the time of flight (ToF). The sensor is organized in the same way as classical cameras, that is it has a retina and a lens. So the output is a depth image with all geometric properties of images. For contemporary ToF cameras the resolution is lower than for classical ones, for instance, Microsoft Kinect 2 has a ToF camera with a resolution 512×424 px and a range of 4.5 m. This technology is one of the most recent depth sensors and it seems to be a good solution for indoor environments, but for outdoors it might have the same problems as any other active optical sensor, that is, in bright sunlight the depth measurements become noisy or completely broken.

GPS This modality stands aside because it is not an actual environment sensor, but rather a giant artificial absolute positioning grid. It has the advantage of global consistency, that is,

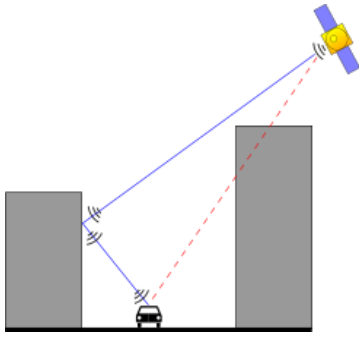


Figure 1.5: Multipath effect. Because of signal reflection the real distance to the satellite is different from the perceived one.

wherever you are on Earth, you can localize yourself up to few meters, under the condition that you capture signals from a sufficient number of satellites (at least four).

But this system has a certain number of issues. First, it does not work indoor because the satellite signals are screened by the building. Second, in urban areas, high buildings reduce the satellite visibility and cause the so-called multi-path effect (see Fig. 1.5). It significantly reduces the localization precision and other modalities have to be used in order to make it possible to navigate. For example, in Kos *et al.* (2010) it is reported that multipath produces additional localization error of about 8 m.

Depending on the application, GPS might not be sufficient as a source of localization data. There is no reason not to use GPS for localization purposes to simplify map-based localization by giving an initial approximation. But in some cases, it cannot be the primary modality in the localization and navigation process because of the following reasons:

- This localization does not give any information about the obstacles and navigable space.
- Even if we had an exact map with all the obstacles and roads, which is impossible because the environment changes over time, the localization is not precise enough. For instance, 5 m precision allows you to know which street you are in (in most cases), but does not allow you to know which lane you are driving in.

3 Navigation and Mapping

Even though this work does not treat the problem of navigation directly, its global context and the ultimate goal is a visual navigation system for mobile robots and self-driving cars. That is why a brief overview of different navigation techniques is provided.

A common definition of *navigation* attributes it the following functions:

- Localization
- Path planning
- Path following and obstacle avoidance

Localization is arguably the most important and the most challenging part of the navigation process. That is why we are primarily interested in different mapping techniques. Bonin-Font *et al.* (2008) suggests the following navigation taxonomy:

- Map-based — the localization is done using an existing map
- Map-building — the system is capable of building its own maps while exploring the environment and then reuse the maps for the navigation purposes

- Map-less — there is not an explicit map and explicit localization, even though in most cases there is a certain environment representation used for navigation

We would add that since the navigation task is usually defined with respect to a map, the map-building navigation systems are extensions of map-based ones in the sense that the map-building systems are able to build a map during the training phase and then operate as a map-based system.

We can imagine a task where the environment is not known and the goal is defined in global coordinates. Unless the environment has a simple structure, like a flat area with a number of convex obstacles, or unless its prior map is given (a street map, for instance) it is an exploration task, which goes beyond the scope of this work.

3.1 Map-Less and Learning-Based Navigation

Reactive Navigation One of the most basic approaches to navigation is the so-called *reactive* navigation. Its objective is to follow a certain structure in the environment, the most commonly, a corridor. Some systems are biologically-inspired. For example, Santos-Victor *et al.* (1995) describes a system based on bees' flying behavior. It compares the left and right optic flows to figure out on which side the obstacles are closer. At the center of the corridor the optic flow is balanced on both sides and if the robot is closer to one of the walls, the corresponding apparent motion will be faster and the implemented control law will make the robot go away from it. Such navigation systems must be combined with a higher level planner, which can change the behavior from "follow the corridor" to "turn to the left" for example.

Learn and Repeat In the case of a more meaningful activity like path following, a usual approach is "learn and repeat". One possibility to learn a trajectory is to store visual input along it. The hypothesis is that the same visual input during the repeat phase corresponds to the same trajectory. Of course, repeating the exact same trajectory and having the same input is virtually impossible. But we can try to get as close as possible. This method uses the so-called vision-based control or *visual servoing*. Instead of controlling the robot position directly, we control some features of its visual input. In an early work by Jones *et al.* (1997), the zero-normalized cross-correlation (ZNCC) is used to associate a live-stream-video frame with an image from a database. The control loop drives the cross-correlation peak to the image center to make sure that the robot moves in the right direction. Once the next image in the trajectory matches the video stream better than the current one, the system switches to it and the process continues.

A more elaborate technique is to use different visual features instead of ZNCC. The robot minimizes the difference between the perceived feature and the ones extracted from the key frame. Once the perceived features' parameters are close enough to the desired ones, the robot takes the next keyframe as the reference. For example, line segments can be used as visual features, as it is suggested in Raj Bista *et al.* (2016a). Another option is to use dense visual information. A system based on the so-called mutual information has been presented in Raj Bista *et al.* (2016b). During the learning phase, the robot acquires key images along the trajectory. During the execution, the robot chooses the key image which has the most mutual information with the actual perception. Then the control is computed so that the mutual information between the two images increases. The maximum of mutual information corresponds to the coincidence of the two viewpoints. This method has the advantage that the mutual information is robust with respect to lighting changes and small modifications of the environment.

Machine-Learning-Based Navigation A method of map-less navigation based on omnidirectional vision and neural networks has been proposed by Giovannangeli *et al.* (2006). Along with a growing popularity of deep learning and convolutional neural networks, end-to-end learning has been applied to the navigation problem. An outstanding result of deep learning has been demonstrated by Mnih *et al.* (2015), where a neural network learns to play video games at a super-human level based only on a video input, using the so-called deep reinforcement learning (DRL). DRL has been applied to the visual navigation problem. For example, Zhu *et al.* (2017) applied DRL to the task-driven navigation problem. The input is the current image and the goal image, the output is a high-level action from the list "move forward", "move backward", "turn left", "turn right". Even though there is no explicit map of an environment, there is an environment-specific layer which has to be pretrained for each newly seen environment, while the most part of the network remains intact.

An idea of augmenting the learning-based system with a map representation has been used in Bhatti *et al.* (2016). The work presents a learning-based artificial intelligence meant to play Doom, a classical video-game shooter. Even though the environment and the goals are different from the real world, the results can be generalized for real robotic applications. The authors combine computer vision for mapping (more exactly, ORB-SLAM by Raúl *et al.* (2015)) and deep neural networks for policy learning to create an agent capable of building and executing a sequence of actions, based only on its visual input.

Another recent work by Gupta *et al.* (2017) describes an end-to-end-learning-based navigation and mapping system. This system exploits the same idea of building and keeping a map of the environment in the memory, but uses a fully learning-based approach for its reconstruction instead of visual SLAM techniques. Even though learning-only systems produce interesting results, it seems to be a better choice to integrate the learning part, which accumulates different experiences and priors about the environment, with existing computer-vision algorithms to solve certain specific problems, for example geometric modeling of cameras.

3.2 Map-Based Navigation

Map-based techniques require a map for localization and path planning purposes. This map can be preconstructed by the system itself during the pretrainig or exploration phase. The navigation system continuously runs its localization loop and, based on the robot's pose, it computes and executes a path to the goal, both defined in the map.

Localization Algorithms State-of-the-art methods of map-based metric localization are based on the probabilistic framework. The robot's position is represented by a probability distribution and its update is usually divided into two steps:

1. Prediction — the robot position belief is updated using proprioceptive sensors, control values, and/or dynamic model.

2. Measurement — the belief is modified in order to maximize the probability of the data given its prior distribution and measurement noise parameters.

Such an approach has been proven to be efficient and robust to a certain extent. For example, it has been used by Burgard *et al.* (1996) for indoor localization of a mobile robot with sonars as distance sensors. The localization belief is represented using a fine grid in the 3D posture space (x, y, ϑ).

This system has been improved even further in Fox *et al.* (1998) by adding an outlier detection module in order to make it possible for the system to operate in highly crowded environments like a museum exposition (where this system has been successfully tested).

More recent approaches use different belief representations. One of them is the so called particle filter (Dellaert *et al.* (1999)). Instead of representing the belief by a uniform grid, a certain number of particle are scattered across the map. At every prediction step each particle generates several children which evolve according to the probabilistic prediction model. At the measurement step, the particles are sampled with probabilities defined by the data likelihood. This method is well-adapted to low-dimensional spaces and is able to represent arbitrary distributions.

Another method is called extended Kalman filter (EKF), which is a generalization of the Kalman filter for non-linear systems. In this approach the noises and state belief are modeled using Gaussian distribution. Even though this method is limited to unimodal distributions, it is popular because in some cases there is no need of multi-hypotheses approaches, and EKF is relatively simple to implement and adjust. Also it can be used with high-dimensional data since its complexity is a polynomial function of the dimension, while the particle filter grows exponentially.

Simultaneous Localization and Mapping If the environment is unknown, its map has to be constructed. It can be done via Simultaneous Localization And Mapping (SLAM), a process which constructs a map from scratch and localizes the agent in it, or more generally reconstructs the agent's trajectory with respect to this map. Constructed maps may have different representations. It can be represented by points or other geometric primitives, all put in the same *local map frame*. For example, a map can be represented by a point cloud, which is a common choice for 3D sensors, like Velodyne. Another popular option is the so-called *occupancy grid*, a bitmap which represents free space and occupied space. The occupancy grid is usually used with 2D laser scanners (for example Wang *et al.* (2007); Dominguez *et al.* (2015)). This kind of maps is a natural choice for active metric sensors, like lidars. A single lidar scan can be considered as a local metric map, and by combining multiple scans, it is possible to construct a precise and detailed map.

An efficient algorithm to process point clouds, called *iterative closest point* (ICP), Besl and McKay (1992), is usually used to register different scans. Registration means bringing two datasets to the same reference frame. In other words, registering two scans means "aligning" them, or finding a transformation between the origins of both scans. ICP works for both 2D and 3D scans. After registration, the map can be extended. It is done either by adding more primitives to the map, that is, points, lines, planes; or by changing values in the occupancy grid.

In the case of vision-based SLAM, the classical approach is to use visual features as landmarks and construct 3D feature point clouds. Some systems maintain the whole map in a single coordinate system. For example, Beall and Dellaert (2014) updates the map by adding new features which correspond to different seasons. It selects only features which can be observed from the neighborhood of the current position to match them with perceived ones. Another prominent example of feature-based SLAM, already mentioned before, is ORB-SLAM by Raúl *et al.* (2015).

If the map is getting too big, multi-map systems can be used. That is, instead of using one single coordinate frame for the whole map, instead multiple map patches, each one having its own frame, are stitched together. A particular case of multi-map system is a keyframe-based system. In this case a single sensory input is used as a local map. Newly arrived sensory inputs are registered (that is, brought to the keyframe coordinate system) and then used to improve the local map precision. Examples of such systems are Engel *et al.* (2014); Biber and Duckett (2005).

The concept of semi-dense reconstruction, introduced by Engel *et al.* (2013) is of a particular interest for us. Instead of reconstructing a dense depth map (like the ones given by

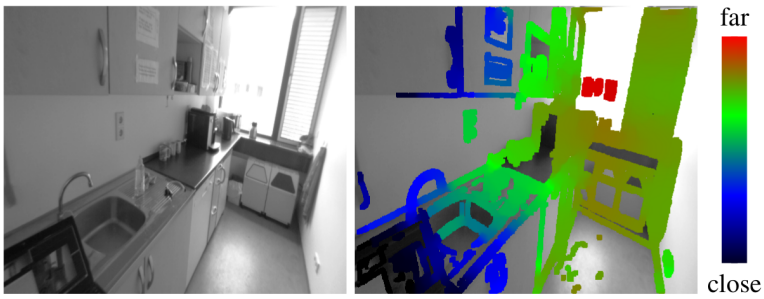


Figure 1.6: Semi-dense reconstruction. Only textured regions are processed. Engel *et al.* (2013)

Kinect — Fig. 1.4), which is computationally expensive and, in the case of passive sensors, requires strong priors on the texture-less regions, the reconstruction is done only for pixels with a certain level if image gradient (Fig. 1.6). That way, the maximum amount of image information is used while the computational cost remains at the level of sparse feature-based approaches. Still, this approach assumes a static environment and brightness constancy.

Another research direction is related to localization in an environment which changes over time. These changes are caused primarily by the weather and lighting conditions: cloudy days, bright sun light which casts sharp and high contrast shadows, nighttime with artificial lights, snow, rain, and so on. One approach which can address the environmental changes is presented in Churchill and Newman (2013). The underlying idea is to learn all the possible place appearances. To do so, every time the vehicle passes by a place, it is trying to localize itself with respect to all prior experiences, which are stored outputs of a visual odometry system constantly running on board. If the number of successful localizations with respect to the existing experiences is less than a certain small threshold, the system assumes that the current representation of the place is not sufficient and the current experience must be stored. Some places require much less experiences than others, depending on how much their appearance changes over time.

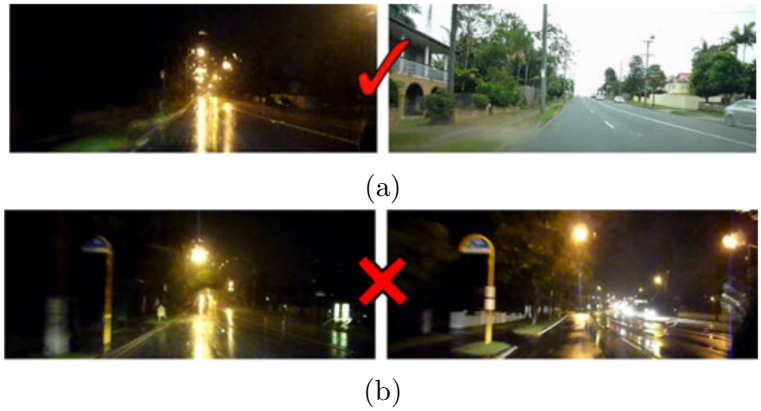
One more way to deal with environmental changes, presented in McManus *et al.* (2015) is to learn reliable and repeatable mid-level features, specific to a certain place. These features don't carry any semantic information, but they are of a higher level than simple point-like features which, while being a nearly optimal solution in the general case, might not be appropriate at all for a particular place.

Loop Closure An important aspect of a SLAM system is the so-called *loop closure*. The idea is to detect the fact that the robot revisits the same place as before, create a loop in the map, and re-optimize the map taking the new constraint into account. Loop closure has two important aspects:

1. Topological. As it is mentioned in Cadena *et al.* (2016), without loop closure the robot lives in an infinite corridor. And if we want to reuse the constructed map to go from point A to point B, the robot will have to follow its original path instead of taking possible shortcuts.

2. Metric. Such constraints have a great impact on the metric map precision and global consistency. Accurate loop closure significantly improves the map, while false ones may completely deteriorate it.

In some cases, loop closure is not necessary. For example, if the robot has to map and to be able to navigate along just a single path, then the "corridor" model is perfectly suitable. But generally, for smart autonomous navigation, being able to reconstruct and use the environment topology is important.



(a)

(b)

Figure 1.7: Visual place recognition systems must be able to (a) successfully match perceptually different images while (b) also rejecting incorrect matches between aliased image pairs of different places. Lowry *et al.* (2016)

3.3 Topological Maps and Place Recognition

A particular place in the navigation and cartography is taken by the so-called visual place recognition. The problem is formulated as follows: given a visual input acquired at a certain spatial location and a set of already visited and mapped locations, determine whether the perceived location corresponds to one of the previously observed and mapped places or is observed for the first time. The key point here is that we are not interested in how similar the visual input is to what we have seen before, but how spatially close the viewpoint to the viewpoints corresponding to the prior observations. Due to the change in illumination conditions and weather, the same place may look differently and two different places might look similar (Fig. 1.7).

An influential paper by Nistér and Stewenius (2006) presents a system which allows us to retrieve similar images from databases of over one million images in real time with little time needed to add images to it. The principle is to create a pretrained vocabulary tree to quantize feature descriptors into binary words and apply inverted-document system to retrieve relevant images. This system has been reused by the authors to implement vision-based topological navigation system (Fraundorfer *et al.* (2007)). Even though the similarity in visual appearance is taken as the criterion for place matching, the proposed method of data retrieval might be considered as a breakthrough.

Later, another influential paper on place recognition and topological localization by Cummins and Newman (2008) came out. It presents a localization system called FAB-MAP, which also uses the concept of vocabulary and descriptor quantization, but adds on top of it a statistical model to take into account cooccurrences of different binary words and computes the normalization term to be able to tell whether the perceived image is observed for the first time or it corresponds to a place from the memory.

Loop closure is done via place recognition in the case of visual SLAM. But any place recognition system is a mapping system by itself. It has to construct and maintain a certain world representation, including or not metric and topological information. Hereafter topology of a map means that for a set of locations (usually finite) represented by this map there is a set of edges which connect "neighbor" locations. These edges represent reachability of one locations from the other. Usually it is an integrated part of a bigger navigation and mapping system. FAB-MAP is *de facto* the most commonly used visual loop closure system.

Some authors (for example McManus *et al.* (2015)) use place recognition and topological localization interchangeably, in the sense that instead of getting precise metric localization with respect to a local landmark or a global map, the system determines that the robot is

at a certain "place", with a vague definition of "place". We would argue that topological localization is somewhat more specific system than a place recognition system, because place recognition does not imply any topology which binds all the places together. In contrast, a topological map provides a richer information on how the places are related in space. The most intuitive way to link the locations is via spatial transformations between them. Global Euclidean consistency here is not an issue. That is, two different topological paths from place A to place B might not sum up to the same euclidean transformation. But what is important is that, locally, the transformations have reasonable precision.

One important implication of a topology imposed over a set of places, as it is mentioned by Cummins and Newman (2008), is that the localization does not have to be done globally every time. Instead, we can reuse the previous localization belief and a motion prediction model to reduce the search space of possible current places.

4 Motivation and Manuscript Structure

In this work, we would like to go through different layers of visual perception based on fisheye cameras and geometric vision, and to establish a solid basis for high-level visual perception algorithms. First we summarize different aspects of vision systems, their advantages and existing problems. Then the principal design choices are explained and the manuscript structure is described.

4.1 Why Vision Is So Attractive

There are a lot of different arguments on why using cameras and images instead of other sensory modalities. We would mention the following ones:

1. Cameras are practical exteroceptive sensors. They are cheap, small, light, they don't have moving parts and thus are mechanically robust. And there is a vast variety of different models with different frequency, resolution, noise level, field of view, and energy consumption.

2. Vision provides dense and rich information about the environment. It is an exteroceptive modality which can be used for all kinds of robotic tasks, including localization, navigation, object manipulation.

3. Since we ultimately want to integrate robots into human society, they have to understand our signal systems. A vast infrastructure with visual signs and markups already exists, and it provides valuable information about how to navigate safely and reach the destination.

4.2 Why Vision Is Difficult

As it will be described later on, image formation is a complicated process, which includes multiple unknowns. Let's consider some of the challenges of computer vision from low- to high-level. The first of them is the fact that cameras are passive sensors, hence visual perception depends on lighting conditions. It means that algorithms have to be robust with respect to lighting changes. Another aspect related to the lighting is the narrow dynamic range that consumer cameras currently demonstrate. It means that it is virtually impossible to have a good contrast in shadows during a sunny day, and at the same time keep the objects in the sunlight unsaturated. But this is more of a hardware issue and, hopefully, sooner or later the dynamic range of digital cameras will attain or even overtake the one of human eyes.

One more difficulty is the fact that the environment has to be textured. Actually, even if we paint everything in white, there still will be different shades and gradients. But what is usually required by modern algorithms is a high-contrast texture, and moreover particular feature detectors require particular types of structures, like corners, straight lines, circles, ellipses and so on. If the environment lacks this king of features, its images become significantly more difficult to process. Dense and semi-dense techniques which treat pixels individually seem to be a good solution to handle structure-less scenes. But so far, there is no widely-known adaptive algorithm which is as good with high-contrast texture as with smooth gradient shading, which again evokes the problem of a universal perception system able of optimally selecting different kinds of information from images.

While geometric sensors (for example lidars) measure directly the distance to an obstacle in a given direction, cameras generate images, tables of numbers which represent the intensity of light coming from different directions. Geometric information (in this case, distance measurements) is much easier to interpret and use. On the other hand photometric information is almost useless by itself. Images have to be intensively processed in order to match the information that they contain, register them, reconstruct the environmental 3D structure, to obtain semantic segmentation (that is regroup pixels according to the category of the object they belong to), or to localize the robot based on what it has seen before. This mid- and high-level processing is currently the bottleneck of vision algorithms.

Yet another difficulty is occlusions and visibility. Actually, occlusions and the fact that some object can be present on one view but not on the other makes the exact correspondence problem intractable because it becomes combinatorial, as it is mentioned in Kolmogorov and Zabih (2001), and thus only approximative algorithms (yet quite efficient for real-world data) have been proposed in the literature. Perhaps, the problem of occlusions must be solved at a higher, semantic level, incorporating knowledge about different objects, their structure and typical behavior.

4.3 Principal Choices

During this work the following ideas were used as guidelines. We repeat and summarize what has been said before about different vision systems.

Wide-Angle Cameras Omnidirectional and wide-angle images provide much better localization properties than classical pinhole cameras. They are used in multiple systems and many authors have mentioned the advantages of wide-angle vision for localization purposes (for example, Bonin-Font *et al.* (2008); Milford *et al.* (2004); Giovannangeli *et al.* (2006)). Multiple papers on omnidirectional localization algorithms have been published recently (for example, Meilland *et al.* (2015); Caruso *et al.* (2015)) In the case of omnidirectional images we are sure to see the same features from the same location even if the orientation is different. Wide-angle cameras (more specifically fisheye cameras with 180° field of view) don't have this strong property, but still the same parts of the scene are being observed longer during rotational motions, which allows us to have a wider localization base in such cases. Fisheye images are also better suited as keyframes, since they include more information and don't require close angular alignment of the current view and the keyframe's view during localization within sensory maps. Two fisheye cameras are enough to cover almost 360° (the vision scheme that most of herbivores have). But all these advantages come at the cost of a complex geometry of fisheye images. In order to be able to use fisheye images as easily as classical pinhole ones, a new model, which is analytically simple and yet expressive enough to approximate a wide range of wide-angle visual sensors, is required.

Geometric Vision In our strong belief, direct geometric modeling of cameras is a better solution than neural-network based projection approximations. Generally speaking, if something can be solved using a specialized technique, it should be done this way. Deep learning must be applied in the case of big data and when the model behind is unclear. In the case of cameras, geometric modeling and geometric constraints greatly increase its precision and accuracy. Epipolar constraints, for instance, significantly reduce the search space in the case of stereo correspondence, but also allow us to check whether different data in a dataset are in consensus with one another. Finally, precise reconstruction requires geometric modeling along with reliable matching algorithms.

In other words, properly modeled and calibrated cameras become a precise and efficient metrological tool, which allows us to measure shapes, distances, velocities and other spatial parameters.

Raw Image Data Classical computer vision and image processing schemes have long filtering pipelines. But since any filtering leads to information loss, we advocate the approach of direct image processing. It means that the input of computer vision systems is as close to the original signal as possible. For example, in the case of stereo correspondence, the undistortion and rectification procedure, which is the standard approach to this problem, leads to breaking the uniformity of image noise and information distribution across the image. On the other hand, direct methods require a more complicated geometric processing. So, one of the goals of this work is to develop a geometric framework which makes direct image processing for stereo correspondence possible.

Sensor Fusion Using different modalities and decorrelated measurements reduces noise impact and increases overall system robustness with respect to outliers. One of the most popular sensor fusion scheme uses IMU along with cameras. Another interesting option is fusion between wheel odometry and visual modality. The odometry, while having a worse rotational precision than IMU, gives a better distance estimation, because only one integration is needed over two for IMU-based translation estimation. And distance measurements play a particular role for visual perception since cameras do not provide metric information, so another modality is needed to estimate the so-called scale factor. On possible solution is to use a calibrated stereo pair, but almost the same result is achievable with only one camera and wheel odometry.

4.4 Manuscript Plan

The manuscript structure reflects the bottom-up nature of the work development and the final system architecture. Once this implementation process is done, it is possible to analyze the whole structure using the top-down view. The general problem is visual localization. We use the choices described above and formulate the following requirements: the system should combines wheel odometry data with direct fisheye image registration. Firstly, the camera-odometry system must be calibrated. It requires, in turn, a calibration methodology and a camera model. Secondly, direct registration requires depth computation or, in other words, stereo processing. As it is stated before, the stereo correspondence should be directly in the distorted image space to avoid additional image filtering. All these ideas lead us to the manuscript structure, depicted in Fig. 1.8 and described in more details below. This structure represents the work in the direction of increasing complexity.

Chapter 2 is dedicated to geometric modeling of wide-angle visual sensors. First, an overview of visual sensors is provided. Then we introduce the Enhanced Camera Model,

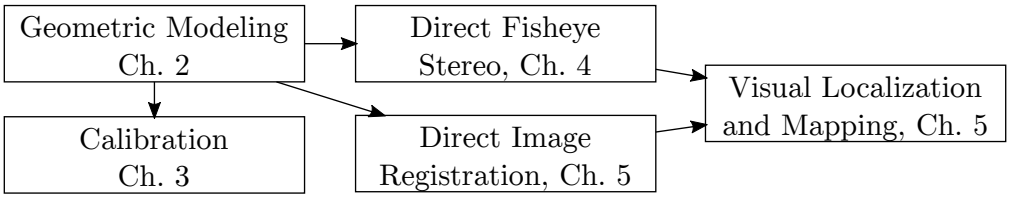


Figure 1.8: Work structure.

based on the Unified Camera Model, and study its geometric properties. In particular, the inverse mapping, straight line projection and epipolar geometry.

Chapter 3 is dedicated to the calibration toolbox developed during this work. We describe the calibration methodology and different aspects of calibration data acquisition. A description of an efficient calibration board detector is given. Monocular, stereo, and extrinsic calibration for mobile robots and wide-angle cameras are addressed. The camera model introduced in Chapter 2 is used and shown to give precise calibration results for real lenses with different distortion.

Chapter 4 describes a direct fisheye stereo correspondence algorithm which combines the epipolar geometry of the Enhanced Camera Model and a rasterization algorithm to compute pixel-disparity-based correspondence cost. Then the Semi-Global Matching can be applied to regularize the depth estimation.

Chapter 5 is dedicated to visual localization and mapping. First, a combined use of wheel odometry and vision to get a better precision and accuracy is advocated. Then we describe a visual odometry system which uses wheel odometry measurements as a prior estimation, which in turn allows us to use less feature points for the RANSAC procedure. Also a semi-dense visual-memory-based localization system is introduced and its performance tested on simulated and real data.

Chapter 2

Geometric Camera Modeling

1 Introduction

Looking at nature, we can see that vision is, potentially, the most efficient and flexible sensory modality. It has a high spatial resolution and a sufficient time frequency. On the other hand, visual information is not usable under its raw form. It has to be intensively processed to extract both symbolic and numeric information about the environment.

Image formation is a complex process. We can see it by looking at its modeling — contemporary 3D graphics. The color of a certain pixel depends on which point of which surface is mapped to this pixel, at which angle the surface is observed, what is the direction and the strength of the light at the point, what is the color and the material of the observed object. All these factors define the light wave which passes through a lens and focuses on a *retina* (a planar grid of light sensitive elements). The quantity of light passing through the lens is controlled by the aperture size. The retina integrates the light intensity over a certain time period, called exposure. Then the integrated value is read out by an electronic circuit and stored in the memory (Fig. 2.1).

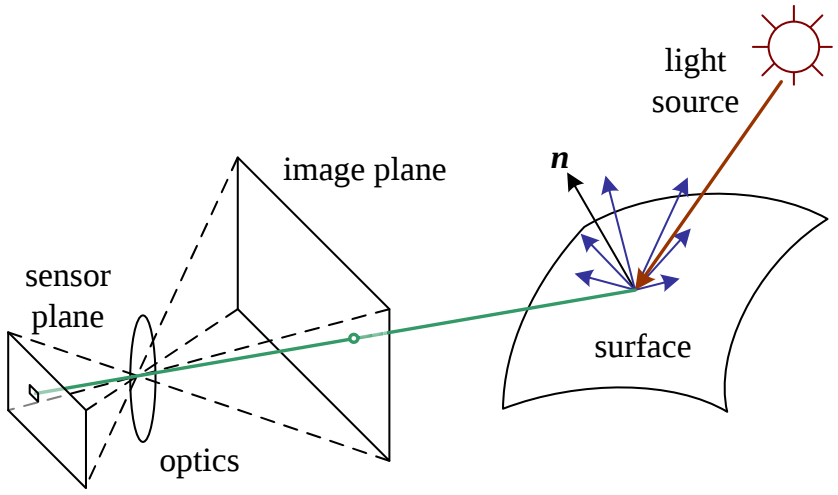


Figure 2.1: Illustration of image formation process. Szeliski (2010)

The obtained image is a 2D array of values, also called pixels. There is a coordinate system attached to it. Its origin is in the upper left corner. The horizontal axis is called u, the vertical one is v (see Fig. 2.2).

In order to produce colored images, colored filters are arranged on the retina in a certain manner (for example, see Fig. 2.3). But color comes at the cost of a lower resolution. In the given example, Bayer mosaic, the ratio between elementary light sensors of red, green, and

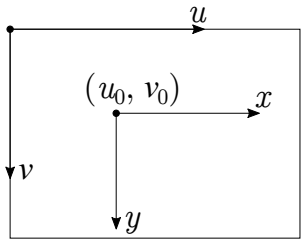


Figure 2.2: Coordinate system in images. Axes u and v are used to represent position of pixels and points in the image; $(u_0, v_0)^T$ is the image center, which is usually associated with the normal plane coordinate system xy, which is metric.

blue colors is 1:2:1, which means that a pure blue object will have 2 times lower resolution as if it were observed by a grayscale camera (4 times fewer pixels).

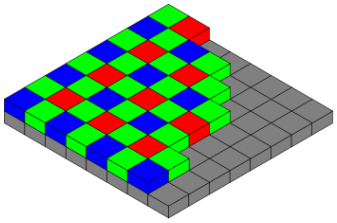
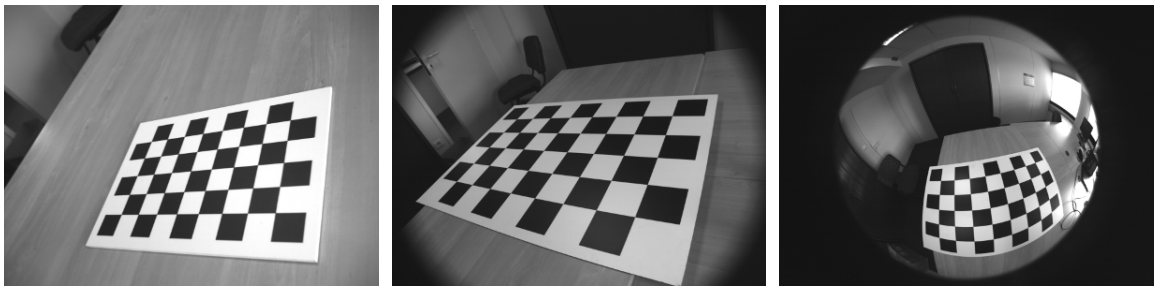


Figure 2.3: Bayer filter mosaic.

Different Lenses and Projection Geometries As later we talk about different projection models and distortions, to give an idea of the subject, a few different real images of a calibration board are presented in Fig. 2.4. There is a trade-off between distortion and field of view. Narrow-angle lenses produce images with a simple geometry (Fig. 2.4a). If we increase the field of view, it will necessarily introduce nonlinear distortion. Fig. 2.4b represents a compromise between the two effects. We can go even further and have a full hemisphere field of view with a single camera (Fig. 2.4c). It comes at the cost of a high distortion which makes the geometry processing for such images a non-trivial problem. This chapter addresses primarily the issue of geometric modeling of such projections.

Another important trade-off is the angular resolution. Since all the presented images have been taken with the same resolution, the angular resolution is different. If we want the camera to observe a distant object, then a narrow-angle lens are the way to go. In contrast, fisheye lenses are better in observing nearby objects and an immediate surrounding.



(a) (b) (c)

Figure 2.4: Images taken with the same camera equipped with different lenses. (a) Narrow angle 8 mm pinhole lens. A smaller calibration board than in the other images has been used to make it fit into the image. (b) Wide-angle 6 mm lens. A small distortion is already apparent. (c) Fisheye 1.8 mm lens. All the straight lines are curved in the image. The field of view is 185° .

2 Projection Geometry

In this section we talk about mathematical camera projection models. These models are purely geometric, they don't take into account such properties as color or light intensity, or wave optics effects.

Viewpoint Usually, when we formulate a projection model, we assume that the system is *single-viewpoint*. It means that the whole scene is observed from a single point, and there is a *diffeomorphism* (continuous differentiable mapping) between image points and directions from the viewpoint. The viewpoint is also called *center of projection*. In other words, all rays mapped to the image intersect in the same spatial point.

The single-viewpoint constraint is closely respected by perspective narrow-angle cameras. It gets less and less true, as the field of view gets larger. Baker and Nayar (2001) describes that it is possible to make a single-viewpoint catadioptric system using a parabolic mirror and a *telecentric camera* (that is, a camera whose effective pinhole is infinitely far away), or a hyperbolic mirror and a perspective camera (Fig. 2.5). But as it is mentioned in Swaminathan *et al.* (2001); Schönbein *et al.* (2014), the single-viewpoint property is sensitive to assembly precision and reduces the flexibility of the optical system design. On the other hand, relaxing this constraint allows us to get more freedom in optical system design and get a better field of view and information distribution in the image. Moreover, none of fisheye cameras are single-viewpoint. For such systems the notion of viewpoint is replaced by the notion of causitc surfaces, analyzed in Swaminathan *et al.* (2001). A caustic surface contains all effective viewpoints for infinitesimal solid angles around all the observed directions.

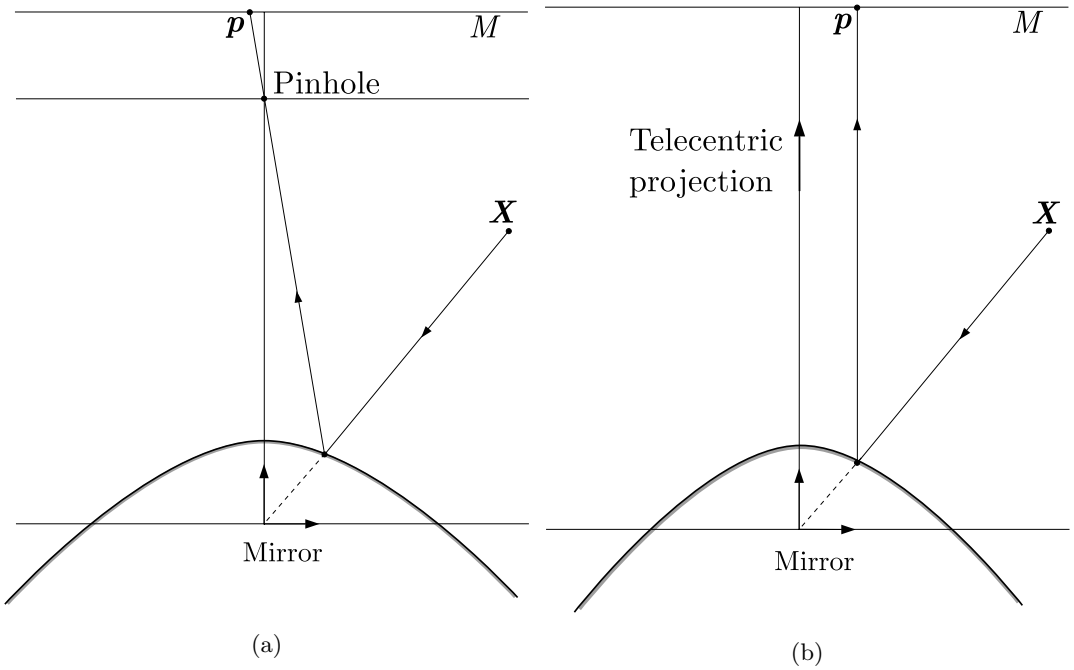


Figure 2.5: Single viewpoint catadioptric systems. (a) Hyperbolic mirror and pinhole camera. (b) Parabolic mirror and telecentric camera. $\boldsymbol{X}$ is a spatial point, $\boldsymbol{p}$ is its projection onto the image plane M. Baker and Nayar (2001)

Even though real cameras are not single-viewpoint, this assumption is important and allows us to come up with simple and yet accurate projection models. Also, it allows us to use simple two-view geometry assumptions. Hereafter we assume that all the cameras we consider

are single-viewpoint. Of course, it is just an approximation but, as the practical results show, a reasonable one.

Image and Space Coordinates Every camera has an optical axis (usually z) which is associated with the direction perpendicular to the image plane. This axis is defined by the lens structure, as every simple single lens has one and an assembled lens has all the optical axes of all its components aligned. Conventionally, x and y axes are aligned with the image coordinate system u, v where units are pixels (Fig. 2.2). The intersection of the optical axis and the image is called *image center*.

Usually the projection is modeled in two steps. First a 3D point is mapped onto the normal plane M (also called intermediate plane or projection plane), which is perpendicular to the optical axis and defined by $z = 1$; Then the normal point $\boldsymbol{m} = (x_m, y_m, 1)^T$ is mapped into the image coordinate system using an affine transformation, represented by matrix K, which usually has the following form:

$$K = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{pmatrix} \tag{2.1}$$

Here u_0 and v_0 define the offset between the image center and the origin of the image coordinate system (Fig. 2.2). f_u and f_v define the focal length, measured in pixels. Sometimes one more parameters, the so-called skew factor s, is added:

$$K = \begin{pmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \end{pmatrix} \tag{2.2}$$

Sometimes, on the contrary, the number of parameters is reduced to three, by assuming that the pixels are perfectly square and saying that $f_u = f_v = f$ (all three are measured in pixels):

$$K = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \end{pmatrix} \tag{2.3}$$

The model should be adapted to a particular camera model. If a simpler version of the model does not give a sufficient calibration precision, a more complex one should be tried out. In this work we use K with four parameters but we don't rely on this fact. Moreover, the results can be generalized straightforwardly.

2.1 Pinhole Camera Model

The first images during human history have been obtained using so-called pinhole cameras. It was just a box with a semi-transparent screen on one side and a small hole in the middle of the opposite side. This system produced dim upside-down images because every point was getting only as much light as could pass through the pinhole. It was a single-viewpoint system with the pinhole playing the role of the viewpoint, or the center of projection. Every ray coming into the pinhole was mapped to a certain point on the image depending on its direction (Fig. 2.6). Here the pinhole worked as a spatial filter.

The simplest camera model is represented by the following projection relation (see Fig. 2.7):

$$\boldsymbol{p} = \frac{1}{z} K \boldsymbol{X} \tag{2.4}$$

there are only 4 parameters, the components of matrix K. They are called *intrinsic parameters* because they are defined by the geometry of the camera itself and do not change from one image to another and from one camera position to another. In the context of camera models we will refer to the intrinsic parameter vector as $\boldsymbol{\alpha}$.

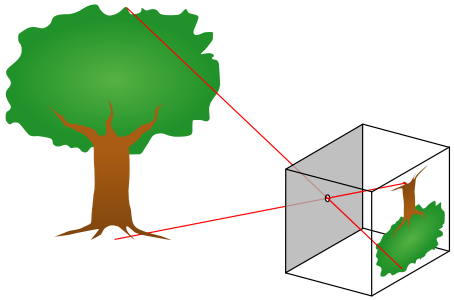


Figure 2.6: Pinhole camera.

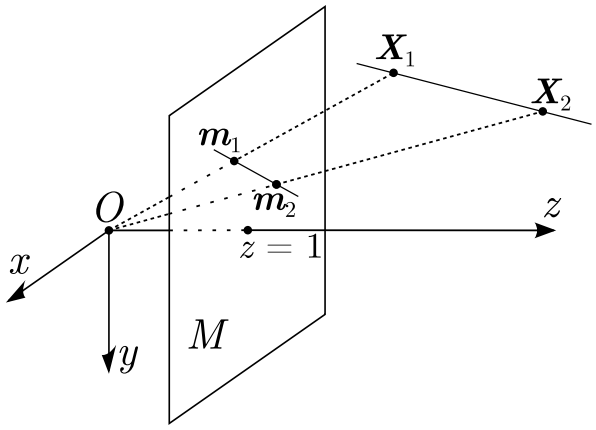


Figure 2.7: The pinhole model illustration. M is the normal plane defined by $z = 1$; by normalizing z-coordinates of spatial points $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ we project them to $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, then the normal points are transformed into the image frame using an affine transform. The straight line, defined by the spatial points will be projected to the straight line defined by the normal points.

You can see that the "perspective" effect is captured by the model. The farther away the object, the smaller its image, because its size is divided by z.

Even though this model is easy to analyze analytically, it has a certain number of drawbacks, for example, a limited field of view. Indeed, there is at least a constraint $z > 0$ to make a point appear in the normal plane. But an image captures only limited area of M, hence all normal points with large x and y, do not appear in it. Also, this model does not correctly approximate wide-angle cameras because such cameras always have a certain non-linear distortion, which is not taken into account by the pinhole model and is discussed later in this chapter.

2.1.1 Inverse Model

Since the projection model is a *many-to-one* mapping, there is no real inverse mapping. But as it has been mentioned earlier, there is a diffeomorphism between directions coming out of the viewpoint and points in image. So what we need as an inverse mapping is a $\mathbb{R}^2 \to \mathbb{R}^3$ function, which transforms 2D image points into 3D vectors which correspond to the point direction. Representing directions by 3D vectors is redundant, but it appears to be practical, since it is easier to compute and to manipulate afterwards.

In the case of the pinhole model, a normal point is expressed as:

$$\boldsymbol{m} = \begin{pmatrix} \dfrac{u - u_0}{f_u} \\ \dfrac{v - v_0}{f_v} \\ 1 \end{pmatrix} \tag{2.5}$$

2.1.2 Jacobian Matrix

Usually, computer vision applications imply the use of non-linear solvers to perform reconstruction, calibration, or mapping. And solvers which use the Levenberg-Marquardt algorithm require Jacobian matrices of projection models. There are two types of Jacobian matrices required.

Projection Jacobian Matrix This matrix is defined as:

$$J_{\boldsymbol{X}} = \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{X}} \tag{2.6}$$

In the case of the pinhole model its expression is:

$$J_{\boldsymbol{X}} = \begin{pmatrix} \dfrac{f_u}{z} & 0 & -\dfrac{f_u x}{z^2} \\ 0 & \dfrac{f_v}{z} & -\dfrac{f_v y}{z^2} \end{pmatrix} \tag{2.7}$$

We can express x and y using the image coordinates:

$$J_{\boldsymbol{X}} = \frac{1}{z} \begin{pmatrix} f_u & 0 & -u \\ 0 & f_v & -v \end{pmatrix} \tag{2.8}$$

It means that we can compute the projection Jacobian matrix up to an unknown scale factor, which depends on z. In some applications, like visual servoing, the system may converge even if the scale factor is unknown.

Parameter Jacobian It is defined as:

$$J_{\boldsymbol{\alpha}} = \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{\alpha}} \tag{2.9}$$

Taking $\boldsymbol{\alpha} = (f_u, f_v, u_0, v_0)^T$ results in:

$$J_{\boldsymbol{\alpha}} = \begin{pmatrix} \dfrac{x}{z} & 0 & 1 & 0 \\ 0 & \dfrac{y}{z} & 0 & 1 \end{pmatrix} \tag{2.10}$$

Again, as in the case of $J_{\boldsymbol{X}}$, we can use image coordinates:

$$J_{\boldsymbol{\alpha}} = \begin{pmatrix} \dfrac{u - u_0}{f_u} & 0 & 1 & 0 \\ 0 & \dfrac{v - v_0}{f_v} & 0 & 1 \end{pmatrix} \tag{2.11}$$

Depending on the application one form or the other can be more suitable.

2.1.3 Straight Line Projection

This model projects 3D straight lines to straight lines in the image. First we can look at point projection onto the normal plane as the intersection between the plane and the straight line passing through projection origin O and 3D point $\boldsymbol{X}$. If we have a 3D line to be projected, we have to repeat this process for all the points which belong to the line. In fact, it is equivalent to finding the intersection of normal plane M and the plane which passes through projection origin O and the 3D line. It is illustrated in Fig. 2.7: the line defined by $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ is indeed the intersection of M and plane $O\boldsymbol{X}_1\boldsymbol{X}_2$. Since the intersection is a straight line, the projection in the image will be a straight line as well, since affine transformations project straight lines to straight lines.

A more general concept of finding projections of straight lines by computing the intersection of a plane and a surface (not necessarily planar) will be developed later in this chapter.

2.2 Cameras with Distortion

Unfortunately, only narrow-angle cameras are well approximated by the pure pinhole model described above. On the other hand, wide-angle cameras are better adapted for environment perception as they cover a larger field of view and make it easier to perceive nearby objects, in contrast to narrow-angle cameras which are meant to observe remote objects. Different techniques have been developed to fit these lenses which distort the geometry of the image. The most intuitive definition of geometric distortion in images is that 3D straight lines don't appear straight (for example Fig. 2.4). The following main types of image distortion can be defined:

1. Radial distortion — acts only along rays coming from the image center. It is covariant with rotation about the optical axis. That is, rotation of camera about z leads to rotation of the image about image center, whatever the radial distortion. Covariant means that no matter whether you first rotate and then apply distortion, or the other way around, the result is the same.

2. Tangential distortion — appears because the retina is not perfectly perpendicular to the optical axis.

3. Irregular distortion — caused by low optic quality, these days its effect is negligible for dioptric systems.

Tangential distortion is included in one of the models reviewed hereafter. The other models do not take it into account, including the one developed within this work. As shown by the calibration results given in the corresponding chapter, high-distortion fisheye cameras can still be modeled and calibrated with sub-pixel precision, even with radial distortion only.

2.2.1 Distortion for Pinhole Cameras

The distortion model is usually added in the following manner. First we project all the points onto the projection plane:

$$\boldsymbol{m} = (x_m \; y_m \; 1)^T = \frac{\boldsymbol{X}}{z} \tag{2.12}$$

Then we apply radial distortion:

$$\boldsymbol{m}_d = \boldsymbol{D}(\boldsymbol{m}) = \begin{pmatrix} D(r)x_m \\ D(r)y_m \\ 1 \end{pmatrix} \tag{2.13}$$

Here $r = \sqrt{x_m^2 + y_m^2}$; $\boldsymbol{D} : \mathbb{P}^2 \to \mathbb{P}^2$ is a radial distortion mapping, where $\mathbb{P}^2$ is a 2-dimensional projective space on $\mathbb{R}$. In other words, $\boldsymbol{D}$ maps points of the normal plane to points of the normal plane. It is defined by $D : \mathbb{R}_{+0} \to \mathbb{R}_+$, that represents the deflection of a ray from its pinhole trajectory; $D(0) = 1$. The last step is to apply the projection matrix K.

Theoretically we can model any projection mapping for $\boldsymbol{X}$ with $z > 0$ using these relations. But there are two problems. The first one is how to choose the distortion function D. In the computer vision library *OpenCV* Bradski (2000) the following form is used:

$$D(r) = \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} \tag{2.14}$$

Obviously these models approximate the distortion just in a limited range of r.

This fact is also related to the second problem, which arises when $z \to 0$. In this case the model is still defined but the farther from the optical axis, the less precise the result, because of the distortion model limitations and numerical problems. And finally, there are cameras which have more than 180° field of view. For them, applying a single pinhole model with distortion is simply impossible. In some works (for example Caruso *et al.* (2015)), multiple pinhole models are used to model a single fisheye camera. The fisheye image is first projected onto a cube, each side of which represents a different pinhole camera.

2.2.2 Capture-Ray-Based Model

The authors of Kannala and Brandt (2006) propose to compute the distance from the projection center to the projected point using a function of the angle between the optical axis and the ray direction. There are multiple possible models in this context. Let ϑ be the angle between the optical axis and the direction to a spatial point X. Let f be the focal length and r the distance from the image center to the projected point $\boldsymbol{p}$ (see Fig. 2.8-b). Then the following projection models are possible (their plots are represented in Fig. 2.8-a):

(i) Perspective Projection:

$$r = f \tan(\vartheta) \tag{2.15}$$

(ii) Stereographic projection:

$$r = 2f \tan\left(\frac{\vartheta}{2}\right) \tag{2.16}$$

(iii) Equidistance projection:

$$r = f\vartheta \tag{2.17}$$

(iv) Equisolid angle projection:

$$r = 2f \sin\left(\frac{\vartheta}{2}\right) \tag{2.18}$$

(v) Ortogonal projection:

$$r = f \sin(\vartheta) \tag{2.19}$$

Also, in the same paper the authors suggest that using the basic pinhole perspective projection model for fisheye cameras is not a reasonable choice because as ϑ approaches $\pi/2$, projection coordinates go to infinity.

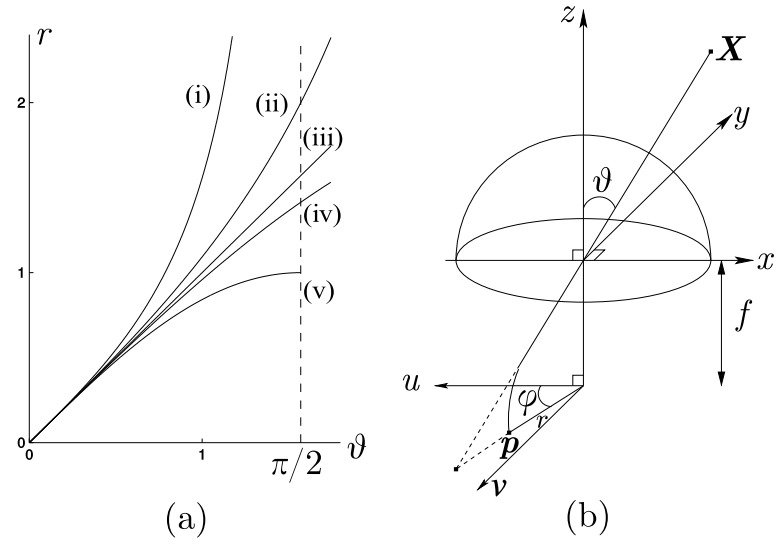


Figure 2.8: The ray-capture-based models. (a) plots of different mappings, presented in the main text. (b) an illustration of the projection modeling process. A ray, captured at angle ϑ, mapped to the image plane using polar coordinates φ, r. Kannala and Brandt (2006)

2.2.3 Unified Camera Model

The so-called unified model is of particular interest to us because our own model is based on it. Geyer and Daniilidis (2000) introduce the unified model and show that it describes all central catadioptric systems (as mentioned above, it means lens-mirror systems). Equivalence between the unified model and the captured-ray-based model as well as the pinhole model has been shown in Courbon *et al.* (2007).

This model aims to correctly represent the projection of points with zero and even negative z. It is needed to model fisheye lenses with more than 180° angle of view. This is done by changing the normalization equation to the following:

$$\boldsymbol{m} = \begin{pmatrix} x_m \\ y_m \\ 1 \end{pmatrix} = \begin{pmatrix} \dfrac{x}{z + \xi\rho} \\ \dfrac{y}{z + \xi\rho} \\ 1 \end{pmatrix} \tag{2.20}$$

where $\rho = \sqrt{x^2 + y^2 + z^2}$; ξ is a projection parameter. When $\xi = 0$ we are back to the pinhole model. But the larger we take ξ, the wider the allowed angle between the optical axis and the point to be projected. The illustration of the projection process is given in Fig. 2.9.

So we expect narrow-angle low-distortion cameras to have this parameter about 0, while for fisheye cameras it should be somewhere around one. Notice that (2.20) generally does not map straight lines to straight lines. So it introduces some distortion, which is, in fact, similar to that of a real fisheye camera.

Distortion Mapping Unfortunately, this distortion is not flexible enough to model real fisheye cameras. In Mei and Rives (2007) the model has been augmented with an additional distortion mapping and a calibration technique has been proposed. The distortion is modeled

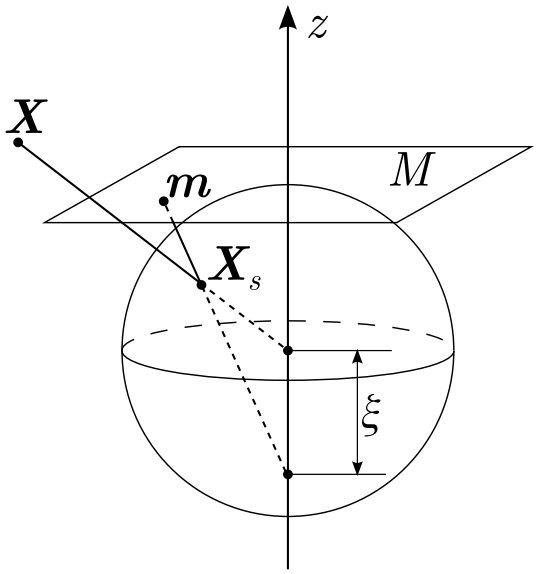


Figure 2.9: Illustration of the Unified Camera Model. A spatial point $\boldsymbol{X}$ is projected onto a unit sphere. Then it is projected onto normal plane M from a projection center, shifted by ξ.

as follows:

$$\boldsymbol{x}_d = \boldsymbol{D}(\boldsymbol{m}) + \boldsymbol{\delta}(\boldsymbol{m})$$

$$\boldsymbol{\delta}(\boldsymbol{m}) = \begin{pmatrix} 2k_4 x_m y_m + k_5(r^2 + 2x_m^2) \\ 2k_5 x_m y_m + k_4(r^2 + 2y_m^2) \\ 0 \end{pmatrix} \qquad \boldsymbol{D}(\boldsymbol{m}) = \begin{pmatrix} D(r)x_m \\ D(r)y_m \\ 1 \end{pmatrix} \tag{2.21}$$

$$D(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6$$

Where $r = \sqrt{x_m^2 + y_m^2}$; $\boldsymbol{\delta}$ models tangential distortion due to a misalignment of the retina normal and the lens optical axis in the camera; $\boldsymbol{D}$ represents the radial distortion.

Finally we apply matrix K:

$$\boldsymbol{p} = K\boldsymbol{X}_d \tag{2.22}$$

Overall we have 10 projection parameters: ξ, $k_{1..5}$, f_u, f_v, u_0, v_0.

Distortion Model Issues Even though the model provides high precision, it has some drawbacks. We can rewrite (2.20) as $\boldsymbol{m} = \phi(r)\boldsymbol{X}_n$, where $\boldsymbol{X}_n = \boldsymbol{X}/z$, and $r = \sqrt{x_n^2 + y_n^2}$. Here ϕ represents the nonlinear part of the projection:

$$\phi(r) = \frac{1}{1 + \xi\sqrt{1 + r^2}} \tag{2.23}$$

We see that the function $\phi(r)$ is even, so its Taylor expansion contains just even degrees of r. Hence:

$$\phi(r) = \phi(0) + \frac{\phi''(0)}{2} r^2 + o(r^2) \tag{2.24}$$

So, the argument here is that in the neighborhood of the projection center the distortion caused by the nonlinear projection model is well-approximated by a second order polynomial with $\phi^* = \phi(0) + \frac{\phi''(0)}{2} r^2$. If we then apply another distortion model that contains a second-order term, we still get a distortion that is well-approximated by a second-order polynomial. Hence, the second order term in the distortion model (2.21) is redundant and it does not improve the model precision.

2.3 Enhanced Unified Camera Model

Important properties of a distortion model are simplicity and flexibility. That is, the model should have few parameters but still allow us to model different types of cameras. Also it is important for a model to have a simple mathematical form, which allows to study the properties of the model analytically.

Tangential distortion is not considered in this work. Though it can be added easily, we argue that without it the model is still accurate, which is demonstrated by the calibration of multiple real fisheye lenses. Maybe, the question of tangential distortion is more important in the case of catadioptric systems, but these systems are not treated in this work, and this model has not yet been tested for such systems.

Here are the proposed projection relations:

$$\boldsymbol{m} = \begin{pmatrix} \dfrac{x}{\alpha\rho + (1-\alpha)z} \\ \dfrac{y}{\alpha\rho + (1-\alpha)z} \\ 1 \end{pmatrix} \qquad \rho = \sqrt{\beta(x^2+y^2)+z^2} \tag{2.25}$$
$$\boldsymbol{p} = K\boldsymbol{m}$$

The two parameters are $\alpha \in [0,1]$ and $\beta > 0$. They allow us to better approximate the properties of lenses with high distortion. This model assumes that the denominator $\alpha\rho + (1-\alpha)z > 0$. K is the same matrix as in (2.1). Hereafter, $\gamma = 1 - \alpha$.

2.3.1 Jacobian Matrices

We provide here the Jacobian matrices of the projection function. They are needed for calibration and for common reconstruction algorithms (such as Bundle Adjustment, see Chapter 3). Let us write the projection relation in the following way:

$$\begin{aligned} \boldsymbol{m} &= \begin{pmatrix} \dfrac{x}{\eta} & \dfrac{y}{\eta} \end{pmatrix}^T \\ \eta &= \gamma z + \alpha\rho \\ \rho &= \sqrt{\beta(x^2+y^2)+z^2} \end{aligned} \tag{2.26}$$

Projection Jacobian First let us compute the partial derivatives of ρ:

$$\frac{\partial\rho}{\partial x} = \frac{\beta x}{\rho} \qquad \frac{\partial\rho}{\partial y} = \frac{\beta y}{\rho} \qquad \frac{\partial\rho}{\partial z} = \frac{z}{\rho} \tag{2.27}$$

Then by applying the chain rule and noticing that $\dfrac{\partial\eta}{\partial\rho} = \alpha$ and $\dfrac{\partial\eta}{\partial z} = \gamma$ we can compute the Jacobian matrix:

$$\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{X}} = \begin{pmatrix} \dfrac{1}{\eta} - \dfrac{\alpha\beta x^2}{\eta^2\rho} & -\dfrac{\alpha\beta xy}{\eta^2\rho} & -\dfrac{x}{\eta^2}\left(\gamma + \dfrac{\alpha z}{\rho}\right) \\ -\dfrac{\alpha\beta xy}{\eta^2\rho} & \dfrac{1}{\eta} - \dfrac{\alpha\beta y^2}{\eta^2\rho} & -\dfrac{y}{\eta^2}\left(\gamma + \dfrac{\alpha z}{\rho}\right) \end{pmatrix} \tag{2.28}$$
$$\frac{\partial\boldsymbol{p}}{\partial\boldsymbol{X}} = \begin{pmatrix} f_u & 0 \\ 0 & f_v \end{pmatrix} \frac{\partial\boldsymbol{m}}{\partial\boldsymbol{X}}$$

Parameter Jacobian Matrix The variable vector with respect to which we differentiate the model is:

$$\boldsymbol{\alpha} = (\alpha, \beta, f_u, f_v, u_0, v_0)^T \tag{2.29}$$

The Jacobian matrix part, corresponding to the projection matrix parameters (the last four) is almost the same as for the pinhole camera. To get the derivatives with respect to α and β, let us first differentiate η:

$$\frac{\partial \eta}{\partial \alpha} = \rho - z \qquad \frac{\partial \eta}{\partial \beta} = \frac{\alpha(x^2 + y^2)}{\rho} \tag{2.30}$$

The last thing we need is:

$$\frac{\partial u}{\partial \eta} = -\frac{f_u x}{\eta^2} \qquad \frac{\partial v}{\partial \eta} = -\frac{f_v y}{\eta^2} \tag{2.31}$$

Now using the chain rule we can compute the final expression:

$$\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\alpha}} = \begin{pmatrix} -\dfrac{f_u x(\rho - z)}{\eta^2} & -\dfrac{f_u x \alpha(x^2 + y^2)}{\eta^2 \rho} & \dfrac{x}{\eta} & 0 & 1 & 0 \\ -\dfrac{f_v y(\rho - z)}{\eta^2} & -\dfrac{f_v y \alpha(x^2 + y^2)}{\eta^2 \rho} & 0 & \dfrac{y}{\eta} & 0 & 1 \end{pmatrix} \tag{2.32}$$

2.3.2 Projection Surfaces

To analyze the model, let us introduce the notion of *projection surface*. In Scaramuzza and Martinelli (2006) the projection is modeled using an intermediate surface. The surface is defined using a polynomial function of image points. The notion is somewhat similar to the one we introduce here, but the other way around: the projection surface is defined using an equation in 3D space and allows to compute projections of spatial points.

This notion can be applied to a wide variety of projection models with different kinds of distortion. Let $\eta : \mathbb{R}^3 \to \mathbb{R}_+$ be a homogeneous function of degree 1:

$$\forall \lambda \in \mathbb{R}_+ \quad \eta(\lambda \boldsymbol{X}) = \lambda \eta(\boldsymbol{X}) \tag{2.33}$$

Let the projection relation be defined as follows:

$$\boldsymbol{m} = \begin{pmatrix} \dfrac{x}{\eta(\boldsymbol{X})} \\ \dfrac{y}{\eta(\boldsymbol{X})} \\ 1 \end{pmatrix} \tag{2.34}$$

Then the projection surface P is defined by the following equation:

$$\eta(\boldsymbol{X}) = 1 \tag{2.35}$$

Every function η defines a different projection model with a different projection surface. How complex such models can get is yet to be studied.

In this work, we consider projection relations with distortion with radial symmetry only (or simply, radial distortion). That is:

$$\begin{cases} x_1^2 + y_1^2 = x_2^2 + y_2^2 \\ z_1 = z_2 \end{cases} \implies \eta(\boldsymbol{X}_1) = \eta(\boldsymbol{X}_2) \qquad \forall \boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathbb{R}^3 \tag{2.36}$$

This means that η can be represented by a function:

$$\hat{\eta} : \mathbb{R}^2 \to \mathbb{R}_+ \tag{2.37}$$

such that:

$$\eta(\boldsymbol{X}) = \hat{\eta}\left(\sqrt{x^2 + y^2}, z\right) = \hat{\eta}(r, z) \tag{2.38}$$

$$\hat{\eta}(r, z) = \hat{\eta}(-r, z), \quad \forall r \in \mathbb{R} \tag{2.39}$$

$\hat{\eta}$ is a homogeneous function of degree 1, since both r and z are homogeneous functions of $\boldsymbol{X}$ of degree 1. (2.39) allows us to operate on $\mathbb{R}^2$, rather than $\mathbb{R}_{+0} \times \mathbb{R}$. In this case $y = 0 \implies \eta(\boldsymbol{X}) = \hat{\eta}(x, z)$, or $\hat{\eta}$ represents η in xz plane.

The projection surface is a surface of revolution. It is generated by rotating the curve $\hat{\eta}(x, z) = 1$ about z-axis. Let us call this curve a *projection curve*. The geometric meaning of (C.4) is that all the points of the projection surface are projected orthogonally to the image plane. So we can think of the projection process as scaling the point $\boldsymbol{X}$ by $\eta(\boldsymbol{X})$ and then projecting it orthogonally onto $\boldsymbol{m} \in M$ — the normal plane (see Fig. 2.10):

$$\begin{aligned} \boldsymbol{X}_p &= \frac{\boldsymbol{X}}{\eta(\boldsymbol{X})} \\ \boldsymbol{m} &= (x_p \; y_p \; 1) \end{aligned} \tag{2.40}$$

Using (C.2) we can deduce:

$$\eta(\boldsymbol{X}_p) = \eta\left(\frac{\boldsymbol{X}}{\eta(\boldsymbol{X})}\right) = \frac{\eta(\boldsymbol{X})}{\eta(\boldsymbol{X})} = 1 \tag{2.41}$$

Hence, all points $\boldsymbol{X}_p$ belong to the projection surface. In fact, $\boldsymbol{X}_p$ is the intersection between the projection surface P and ray $O\boldsymbol{X}$ (Fig. 2.10).

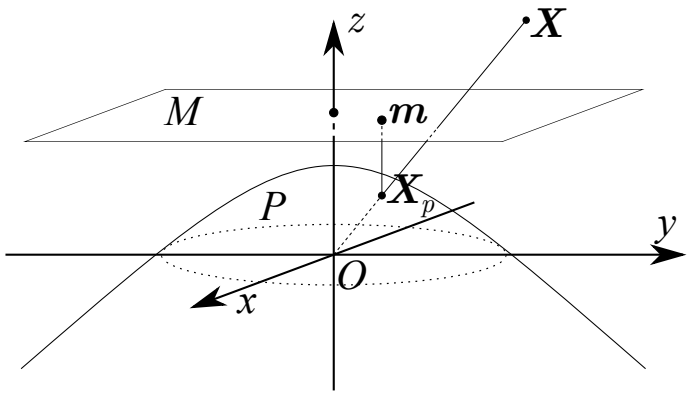


Figure 2.10: Illustration of the notion of projection surface. z is the optical axis; O is the center of projection; $\boldsymbol{X}_p$ is obtained by projecting $\boldsymbol{X}$ to P along $O\boldsymbol{X}$ ray. Then this point is transformed into $\boldsymbol{m}$ by projecting it orthogonally onto an intermediate projection plane M which is defined as $z = 1$

One convenience of the notion is that it is relatively easy to see for which spatial points the projection is defined. For example, (2.12) corresponds to $\eta(\boldsymbol{X}) = z$. So the projection surface in this case is defined by $z = 1$. It is a plane, and all the points with $z \leq 0$ do not define rays that intersect the surface.

Let us apply the notion to the proposed model. In this case:

$$\eta(\boldsymbol{X}) = \alpha\sqrt{\beta(x^2 + y^2) + z^2} + (1 - \alpha)z \tag{2.42}$$

So $\eta(\boldsymbol{X}) = 1$ leads to:

$$\alpha\sqrt{\beta(x^2 + y^2) + z^2} + (1 - \alpha)z = 1 \tag{2.43}$$

Let us replace $1-\alpha$ by γ and x^2+y^2 by r^2:

$$\alpha\sqrt{\beta r^2+z^2}=1-\gamma z \tag{2.44}$$

By squaring both sides we get:

$$\alpha^2\beta r^2+\alpha^2 z^2=1-2\gamma z+\gamma^2 z^2 \tag{2.45}$$

We should remember that, as we have squared both sides, we may get some solutions that do not satisfy (2.44). Let us also note that $\gamma^2-\alpha^2=\gamma-\alpha$, hence:

$$\alpha^2\beta r^2=1-2\gamma z+(\gamma-\alpha)z^2 \tag{2.46}$$

$$\alpha^2\beta r^2+(\alpha-\gamma)z^2+2\gamma z=1 \tag{2.47}$$

This equation defines a second-order projection curve. $\alpha=0.5$ leads to $\alpha=\gamma$ and

$$z=1-0.25\beta r^2 \tag{2.48}$$

That is, the projection curve is a parabola. If $\alpha<0.5$, (C.6) defines a hyperbola and if $\alpha>0.5$, it is an ellipse (because the coefficient in front of z^2 is negative and positive respectively). We can see that $r=0,\ z=1$ always satisfies (2.44). So the projection surface is a surface of revolution, which is defined by a conic projection curve that passes through $(0\ 0\ 1)^T$.

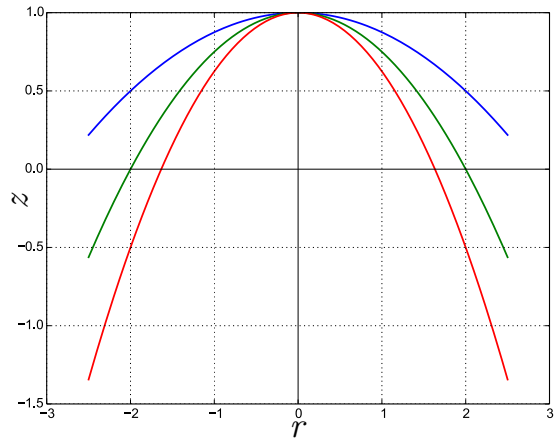


Figure 2.11: The projection curve is a parabola when $\alpha=0.5$. We can see different parabolas for $\beta=0.5$ (blue), 1 (green), and 1.5 (red) (wider parabola corresponds to smaller β).

That is the difference between the proposed model and (2.20). The latter allows to get just one parabola as a projection curve when $\xi=1$, while the former allows to scale this parabola along the x-axis (Fig. 2.11). In the case when $\xi\neq 1$ the same is true, but it is just less obvious. β allows us to adjust the projection surface, while α defines its shape.

2.3.3 Completeness of the Model

In fact, we can show that (C.6) describes all the possible conics that pass through $(0\ 1)^T$ and are symmetric with respect to z-axis (here we consider rz coordinate plane). To sketch out a proof, let us consider a general conic equation:

$$Ar^2+Brz+Cz^2+Dr+Ez=1 \tag{2.49}$$

To make it symmetric with respect to z-axis we have to have $B = 0$ and $D = 0$. Then, by substituting $z = 1$, $r = 0$ we get:

$$C + E = 1 \tag{2.50}$$

If we check (C.6), we see that all these conditions are satisfied. Indeed:

$$\begin{aligned} &B = 0 \\ &D = 0 \\ &C + E = \alpha - \gamma + 2\gamma = \alpha + \gamma = 1 \end{aligned} \tag{2.51}$$

In total we have 5 parameters, and 3 constraints, which lives us with two degrees of freedom which, in turn, are represented by projection parameters α and β.

But what if there is a projection curve which does not pass through $(0\ 1)^T$? Actually we can scale it so that it does (using parameters f_u and f_v — see Fig. 2.12). So we can say that the proposed model is complete in the sense that it can fit any projection whose projection curve is a conic section.

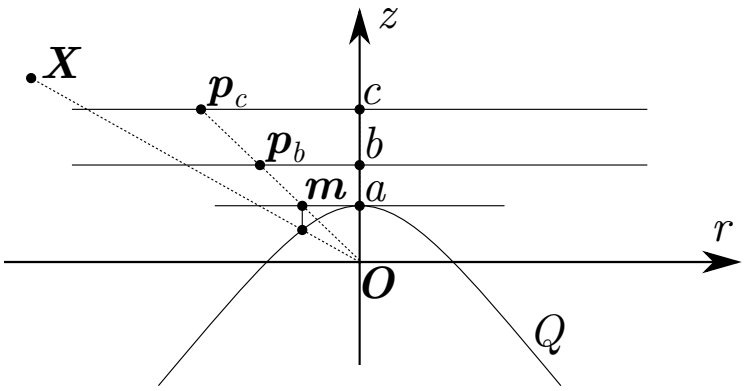


Figure 2.12: Using projection curve Q, point $\boldsymbol{X}$ is projected to $\boldsymbol{m}$, the z-coordinate of which is a. If $a = 1$ and the focal length $f = c$ then the final projection is $\boldsymbol{p}_c = f\boldsymbol{m}$. But if $a < 1$, let us say that $b = 1$, then $\boldsymbol{p}_b = \frac{\boldsymbol{m}}{a}$ and the final projection $\boldsymbol{p}_c = f\boldsymbol{p}_b = \frac{f}{a}\boldsymbol{m} = f'\boldsymbol{m}$. So we see that the projection defined by Q and f' is equivalent to one defined by f and Q scaled so that it pass through 1. The same argument is true when $a > 1$.

2.3.4 Inverse Model

Let $\mathfrak{f} : \mathbb{R}^3 \backslash 0 \to \mathbb{R}^2$ be the projection model defined in (C.1). Let $\mathfrak{f}^{-1} : \mathbb{R}^2 \to \mathbb{R}^3$ be the right inverse of $\mathfrak{f}$:

$$\mathfrak{f}(\mathfrak{f}^{-1}(\boldsymbol{m})) = \boldsymbol{m} \tag{2.52}$$

or, in other words, $\mathfrak{f} \circ \mathfrak{f}^{-1} = I$. Again, what we are looking for is a diffeomorphism between the image points and directions within the camera's field of view. We know that the points of the projection surface are projected orthogonally. So $\mathfrak{f}^{-1}$ may be defined as:

$$\mathfrak{f}^{-1} : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ z(x,y) \end{pmatrix} \tag{2.53}$$

Where $z(x, y)$ is an explicit solution of (2.44) with $r = \sqrt{x^2 + y^2}$. To do that we can solve (C.6) and then choose the proper solution. It is a quadratic equation:

$$\begin{aligned} &Az^2 + Bz + C = 0 \\ &A = \alpha - \gamma \\ &B = 2\gamma \\ &C = \alpha^2\beta r^2 - 1 \end{aligned} \tag{2.54}$$

This is solved as follows:

$$\begin{aligned} &D = B^2 - 4AC = \\ &= 4\left(\gamma^2 - (\alpha-\gamma)(\alpha^2\beta r^2 - 1)\right) \\ &z = \frac{-B \pm \sqrt{D}}{2A} = \\ &= \frac{-\gamma \pm \sqrt{D/4}}{\alpha - \gamma} \end{aligned} \tag{2.55}$$

We can choose the solution by using the fact that $z(0,0) = 1$. Notice that $r = 0 \implies D/4 = \alpha^2$. Hence, the solution must be defined by:

$$z = \frac{\sqrt{\gamma^2 - (\alpha-\gamma)(\alpha^2\beta r^2 - 1)} - \gamma}{\alpha - \gamma} \tag{2.56}$$

We see that (2.56) is not defined when $\alpha = 0.5$. It is so because (C.6) is no longer a quadratic equation.

We can avoid the singularity by multiplying both numerator and denominator of (2.56) by:

$$\sqrt{\gamma^2 - (\alpha-\gamma)(\alpha^2\beta r^2 - 1)} + \gamma \tag{2.57}$$

The result is:

$$\begin{aligned} z &= \frac{\gamma^2 - (\alpha-\gamma)(\alpha^2\beta r^2 - 1) - \gamma^2}{(\alpha-\gamma)\left(\sqrt{\gamma^2 - (\alpha-\gamma)(\alpha^2\beta r^2 - 1)} + \gamma\right)} = \\ &= \frac{1 - \alpha^2\beta r^2}{\sqrt{\gamma^2 - (\alpha-\gamma)(\alpha^2\beta r^2 - 1)} + \gamma} \end{aligned} \tag{2.58}$$

Also let us consider the expression under the square root in the denominator:

$$\begin{aligned} &\gamma^2 - (\alpha-\gamma)(\alpha^2\beta r^2 - 1) = \\ &= (\alpha-1)^2 + 2\alpha - 1 - (\alpha-\gamma)\alpha^2\beta r^2 = \\ &= \alpha^2 - 2\alpha + 1 + 2\alpha - 1 - (\alpha-\gamma)\alpha^2\beta r^2 = \\ &= \alpha^2(1 - (\alpha-\gamma)\beta r^2) \end{aligned} \tag{2.59}$$

hence, we can rewrite the solution as:

$$z = \frac{1 - \alpha^2\beta r^2}{\alpha\sqrt{1 - (\alpha-\gamma)\beta r^2} + \gamma} \tag{2.60}$$

We can get rid of γ :

$$\alpha - \gamma = 2\alpha - 1 \tag{2.61}$$

Because of the square root, z is defined as a real value when:

$$1 - (2\alpha - 1)\beta r^2 \geq 0 \tag{2.62}$$

If $\alpha \leq 0.5$ then $2\alpha - 1 \leq 0$ and (2.62) is always true. Therefore, there is no limit on r. On the other hand if $\alpha > 0.5$ then z is defined for:

$$r^2 \leq \frac{1}{(2\alpha - 1)\beta} \tag{2.63}$$

It is so because the projection curve for $\alpha > 0.5$ is an ellipse, so the projection relation is not surjective (Fig. 2.13). It should be noticed, that it is not a singularity, but it defines limits on values of r during the reconstruction process.

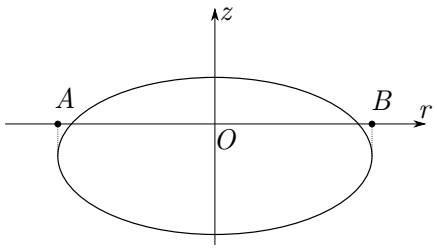


Figure 2.13: In case of an ellipse as a projection curve all the spatial points are projected in between A and B, hence, the inverse is not defined beyond $[A, B]$.

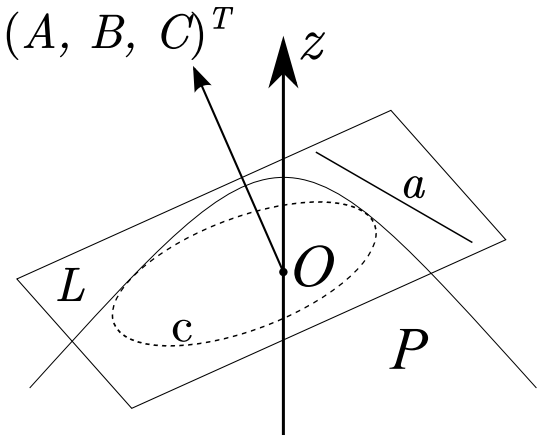


Figure 2.14: Straight line a together with projection origin O define plane L. Intersection of L and projection surface P defines a curve c. By projecting c orthogonally onto the normal plane we get the projection of a.

2.3.5 Straight Line Projection

Using the notion of projection surface, we can show that straight lines are projected as conic sections. Given straight line a, let us define plane L which passes through a and O (Fig. C.1). It is defined by the following equation:

$$Ax + By + Cz = 0 \tag{2.64}$$

To find the line projection, first we need to project it onto the projection surface. To do that, we need to find intersection c between surface P and plane L. Combining (C.6) with (C.10) we get the following system of equations:

$$\begin{cases} Ax + By + Cz = 0 \\ \alpha^2\beta(x^2 + y^2) + (\alpha - \gamma)z^2 + 2\gamma z = 1 \end{cases} \tag{2.65}$$

The next step in the projection process is to project the curve orthogonally onto the normal plane. It means that we need to exclude z from the equations. If $C = 0$ then the projection is a straight line, defined by

$$Ax + By = 0 \tag{2.66}$$

This line passes through the image center. There is an intuitive explanation to this fact: the system has only radial distortions and the only straight lines that are projected into straight lines are the ones which pass through the optical axis. If $C \neq 0$ then we can find z from the first equation:

$$z = -\frac{Ax + By}{C} \tag{2.67}$$

and substitute it into the second one:

$$\alpha^2\beta(x^2+y^2)+(\alpha-\gamma)\left(\frac{Ax+By}{C}\right)^2-2\gamma\frac{Ax+By}{C}=1 \tag{2.68}$$

As can be seen, we have a second degree polynomial of x and y. This polynomial defines a conic section.

Notice that using this method, we can compute an equation of a straight line projection for any projection model defined via a projection surface.

3 Two-Views Geometry

In this section we consider the case when there are two cameras observing the same scene, or the same camera has been used twice to take images of the same scene. One important underlying assumption is that the scene geometry remains the same for both observations. In this case we can assume that all the changes in images are caused only by the viewpoint change, and perhaps by the difference in camera intrinsic parameters.

3.1 Pinhole Model

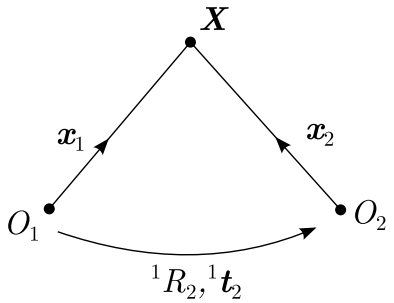


Figure 2.15: Epipolar geometry: given two cameras with projection origins O_1 and O_2, as well the transformation between the camera frames, represented by rotation matrix ${}^1R_2$ and translation vector ${}^1\boldsymbol{t}_2$, for any 3D point $\boldsymbol{X}$ we can formulate the epipolar constraint. If $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are direction vectors towards $\boldsymbol{X}$ from the camera origins, then vectors $\boldsymbol{t}$, $\boldsymbol{x}_1$, and $\boldsymbol{x}_2$ are coplanar, because they lie in plane $\boldsymbol{X}O_1O_2$.

Essential Matrix As long as two feature points are the projections of the same 3D point, we can figure out some relations. Fig. 2.15 illustrates the epipolar constraint. Let $\boldsymbol{X}$ be a 3D point, while $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be the direction vectors towards $\boldsymbol{X}$ from the first and the second viewpoints. Note that $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and the stereo base $\boldsymbol{t}$ are coplanar. It means that their mixed product is zero. If we project them to the same frame then we can write it down as follows:

$${}^1\boldsymbol{x}_1\cdot({}^1\boldsymbol{t}\times{}^1\boldsymbol{x}_2)=0 \tag{2.69}$$

We can rewrite this expression using the matrix form:

$${}^1\boldsymbol{x}_1^T[{}^1\boldsymbol{t}]_\times{}^1R_2{}^2\boldsymbol{x}_2=0 \tag{2.70}$$

where $[\boldsymbol{t}]_\times$ is the skew-symmetric matrix. The matrix in the middle $[{}^1\boldsymbol{t}]_\times{}^1R_2=E$ is called *essential matrix*, It depends only on the cameras relative position and represents the constraint:

$${}^1\boldsymbol{x}_1^TE{}^2\boldsymbol{x}_2=0 \tag{2.71}$$

where x_i^i stands for the point projected into camera i, expressed in this camera frame. Note that this expression must be true for any two projections of the same point $\boldsymbol{X}$.

Fundamental Matrix In the case of a pinhole camera, we can express the epipolar constraint directly using the image points. (2.71) is true if we use normal points $\boldsymbol{m}$ as a special case of direction vectors $\boldsymbol{x}$. Then we can recall the link between normal points and image points:

$$\boldsymbol{p} = K\boldsymbol{x} \tag{2.72}$$

Let us use homogeneous coordinates for image points:

$$\boldsymbol{p} = (u, v, 1)^T \tag{2.73}$$

Then K becomes 3×3 and invertible:

$$K = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.74}$$

The reconstructed directions can be expressed as follows:

$$\boldsymbol{m} = K^{-1}\boldsymbol{p} \tag{2.75}$$

Substituting (2.75) into (2.71) yelds:

$$\boldsymbol{p}_1^T F \boldsymbol{p}_2 = 0 \tag{2.76}$$

F is called *fundamental matrix*:

$$F = K_1^{-T} E K_2^{-1} \tag{2.77}$$

Note that this constraint is valid even if the projection matrix is not the same for the two cameras. Moreover, we can compute F for uncalibrated cameras (that is, cameras with unknown intrinsic parameters).

3.2 Two-Views Geometry for Fisheye Lenses

3.2.1 Homography Matrix

In Courbon *et al.* (2012) it has been shown that the Unified Model allows to use the homography matrix to perform 3D reconstruction. The Enhanced Unified Camera Model also alows us to do it. Suppose that we have a set of points $\boldsymbol{X}_{1..N}$ which belong to the same plane and we observe it with two calibrated cameras a and b (with projection functions defined by η_a and η_b respectively). Let us define the plane by the following equation:

$${}^a\boldsymbol{n} \cdot {}^a\boldsymbol{X} = d_a \tag{2.78}$$

Here superscript a stands for the first camera frame. We assume $d_a \neq 0$. Then we can apply the projection model:

$${}^a\boldsymbol{X}_p = \frac{{}^a\boldsymbol{X}}{\eta_a({}^a\boldsymbol{X})} \tag{2.79}$$

On the other hand we can reconstruct $\boldsymbol{X}_p$ from a point $\boldsymbol{m} = (x_m \; y_m \; 1)$ from the normalized plane M (see Fig. 2.10) using (C.9):

$$\boldsymbol{X}_p = \begin{pmatrix} x_m \\ y_m \\ z(x_m, y_m) \end{pmatrix} \tag{2.80}$$

By computing the scalar product with ${}^a\boldsymbol{n}$ for both sides of (2.79) and applying (2.78) we get:

$$\frac{1}{\eta_a({}^a\boldsymbol{X})} = \frac{{}^a\boldsymbol{n} \cdot {}^a\boldsymbol{X}_p}{d_a} \tag{2.81}$$

The transformation between frames a and b is defined as:

$$ {}^b\boldsymbol{X} = {}^bR_a\,{}^a\boldsymbol{X} + {}^b\boldsymbol{t}_a \tag{2.82} $$

By replacing $\boldsymbol{X}$ by $\boldsymbol{X}_p$ and using (2.81) we obtain:

$$ \frac{\eta_b({}^b\boldsymbol{X})}{\eta_a({}^a\boldsymbol{X})}\,{}^b\boldsymbol{X}_p = {}^bR_a\,{}^a\boldsymbol{X}_p + \frac{{}^a\boldsymbol{n}\cdot{}^a\boldsymbol{X}_p}{d_a}\,{}^b\boldsymbol{t}_a = \left({}^bR_a + \frac{{}^b\boldsymbol{t}_a({}^a\boldsymbol{n})^T}{d_a}\right){}^a\boldsymbol{X}_p \tag{2.83} $$

We can rewrite the last equation as:

$$ {}^b\boldsymbol{X}_p \propto {}^bH_a\,{}^a\boldsymbol{X}_p \tag{2.84} $$

where ${}^bH_a = {}^bR_a + \frac{{}^b\boldsymbol{t}_a({}^a\boldsymbol{n})^T}{d_a}$ is a 3×3 homography matrix. Reconstructing $\boldsymbol{X}_{p1..N}$ for both cameras a and b we can get the following system of equations:

$$ \begin{cases} \lambda_1\,{}^b\boldsymbol{X}_{p1} = {}^bH_a\,{}^a\boldsymbol{X}_{p1} \\ \vdots \\ \lambda_N\,{}^b\boldsymbol{X}_{pN} = {}^bH_a\,{}^a\boldsymbol{X}_{pN} \end{cases} \tag{2.85} $$

Here $\lambda_{1..N}$ are unknown scale factors. As matrix ${}^bH_a$ is defined up to a scale factor, we have $8+N$ unknowns and $3N$ equations. So we can estimate ${}^bH_a$ using just 4 points.

3.2.2 Epipolar Curve Equation

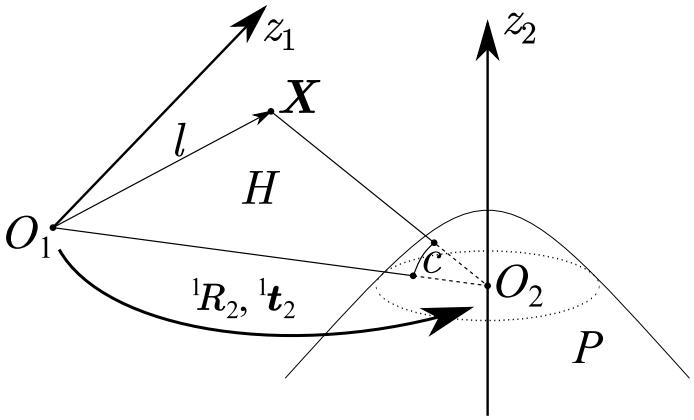


Figure 2.16: A calibrated stereo system. ${}^1R_2$ and ${}^1\boldsymbol{t}_2$ define the transformation between the camera frames (rotation matrix and translation vector); $\boldsymbol{X}$ is a reconstructed point; straight line l passes through it and the center of projection O_1 of the first camera. Plane H passes through O_2 and l. Curve c is the intersection between the plane and projection surface P. To get an epipolar curve we need to exclude the z-coordinate from the equation of c.

We can get closed-form expressions for the coefficients of epipolar curve equations for a calibrated stereo system. Consider two cameras with calibrated projection models f_1 and f_2. The transformation between them is known and represented by a rotation matrix ${}^1R_2$ and a translation vector ${}^1\boldsymbol{t}_2$ (see Fig. C.2). Hereafter the superscript defines the projection frame. For a reconstructed point ${}^2\boldsymbol{x}_2 = f_2^{-1}(\boldsymbol{p}_2)$, we can define a plane that passes through it and both centers of projection, using (2.71):

$$ \boldsymbol{A}\,{}^2\boldsymbol{x}_2 = 0 \tag{2.86} $$

Where $\boldsymbol{A} = {}^1\boldsymbol{x}_1^T E$. (2.86) is an equation of type (C.10). Hence we can find the equation of projection of the intersection of this plane with the projection surface of the second camera using (C.12) or (C.14). The final step is to replace x and y by their expressions as functions of u and v:

$$x = \frac{u - u_0}{f_u} \quad y = \frac{v - v_0}{f_v} \tag{2.87}$$

to get a polynomial of the following form:

$$k_{uu}u^2 + k_{uv}uv + k_{vv}v^2 + k_u u + k_v v + k_1 = 0 \tag{2.88}$$

Let us denote this polynomial by $h(u, v)$ or $h(\boldsymbol{p})$. Except for the epipolar lines that pass close to the projection center (in which case we can apply (C.12)), the expressions of the polynomial coefficients in this equation are cumbersome, especially k_1:

$$\begin{aligned}
k_{uu} &= \tfrac{A^2\kappa + C^2\alpha^2\beta}{C^2 f_u^2} \\
k_{uv} &= \tfrac{2AB\kappa}{C^2 f_u f_v} \\
k_{vv} &= \tfrac{B^2\kappa + C^2\alpha^2\beta}{C^2 f_v^2} \\
k_u &= 2\tfrac{-A^2 f_v u_0\kappa - ABf_u v_0\kappa - ACf_u f_v\gamma - C^2\alpha^2\beta f_v u_0}{C^2 f_u^2 f_v} \\
k_v &= 2\tfrac{-ABf_v u_0\kappa - B^2 f_u v_0\kappa - BCf_u f_v\gamma - C^2\alpha^2\beta f_u v_0}{C^2 f_u f_v^2} \\
k_1 &= \tfrac{h_{A^2}A^2 + h_{AB}AB + h_{AC}AC + h_{B^2}B^2 + h_{BC}BC + h_{C^2}C^2}{C^2 f_u^2 f_v^2} \\
h_{A^2} &= f_v^2 u_0^2\kappa \\
h_{AB} &= 2f_u f_v u_0 v_0\kappa \\
h_{AC} &= 2f_u f_v^2\gamma u_0 \\
h_{B^2} &= f_u^2 v_0^2\kappa \\
h_{BC} &= 2f_u^2 f_v\gamma v_0 \\
h_{C^2} &= \alpha^2\beta v_0^2(f_u^2 + f_v^2) - f_u^2 f_v^2 \\
\kappa &= \alpha - \gamma = 2\alpha - 1
\end{aligned} \tag{2.89}$$

Fortunately, we don't have to evaluate all of them. If we know the projection of the epipole $\boldsymbol{e}$, we can calculate k_1 using the fact that $h(\boldsymbol{e}) = 0$ (since the epipolar curves pass through it). If we denote the first five terms of $h(u, v)$, defined in (C.17), by $h'(u, v)$ then:

$$k_1 = -h'(\boldsymbol{e}) \tag{2.90}$$

4 Conclusions

Existing fisheye projection models demonstrated different combinations of advantages and disadvantages. But none of them is really universal because certain are not precise enough or suit only for particular types of lenses, while others are to computationally expensive and are not invertible. In this work we addressed this issue by proposing a novel projection model and obtained important results on its geometric properties. The description of the Enhanced Unified Camera Model has been published in Khomutenko *et al.* (2016a). We summarize the contributions in the following paragraphs.

The Enhanced Unified Camera Model By augmenting the Unified Model we get a model which, while keeping its analytical elegance and simplicity, approximates fisheye lenses

better, due to one more degree of freedom. Quantitative results of calibration are given in Chapter 3. These results show that the Enhances Unified Camera Model makes an additional distortion mapping unnecessary even for fisheye cameras with a high distortion.

Projection Surface This notion is a handy tool for analyzing geometric properties of projection models. Using this concept, a closed-form inversion of the Enhanced model has been found, its ability to model any second-order radially-symmetric projection surface has been demonstrated.

There are still a few open questions:

1. Obviously, any surface defined as $\rho(\phi, \vartheta)$ in spherical coordinates defines a projection model using this intersection-orthogonal projection paradigm. What kind of such surfaces leads to an analytic expression η from (C.3), Which of them would be interesting for projection modeling?

2. What would be the best way to change the model in order to relax the radial symmetry condition (2.36)? Will it give any significant benefit in model precision? How would it affect the inverse mapping?

Straight Line Projection Using the notion of projection surface, it has been shown that the straight lines are projected into conic sections by the Enhanced Model, and a method to compute an equation of a straight line projection has been proposed. For a calibrated stereo system, it is possible to compute the epipolar curve equations, which is used further in this work for a direct fisheye stereo correspondence algorithm.

Chapter 3

Calibration of Visual Perception Systems

1 Introduction

The majority of real-world applications rely on a certain kind of model, usually task-specific. In our case the modeled object is a physical camera, which transforms light waves into digital images. Any model contains parameters which must be identified before its application.

Generally, the identification can be viewed as a supervised machine-learning problem. In order to identify system parameters, a learning dataset must be collected and then a non-linear optimization is solved to find the model parameters which make it fit the learning data the best (Fig. 3.1).

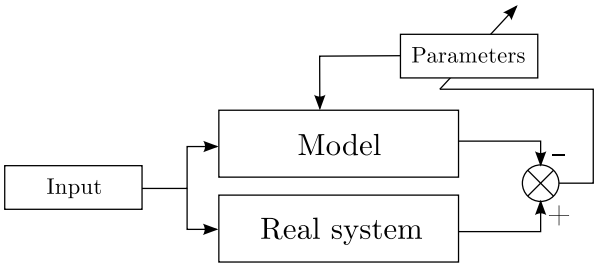


Figure 3.1: General scheme for identification problems.

The methodology described here is applied to calibration of visual systems for mobile robots, and in particular for self-driving cars. The described methods are mostly based on classical calibration approaches with some modifications and improvements. These calibration tools have been successfully applied for extrinsic calibration of a camera attached to a robotic arm, and the toolbox can face a broad variety of calibration applications, with certain adjustments according to the context.

1.1 Camera Calibration Pipeline

In the case of cameras, the calibration is usually done via a calibration object, that is an object, which is easy to detect in images and whose geometric model is precisely known. The most common type of calibration object is the *calibration board*, a flat surface with a high-contrast regular pattern (Fig. 3.2).

The general method includes the following steps:

1. Collect calibration data, that is images of the calibration board.

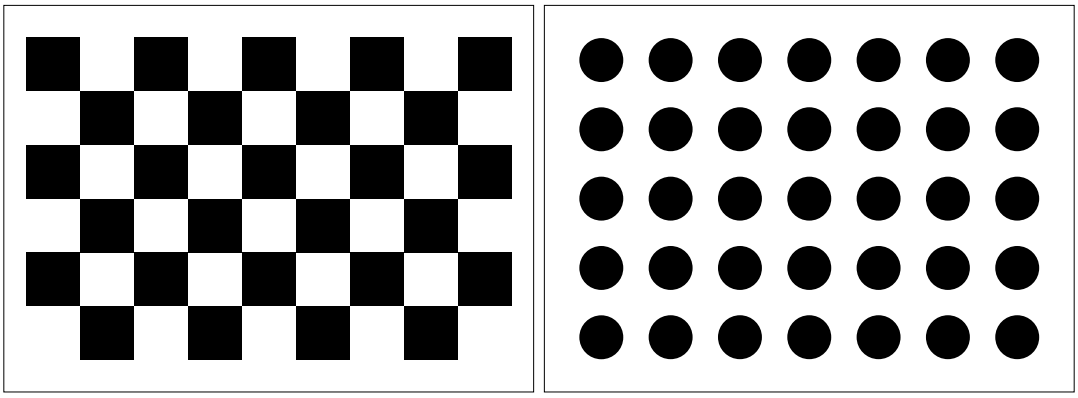


Figure 3.2: Two common calibration patterns: checkerboard and circle grid. The features are the inner corners for the former and circle barycenters for the latter.

2. Extract geometric data from images by detecting the board and expressing the positions of its feature projections in image coordinates.
3. Initialize unknown parameters.
4. Solve a global optimization problem.

These steps are discussed in details in this chapter.

Global Optimization Formulation In this work we advocate a general nonlinear-optimization-based calibration method. Even though we have to do some specific calculations to get initial values of the variables, the problem is still solved globally and all the parameters are computed simultaneously in the optimization loop.

Let us first consider monocular camera calibration.

- Let $\boldsymbol{\alpha} \in \mathbb{R}^K$ be camera intrinsic parameters.
- Let $\mathcal{X} = \{\boldsymbol{X}_1..\boldsymbol{X}_M\} \subset \mathbb{R}^3$ be the calibration board geometric model, that is 3D coordinates of its corners in its own frame (Fig. 3.3a).
- Let $\mathfrak{f} : \mathbb{R}^3 \times \mathbb{R}^K \to \mathbb{R}^2$ be a projection function. Its arguments are $\boldsymbol{X} \in \mathbb{R}^3$ and $\boldsymbol{\alpha} \in \mathbb{R}^K$.
- Let $\Xi = \{\xi_1..\xi_N\} \subset \mathrm{SE}(3)$ be the set of transformations between camera projection frame O_c and board frame O_b (Fig. 3.3b).

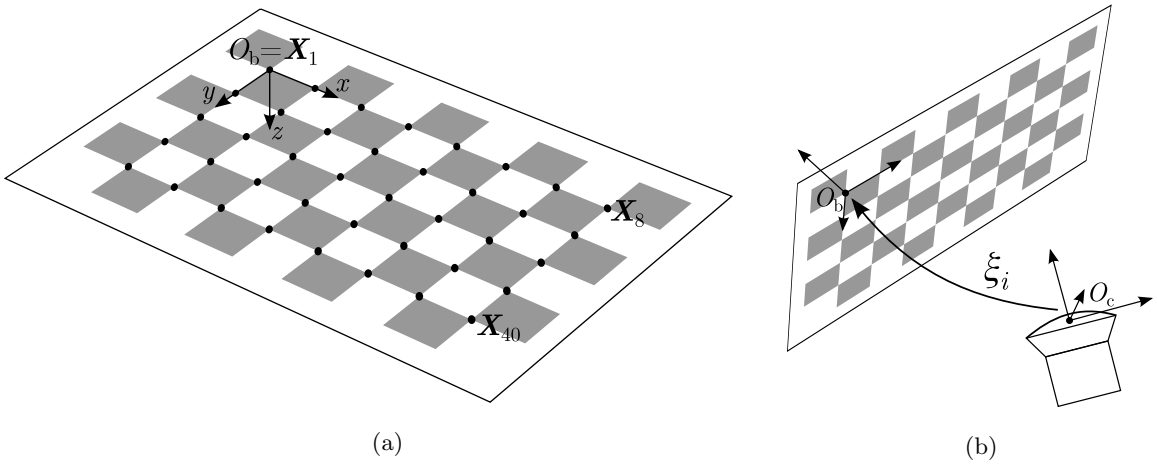
Then the model prediction is defined as follows:

$$\hat{\boldsymbol{\mathcal{P}}}(\Xi) = \{\hat{\mathcal{P}}(\xi) \mid \xi \in \Xi\} \qquad \hat{\mathcal{P}}(\xi) = \{\mathfrak{f}(\xi(\boldsymbol{X}), \boldsymbol{\alpha}) \mid \boldsymbol{X} \in \mathcal{X}\} \tag{3.1}$$

Ξ is unknown, so it must be included into the model parameters (so-called *extrinsic parameters*).

On the other hand we have the images acquired with the real camera. Let $\boldsymbol{\mathcal{P}} = \{\mathcal{P}_1..\mathcal{P}_N\}$ be the set of observations where each $\mathcal{P}_i = \{\boldsymbol{p}_1..\boldsymbol{p}_M\} \subset \mathbb{R}^2$ is the set of detected corners coordinates. It should be noted that Ξ and $\boldsymbol{\mathcal{P}}$ are homologous, that is, elements in these two sets associated with the set of calibration images. Hence they should be treated in parallel. The following convention is used hereafter: if two sets A and B are homologous then for a certain function ϕ:

$$\sum_{\substack{a \in A \\ b \in B}} \phi(a, b) = \sum_{i=1}^{\|A\|} \phi\left(A[i], B[i]\right) \tag{3.2}$$



(a) (b)

Figure 3.3: (a) Frame and geometric model of calibration board with $K = 40$ corners.(b) Acquisition of calibration board images with a camera.

It means that the summation is done once by iterating simultaneously over both sets.

It shall be noted that $\mathcal{P}$ and $\mathcal{X}$ are also homologous because elements of $\mathcal{P}$ represent detected via image processing projections of 3D points of the calibration model, defined in $\mathcal{X}$. The cost function is defined as follows:

$$E\left(\boldsymbol{\alpha}, \Xi, \mathcal{X}, \boldsymbol{\mathcal{P}}\right) = \sum_{\substack{\xi \in \Xi \\ \mathcal{P} \in \boldsymbol{\mathcal{P}}}} \sum_{\substack{\boldsymbol{p} \in \mathcal{P} \\ \boldsymbol{X} \in \mathcal{X}}} \left\| \boldsymbol{p} - \mathfrak{f}(\xi(\boldsymbol{X}), \boldsymbol{\alpha}) \right\|^2 \tag{3.3}$$

And the optimization problem is defined as:

$$\{\boldsymbol{\alpha}^*, \Xi^*\} = \underset{\boldsymbol{\alpha}, \Xi}{\operatorname{argmin}} E\left(\boldsymbol{\alpha}, \Xi, \mathcal{X}, \boldsymbol{\mathcal{P}}\right) \tag{3.4}$$

In other words, the objective is to minimize the sum of squared differences between the detected calibration board features $\boldsymbol{\mathcal{P}}$ and their modeled projections $\hat{\boldsymbol{\mathcal{P}}}(\Xi)$. The structure of this problem allows us to apply the Levenberg-Marquardt algorithm, which is one of the most efficient nonlinear optimization algorithms.

Strictly speaking, corner detector is a part of the real system, as it introduces some measurement noise. But we assume that this noise distribution does not depend on image coordinates of the extracted corner and is small. In other words, we assume that the camera gives us geometric information directly, even though it is not quite true. Theoretically, it is possible to calibrate the camera using the photometric information, but it has not been done in this work.

1.2 Data Acquisition

In the case of monocular calibration, a dataset is just a set of images of a calibration board taken with the same camera with the same hardware and software parameters. By hardware parameters we mean aperture and focus of the lens. Software parameters include resolution and image offset. Of course, the images should not undergo any geometric transformation. Calibration of cameras with autofocus and adaptive aperture is not in the scope of this work.

It is possible to calibrate a camera with a single image. But to make the calibration more robust and precise, it is recommended to acquire more images. In this work, it is usually between 50 and 200 images per camera.

Image Quality The chessboard pattern must be clear. Evidently we cannot expect the black squares to have image value 0 and that white squares to be saturated, but the contrast should be high. The more uniform the brightness across the squares of the same color, the better (compare Fig. 3.4abc against Fig. 3.4d). Image sharpness is of great importance, a blurred image is equivalent to an image of a lower resolution, hence the precision of parameters identified with blurred images is intrinsically lower.

One of the main drawbacks of this pattern is that overexposure deforms the features (Fig. 3.4e), that is the white squares dilate and fuse with one another. So the exposure and lighting must be adjusted so that the corners be acute and well-defined. Any kind of corner distortion leads to a drop in calibration quality. Reflections of bright light sources may lead to deterioration of the calibration pattern in the image (Fig. 3.4f) and must be avoided.

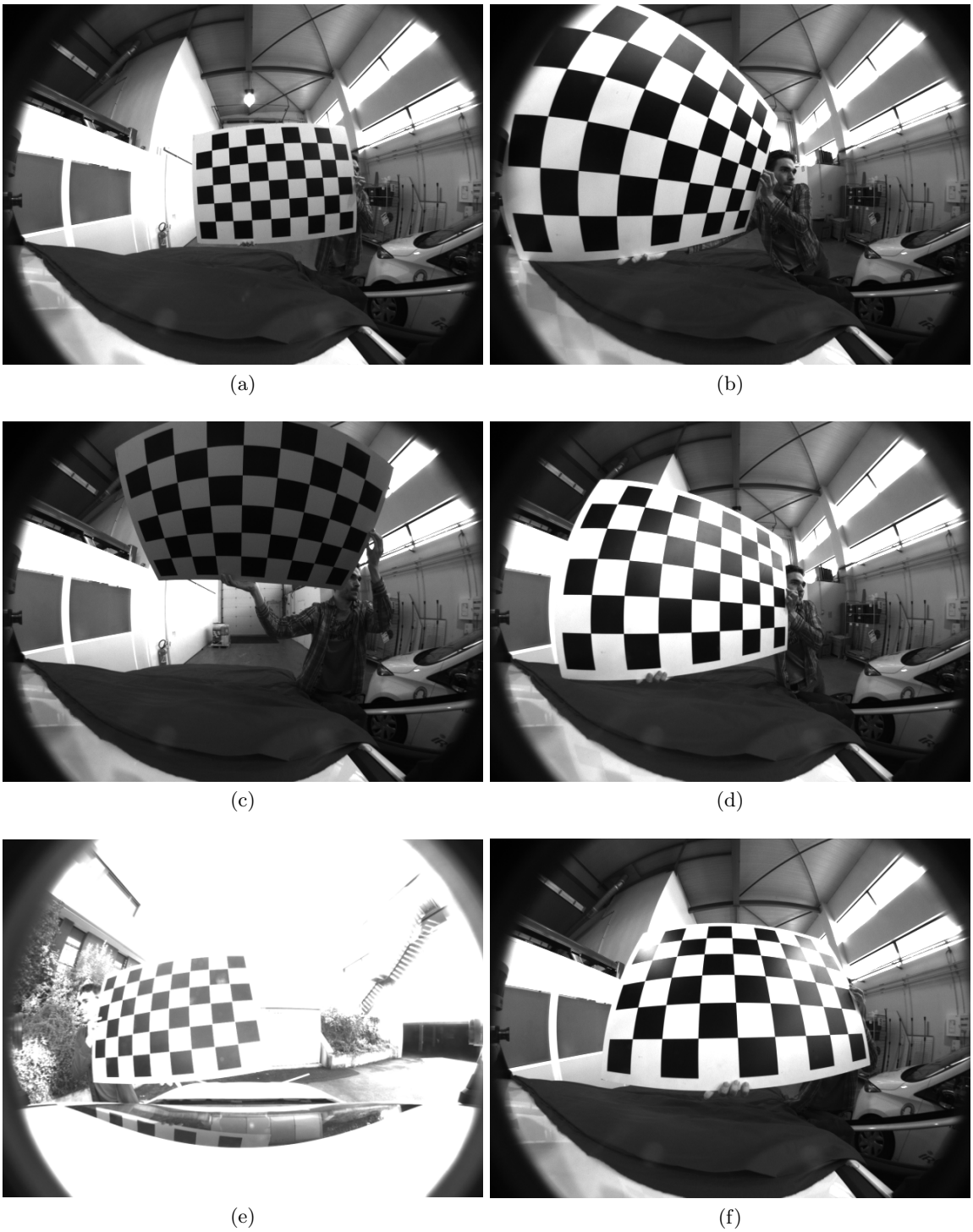
Usually, the acquisition is done with a fixed camera and moving calibration board (especially if the camera is attached to a car). In this case one should take care not to move the calibration board too fast as it might cause motion distortion. Another possible type of distortion is a nonlinear pattern deformation because of the rolling-shutter effect. Some cameras have this effect, other do not. Another motion effect is a simple motion blur (Fig. 3.5). One should take special care in the case of indoor acquisition, as artificial light is weaker, and to have a proper image dynamics (that is, difference between dark and light pixels) the exposure is longer.

Geometric Layout The layout of the calibration board in the images in the calibration dataset is important. The more uniformly the board is spread across the field of view, the better the intrinsic parameters are defined. The apparent board size in images matters as well. If the board is far away, its shape is close to rectangular for any distortion parameters (compare Fig. 3.4a and b). To make the distortion well-defined, we should have it appear within a single image. That is, it is better to have fewer images with large and distorted board rather than a lot of images with small boards.

It may be possible to compute the optimal board layout, but usually the board is handheld and it would be difficult to follow exactly the precomputed layout. On the other hand, it is not much of a problem to acquire more images to avoid overfitting.

It is mentioned in Sturm and Maybank (1999) that, in the case of a pinhole camera, only the board orientation matters for calibration singularities, and two images of a calibration board parallel to itself do not provide more information than only one of them, except that we have more features and it makes the algorithm more robust. Also it is mentioned that, if the calibration board is parallel to the normal plane, then it is a singular configuration. It means that the model is not fully identifiable and some of its parameters become dependent. In the case of the enhanced model, the dependence is non-linear and more complicated, but it is still a singularity. That is why it is important to have images with the board rotated about x and y axes.

The problem of the singularity is that in its neighborhood the noise gets amplified a lot. That is, if we have the board perfectly perpendicular to the optical axis, then the parameters are not defined. But in practice it is almost impossible, and even if we tried to align the camera and the board, there still would be some error, and the solution would be found. But the impact of the noise (detection noise mostly) in this case would be so strong, that the identified parameters could be quite far away from their real values. To summarize this paragraph about the geometric layout, the farther away from the singular configurations described above, the better the precision and the weaker the noise impact.



(a) (b)

(c) (d)

(e) (f)

Figure 3.4: Calibration images. (a),(b) Fair image quality, dark squares are uniform, the contrast is high. (c) Calibration board somewhat too dark, yet still usable in calibration. (d) Some reflections make the dark squares non uniform, which may cause slight corner displacement; also it makes the detection more challenging, since it is easier to adjust the detector to detect either clear sharp corners like on the bottom left, or slightly overexposed, like on the top right, but not both of them simultaneously. (e) A strong overexposure significantly decreases the corner detection precision. (f) strong reflections lead to saturation within dark squares which makes the corner detection completely impossible.

Figure 3.5: Motion blur.

2 Board Detection

The board detector is an essential part of a camera calibration system. Its ability to detect the board in challenging distorted images removes undesirable constraints on calibration data and facilitates the acquisition. Its precision has a direct impact on the calibration quality. In terms of robustness to distortion, the checkerboard seems to be the best solution, since for any kind of continuous distortion it is possible to reconstruct the grid using image gradient information.

At first, the OpenCV detector has been used for calibration purposes. But it has been found that this detector was a significant limitation since it did not manage to detect the pattern in many fisheye images, and when the pattern had been detected globally, local corner detection precision was poor as the detector is sensitive to overexposure.

The first step was to replace the OpenCV subpixel detector with a new one, while still using its global board detector for initialization purposes. Later this part has been replaced by a custom detector, adapted to fisheye images with high distortion. Let us proceed to its detailed description.

2.1 Detection

The first step is to detect pixels which correspond to the pattern corners. In order to do that, we follow these steps:

1. By means of a response function, find all pixels which are possible corner projections.

2. Discard unlikely candidates by analyzing the point neighborhood.

3. For the rest of candidates, using an algorithm of pixel search along strong brightness transitions, construct a graph of possible point connections.

4. Select a subgraph which corresponds to the pattern parameters.

Some intermediate steps are illustrated in Fig. 3.6.

2.1.1 Response Function

The calibration board corners are in fact saddle points of the brightness function. Saddle points of functions like $\mathbb{R}^2 \to \mathbb{R}$ have a sufficient condition which is a combination of two necessary conditions:

1. $\nabla I(\boldsymbol{p}) = 0$ — the extremum necessary condition.

2. $\det H < 0$ — a condition to check whether the Hessian matrix is *indefinite*. $\det H = \lambda_1 \lambda_2$, hence $\det H < 0 \Longrightarrow \operatorname{sign} \lambda_1 \neq \operatorname{sign} \lambda_2$

We can integrate both conditions in the following response function F:

$$F = -\det H - k \left\| \nabla I \right\|^4 = -I_{uu} I_{vv} + I_{uv}^2 - k(I_u^2 + I_v^2)^2 \tag{3.5}$$

If the image gradient is substantial in $\boldsymbol{p}$, F is negative. But if it is negligible we can consider the point as an extremum. Then the Hessian determinant comes into play: if it is negative, F becomes positive. There is an adjustment parameter k which defines the balance between the two criteria. In our application $k = 0.001$. The fourth power of gradient is necessary to have the same units for both terms: $1/\mathrm{px}^4$.

Let Ω_r define a punctured disc of radius r:

$$\Omega_r : \{\boldsymbol{q} \in \mathbb{R}^2 \mid 0 < \|\boldsymbol{q}\| <= r\} \tag{3.6}$$

The set $\boldsymbol{P}$ of all local maximum locations of F is defined as follows:

$$\boldsymbol{P} = \{\boldsymbol{p} \mid F(\boldsymbol{p}) > F(\boldsymbol{p} + \boldsymbol{q}) \; \forall \boldsymbol{q} \in \Omega_r\} \tag{3.7}$$

In our case $r = 3$.

2.1.2 Reducing the Number of Candidates

To simplify the step of finding the grid among the detected local maxima, we check some necessary conditions for a detected point to be a board inner corner.

Number of Brightness Transitions If we sample an image around a calibration board inner corner, we will find in the sample sequence two strong positive and two strong negative transitions ("steps"). Moreover, the distance between two transitions of the same type is one half of the circle length. We can check the both of these facts for the potential candidates. The sampling to get a sequence of image values $\{I_1..I_L\}$ is done using the curve rasterization algorithm described in Chapter 4.

The properties are checked for a range of circle radii. the test must be successful for all the radii within the range to accept the point. But sometimes the pattern is so small that a radius of 8 px gets to the neighbor square, and the test fails. On the other hand, in some conditions (especially overexposure, Fig. 3.8), taking this radius too small also leads to the test failure. That is why the test is done as follows:

1. Minimum radius $r_{\min}$ is taken from 1 to 6

2. Maximum radius $r_{\max} = r_{\min} + \max(3, r_{\min})$

3. The test must be successful for at least one pair of $r_{\min}, r_{\max}$

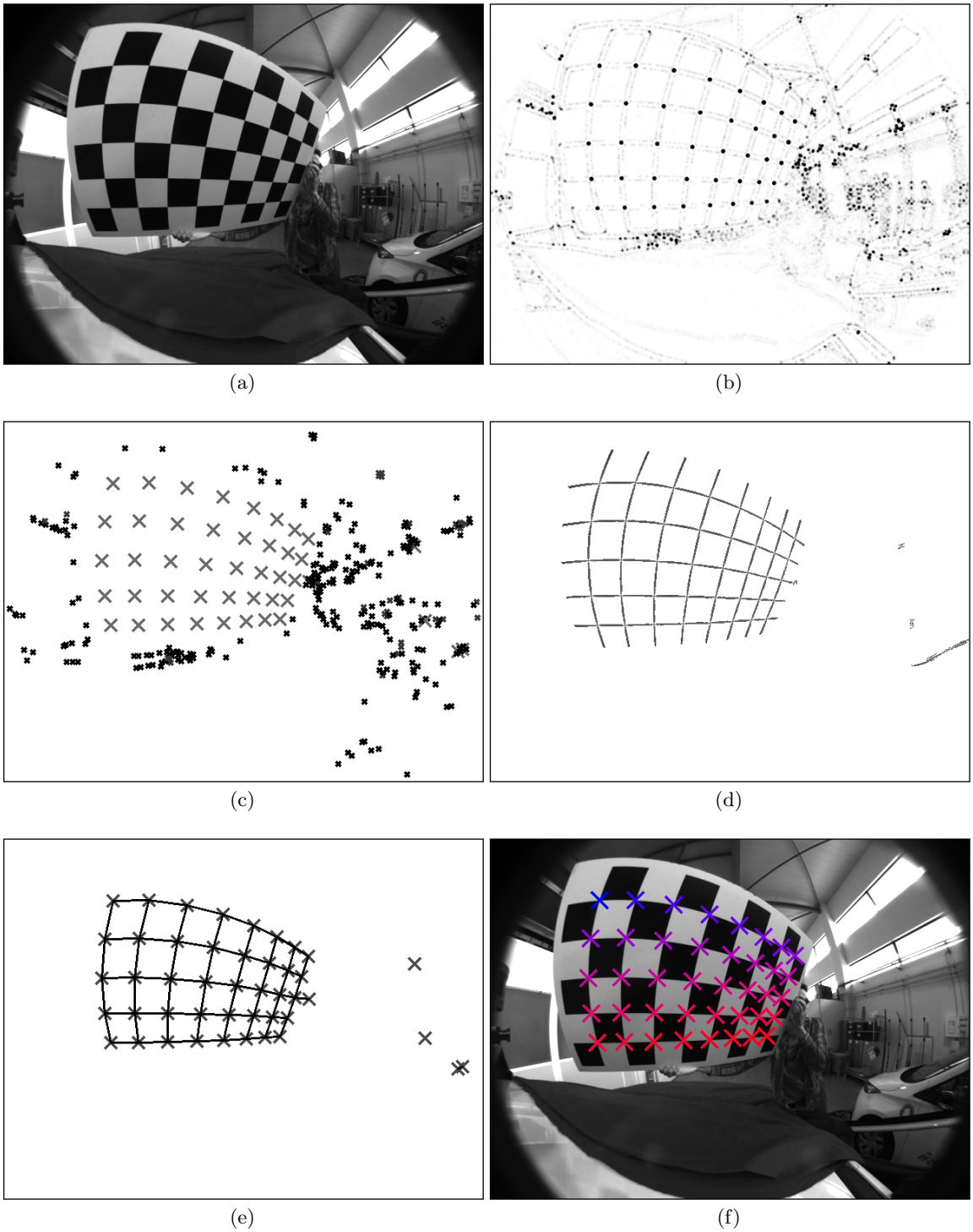


Figure 3.6: Board detection steps. (a) Input image. (b) Corner detector response. (c) Detected maxima; final candidates are marked with larger crosses. (d) Pixel search, performed along strong properly oriented edges. (e) Constructed graph. Small crosses on edges mark edge directions. (f) Result after graph analysis.

To test the transition property we differentiate the sample sequence using the central differences:

$$\Delta_i = I_{i+1} - I_{i-1} \quad i = 1..L \tag{3.8}$$

keeping in mind that actually it is a loop, so the last sample is followed by the first. Then we take the three strongest local maxima $\Delta_{\text{max1}} > \Delta_{\text{max2}} > \Delta_{\text{max3}}$ and check the following property:

$$\Delta_{\text{max2}} > \alpha_1 \Delta_{\text{max1}} \qquad \Delta_{\text{max3}} < \alpha_2 \Delta_{\text{max2}} \tag{3.9}$$

Where $\alpha_1, \alpha_2 \in]0, 1[$ are algorithm parameters.

A similar test is done for the three strongest local minima. The radial symmetry implies that the circular distance between the two strongest maxima should be $L/2$. Sometimes the maximum is shifted by a pixel due to the discrete nature of the response function, and hence the distance between the transitions can be different from the half of the sample vector length. We have chosen a threshold of $L/2 - 2$, where L is the number of samples. Let i_1, i_2 be the indices of the two maxima. Then the condition is the following:

$$L/2 - 2 < |i_1 - i_2| \ < L/2 + 2 \tag{3.10}$$

In Fig. 3.6c, small black crosses represent the points filtered out by the described method.

Scale Invariance In OpenCV the scale invariance property is used to do the subpixel detection. It is formulated as follows:

$$\boldsymbol{p}^* = \underset{\boldsymbol{p}}{\operatorname{argmin}} \sum_{\boldsymbol{q} \in \Omega_r} w(\boldsymbol{q}) \nabla I(\boldsymbol{p} + \boldsymbol{q}) \cdot \boldsymbol{q} \tag{3.11}$$

Here w is a weight function, Ω_r is defined in (3.6). Since all $\boldsymbol{q} \in \Omega_r$ are predefined, w includes the normalization factor $\|\boldsymbol{q}\|^{-1}$. The meaning of this cost function is that the gradient should be perpendicular to the rays which come from $\boldsymbol{p}$. In other words, it means that the neighborhood of $\boldsymbol{p}$ must be scale-invariant.

Later we describe yet another subpixel corner detector. However, such a response appears to be an efficient criterion to discard false-positive detections. For each candidate $\boldsymbol{p} \in \boldsymbol{P}$ the following cost function is computed:

$$E(\boldsymbol{p}) = \sum_{\boldsymbol{q} \in \Omega} w(\boldsymbol{q}) (\nabla I(\boldsymbol{p} + \boldsymbol{q}) \cdot \boldsymbol{q})^2 \tag{3.12}$$

Fig. 3.6c shows that most false positives are filtered out using the two simple techniques described above.

2.1.3 Graph Construction and Grid Selection

Once we have the corner candidates, the next step is to assemble them into a graph. As it has been mentioned before, the task is facilitated thanks to the edges which connect the corners on the image. By following the strong gradient via breath-first search we find which points are interconnected. (Fig. 3.6d,e) the graph is represented as a directed graph with two arcs per edge (one in either direction). Each arc has a sign, depending on how the gradient is oriented along it. For example, if while traversing an arc, we see the black square on the left then the arc is positive. This property will be important for the following step, subgraph selection.

Each point is then checked whether it can be the origin and whether we can select a subgraph with a grid structure. Generally we want to start with the upper left corner, so the points are checked in the order of increasing $u + v$. Let the grid be $N_u \times N_v$. For each potential pattern origin, first we are looking for two sequences $\mathcal{U}_1 \subset \mathbb{R}^2$ and $\mathcal{V}_1 \subset \mathbb{R}^2$ of connected corners with the following properties:

1. Arcs in each sequences have alternate signs.

2. The first arcs of $\mathcal{U}_1$ and $\mathcal{V}_1$ have opposite signs.

3. $\mathcal{U}_1$ has at least N_u corners (including the origin)

4. $\mathcal{V}_1$ has at least N_v corners

5. If $\boldsymbol{a} = \mathcal{U}_1[2] - \mathcal{U}_1[1]$ and $\boldsymbol{b} = \mathcal{V}_1[2] - \mathcal{V}_1[1]$, then:

$$0 < \varepsilon < \frac{\boldsymbol{a}_u \boldsymbol{b}_v - \boldsymbol{a}_v \boldsymbol{b}_u}{\|\boldsymbol{a}\| \, \|\boldsymbol{b}\|} \tag{3.13}$$

   where ε is a small threshold to make sure that $\boldsymbol{a}$ and $\boldsymbol{b}$ are not parallel.

Property 5 assures that the two "base" arcs are properly oriented. Based only on the first four properties, we can select the inverted version of the pattern starting from the upper right corner, for example.

If such sequences have been found, then we discard all the "excess" points, if there are any, to make $\|\mathcal{U}_1\| = N_u$ and $\|\mathcal{V}_1\| = N_v$. Then we try to select a sequence $\mathcal{U}_i$ which starts from $\mathcal{V}_1[i]$, $i = 2..N_v$ and has similar properties as above:

1. Arcs in the sequences have alternate signs.

2. $\mathcal{U}_i$ has at least N_u corners (including the origin)

3. $\boldsymbol{a} = \mathcal{U}_i[2] - \mathcal{U}_i[1]$ and $\boldsymbol{b} = \mathcal{V}_1[i] - \mathcal{V}_1[i-1]$ should satisfy (3.13).

Once $\mathcal{U}_2..\mathcal{U}_{N_v}$ have been detected, the last step is to validate the grid. In order to do that, a similar process is done: find sequences $\mathcal{V}_j$, $j = 2..N_u$ which start from $\mathcal{U}_j$ with similar properties as above. The grid is validated if $\mathcal{U}_i[j] = \mathcal{V}_j[i]\ \forall i = 2..N_v, j = 2..N_u$

If a subgraph with such properties has been found, then the algorithm returns it. If not, the algorithm reports that it has failed to find the pattern.

2.2 Subpixel Refinement

The detection step finds the pattern corners with pixel precision. Yet the image provides enough information to make the estimation more accurate. The idea is to fit two segments along square edges to maximize the gradient flow. The objective function to maximize is defined as follows:

$$E = \int\limits_{S_1 \backslash \boldsymbol{p}} \frac{\boldsymbol{s} \otimes \nabla I}{\|\boldsymbol{s}\|} \mathrm{d}s - \int\limits_{S_2 \backslash \boldsymbol{p}} \frac{\boldsymbol{s} \otimes \nabla I}{\|\boldsymbol{s}\|} \mathrm{d}s \tag{3.14}$$

where S_1 and S_2 are the two segments; $\boldsymbol{s}$ is the vector relating intersection $\boldsymbol{p}$ with the actual point; by $\otimes$ we mean the following operation:

$$\boldsymbol{a} \otimes \boldsymbol{b} = a_u b_v - a_v b_u \tag{3.15}$$

We exclude $\boldsymbol{p}$ to avoid division by zero in the definition. The subtract the second term from the first because for one segment the transition dark-bright happens clockwise, while for the other it is counterclockwise. Fig. 3.7 illustrates a presumably optimal configuration.

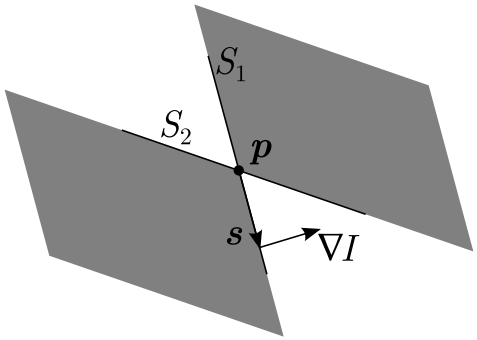


Figure 3.7: An illustration of the optimal subpixel corner detector

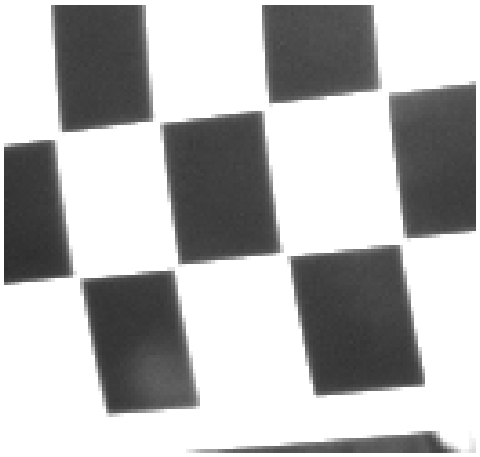


Figure 3.8: Because of an overexposure the black squares are separated and traditional detectors, based on local image filtering, fail to detect the corner in the middle of the white separation strip. This example is exaggerated, but even a slight overexposure, not visible to the bare eye, gives a characteristic noise pattern after the calibration.

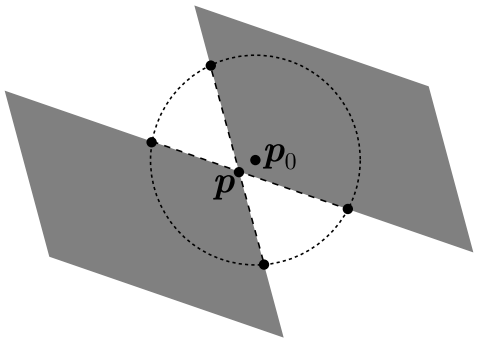


Figure 3.9: An illustration of the initialization process. $\boldsymbol{p}_0$ is the initial guess. By looking for the brightness transitions along a circle and connecting them we find the initial approximation of the point position $\boldsymbol{p}$ and the segments' orientation.

Initialization We assume that the initial approximation of the corner position is in the neighborhood of the real one. The size of this neighborhood is chosen depending on the image resolution, expected square size, and the level of overexposure. The latter appears to be the main source of non-Gaussian noise. The initialization of the segments is done in the spirit of FAST corner detector Rosten and Drummond (2006). By iterating along a circular path around the initial point we look for brightness transitions.

Four strong transitions must be found. Moreover we expect them to have a particular geometric configuration. Brightness increases must be located diametrically in the ideal case.

To find the correct transitions we add a constraint that the transitions of the same sign (see Fig. 3.9). are separated at least by $\pi/2$ along the circle. The algorithm goes as follows:

1. Sample the image along a circular path around the initial guess, given by another detector.
2. Estimate the transition strength by using the central differences.
3. Find the largest increase.
4. Find the largest increase within the opposite semicircle.
5. Repeat the procedure for the decreases.
6. Find the intersection between the segment relating the increase points and the one, relating the decrease points. It is the initial corner position for the optimization loop.
7. Find the segments' orientation and use it as the initial orientation in the optimization process.

Optimization The corner is parametrized as follows (see Fig. 3.10):

- $\boldsymbol{p} = (u, v)$ — the corner's position.
- ϑ_1, ϑ_2 — the segments' orientation.
- h — shift in gradient due to the blur and overexposure.

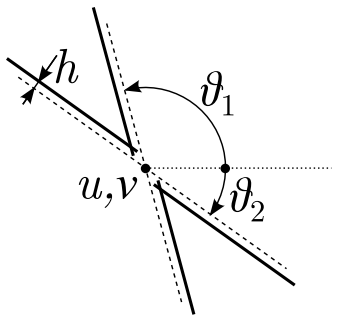


Figure 3.10: Parametrization for the subpixel corner detector. The gradient flow is computed along the solid lines.

The segments used in the optimization are not exactly the same as the ones used in the initialization. They inherit their orientation, but their length is fixed and they are centered at $\boldsymbol{p}$. The discretized version of the cost function is expressed as follows:

$$E = \sum_{i=1}^{2} \sum_{j=1}^{M} (-1)^i \left(\nabla I(\boldsymbol{p} + \Delta \boldsymbol{p}_{ij}) \otimes \boldsymbol{r}_i - \nabla I(\boldsymbol{p} - \Delta \boldsymbol{p}_{ij}) \otimes \boldsymbol{r}_i \right) \tag{3.16}$$

where $\boldsymbol{r}_i = (\cos \vartheta_i, \sin \vartheta_i)$ is the segment direction vector; Δu_{ij} and Δv_{ij} define the displacement:

$$\Delta \boldsymbol{p}_{ij} = \begin{pmatrix} j \cos \vartheta_i + (-1)^i h \sin \vartheta_i \\ j \sin \vartheta_i - (-1)^i h \cos \vartheta_i \end{pmatrix} \tag{3.17}$$

Fig. 3.11 shows how the proposed detector improves the corner position in comparison to the OpenCV subpixel detector. The comparison of the error distribution after calibrating cameras will be given in the next section.

Table 3.1: Detector temporal performance — average time per image. Failed detections are taken into account for the average time.

Dataset		OpenCV		Proposed Detector	
Resolution	Images	Detected	T, s	Detected	T, s
640×480	284	115	0.111113	257	0.0313478
640×480	276	112	0.117382	202	0.035969
1280×960	552	311	0.251513	407	0.0929256

One of the advantages of the proposed method is its performance, represented in Table 3.1. The proposed detector finds more patterns and does it quicker than the OpenCV detector.

We have tested the time performance of another existing detector from Schönbein *et al.* (2014), implemented in MATLAB. It processed a dataset of 54 640×480 images in 572 seconds, or about 10 seconds per image, which is much slower than C++ detectors compared here. Such a difference in performance may be due to the interpreted nature of MATLAB, though it is difficult to say how much this time can be improved by using a different language. For real applications, the proposed algorithm with its current implementation is a good alternative when large datasets have to be treated or real-time performance is required.

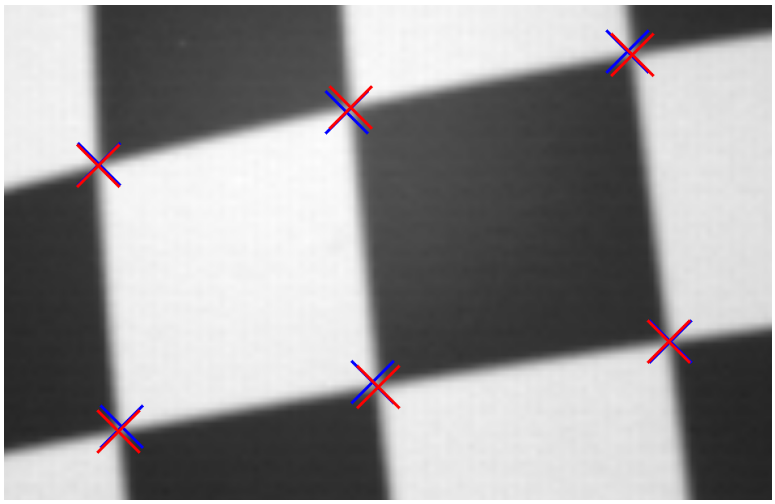


Figure 3.11: An example of corner detection improvement (image is zoomed). Blue — corners given by the OpenCV detector, Red — improved corners.

3 Parameter Initialization

Intrinsic Parameter Initialization This calibration technique seems to be quite robust with respect to the intrinsic parameter initialization. Yet it is better to have reasonable initial values. If the camera to be calibrated is fisheye, then the distortion parameters can be chosen as $\alpha = 0.5$ and $\beta = 1$.

Concerning the projection matrix, these parameters must be chosen so that the projection center be in the middle of the image and that the horizontal field of view be about $180°$. It is easy to achieve if we look at the Enhanced model projection properties for $\alpha = 0.5$, when the directions at $90°$ with respect to the optical axis are mapped to a circle of radius 2 on the normal plane. So, we suggest the following initialization:

1. $u_0 = u_{\max}/2$
2. $v_0 = v_{\max}/2$
3. $f_v = f_u = u_0/2$

Of course, for low-distortion narrow-angle cameras the initial parameters might be quite different.

3.1 Local Minimum of Calibration Board Pose

Unfortunately, the experiments show that the calibration problem has multiple local minima. The scheme of a possible local minimum is given in Fig. 3.12. The practice shows that without an appropriate initialization, this configuration occurs quite frequently. An example of such a local minimum is shown in Fig. 3.13. Notice that the cost function and the data, that is the extracted grid, are the same in both cases.

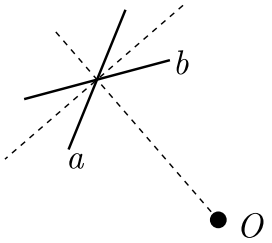


Figure 3.12: An illustration of reprojection error local minimum (view from the top): a is the real board position, b is the computed position; O is the camera origin.

3.2 Calibration Board Pose Initialization

The initialization is done using the initial intrinsic parameters. These parameters may be quite different from their true values. Fig. 3.15 shows that the corner reprojections don't actually fit the detected corners. As long as initial projection parameters are arbitrary, the position estimation is also poor and cannot be considered as a real estimation. But it brings the initial parameters of the global optimization problem to a "valley" from where they will converge to the global optimum.

Direct Initial Pose Computation This procedure is necessary to avoid the local minimum described above. Hence, the main goal is to get the initial orientation right. We use a simple heuristics: if one of two parallel edges has a smaller angular size, it is farther away from the camera. Knowing the model size and assuming that the edges are close to perpendicular to the camera direction, we can directly estimate the distance to them. Let us first define the necessary notation (Fig. 3.14).

- Subscripts T, B, L, R correspond to "top", "bottom", "left", "right" respectively (for example TL means "top-left").
- Let $\boldsymbol{X}_{\mathrm{TL}} \in \mathbb{R}^3$ be a calibration board corner.
- Let $\boldsymbol{d}_{\mathrm{TL}} \in \mathbb{S}^2$ be a reconstructed unit direction vector towards a calibration board corner. Initial guess for intrinsic parameters is used in the reconstruction.

$$\boldsymbol{d}_{\mathrm{TL}} = \frac{X_{\mathrm{TL}}}{\|X_{\mathrm{TL}}\|} \tag{3.18}$$

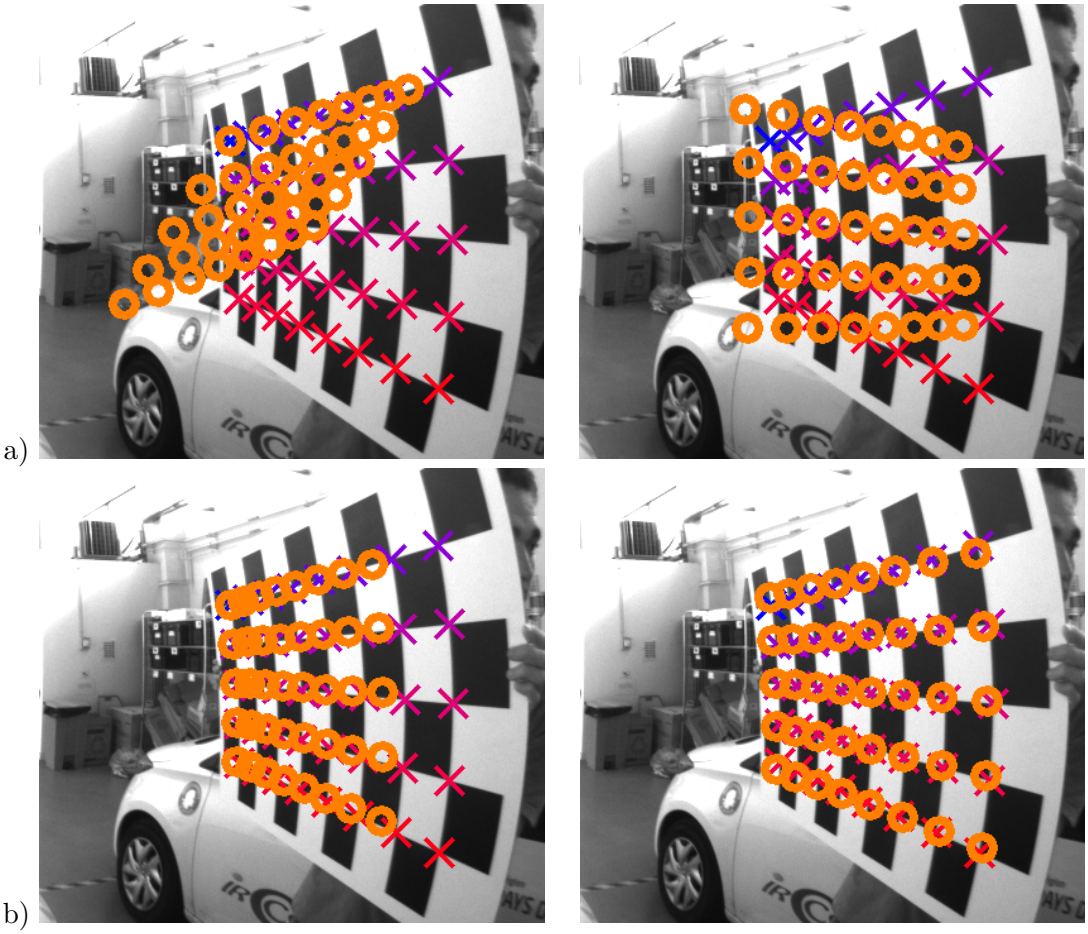


Figure 3.13: The impact of initialization on convergence. On the left we have a precomputed pose estimation. On the right the result of reprojection error minimization over the board pose. Notice that the intrinsic parameters are arbitrary since the calibration has not yet been done. (a) a poor initialization may lead to a local minimum. (b) a better initialization makes it possible to find the global solution.

- Let $\boldsymbol{e}_T$, $\boldsymbol{e}_B$, $\boldsymbol{e}_L$, $\boldsymbol{e}_R$ be calibration board edge vectors. They are defined as follows:

$$\begin{aligned} \boldsymbol{e}_T &= \boldsymbol{d}_{\mathrm{TR}} - \boldsymbol{d}_{\mathrm{TL}} \qquad & \boldsymbol{e}_B &= \boldsymbol{d}_{\mathrm{BR}} - \boldsymbol{d}_{\mathrm{BL}} \\ \boldsymbol{e}_L &= \boldsymbol{d}_{\mathrm{BL}} - \boldsymbol{d}_{\mathrm{TL}} \qquad & \boldsymbol{e}_R &= \boldsymbol{d}_{\mathrm{BR}} - \boldsymbol{d}_{\mathrm{TR}} \end{aligned} \tag{3.19}$$

- Let $L_u = \|\boldsymbol{X}_{\mathrm{TR}} - \boldsymbol{X}_{\mathrm{TL}}\|$ and $L_v = \|\boldsymbol{X}_{\mathrm{BL}} - \boldsymbol{X}_{\mathrm{TL}}\|$ be the board dimensions.

Let us compute the direction scales:

$$\begin{aligned} \lambda_T &= \frac{L_u}{\|\boldsymbol{e}_T\|} \qquad & \lambda_B &= \frac{L_u}{\|\boldsymbol{e}_B\|} \\ \lambda_L &= \frac{L_v}{\|\boldsymbol{e}_L\|} \qquad & \lambda_R &= \frac{L_v}{\|\boldsymbol{e}_R\|} \end{aligned} \tag{3.20}$$

Now we can estimate the board position:

$$\hat{\boldsymbol{X}}_{\mathrm{TL}} = \boldsymbol{d}_{\mathrm{TL}} \min(\lambda_T, \lambda_L) \tag{3.21}$$

Using min is a heuristics. We can illustrate the idea behind with an example. Let us assume that the calibration board is rotated about its x-axis, and $\boldsymbol{X}_{\mathrm{TL}}$ is farther away from the

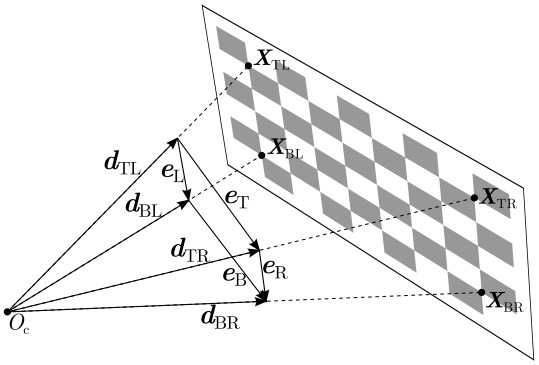


Figure 3.14: Definition of notation for initial pose computation.

camera than $\boldsymbol{X}_{\mathrm{BL}}$ (Fig. 3.14). It means that the top edge appears smaller than the bottom edge because it is still parallel to the image plane but just farther away. On the other hand, the left edge appears smaller because it is twisted and its apparent size is roughly proportional to the cosine of the rotation angle about x-axis. Since the change in apparent size due to the distance is generally smaller than the one due to the rotation, we take the minimum scale factor. If there is more rotation about y than x, then λ_L will be preferred over λ_T, and vice versa.

We estimate the positions of TR and BL corners in a similar way:

$$\begin{aligned}\hat{\boldsymbol{X}}_{\mathrm{TR}} &= \boldsymbol{d}_{\mathrm{TR}} \min(\lambda_T, \lambda_R) \\ \hat{\boldsymbol{X}}_{\mathrm{BL}} &= \boldsymbol{d}_{\mathrm{BL}} \min(\lambda_B, \lambda_L)\end{aligned} \tag{3.22}$$

Then let us construct a local basis. $\boldsymbol{x}_{\mathrm{b}}$ is aligned with the estimated board top edge:

$$\boldsymbol{x}_{\mathrm{b}} = \text{normalize}\left(\hat{\boldsymbol{X}}_{\mathrm{TR}} - \hat{\boldsymbol{X}}_{\mathrm{TL}}\right) \tag{3.23}$$

$\boldsymbol{y}_{\mathrm{b}}$ is the component of the reconstructed left edge, orthogonal to $\boldsymbol{x}_{\mathrm{b}}$:

$$\boldsymbol{y}_{\mathrm{b}} = \text{normalize}\left(\left(I_3 - \boldsymbol{x}_{\mathrm{b}}\boldsymbol{x}_{\mathrm{b}}^T\right)\left(\hat{\boldsymbol{X}}_{\mathrm{BL}} - \hat{\boldsymbol{X}}_{\mathrm{TL}}\right)\right) \tag{3.24}$$

The definition of $\boldsymbol{z}_{\mathrm{b}}$ is straight-forward:

$$\boldsymbol{z}_{\mathrm{b}} = \boldsymbol{x}_{\mathrm{b}} \times \boldsymbol{y}_{\mathrm{b}} \tag{3.25}$$

The board transformation then is defined as follows:

$${}^{\mathrm{c}}T_{\mathrm{b}} = \begin{pmatrix} \boldsymbol{x}_{\mathrm{b}} & \boldsymbol{y}_{\mathrm{b}} & \boldsymbol{z}_{\mathrm{b}} & \hat{\boldsymbol{X}}_{\mathrm{TL}} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.26}$$

Pose Refinement The initialization described above improves chances that the board is in the basin of attraction of the global minimum. Then we can refine it to get as close to the optimization valley as possible. To do that we just solve the optimization problem described in (3.4) individually for each board while fixing $\boldsymbol{\alpha}$:

$$E_{\mathrm{init}}\left(\boldsymbol{\alpha}, \xi, \mathcal{X}, \mathcal{P}\right) = \sum_{\substack{\boldsymbol{p}\in\mathcal{P} \\ \boldsymbol{X}\in\mathcal{X}}} \left\|\boldsymbol{p} - \mathfrak{f}(\xi(\boldsymbol{X}), \boldsymbol{\alpha})\right\|^2 \tag{3.27}$$

$$\xi_{\mathrm{init},i} = \underset{\xi}{\operatorname{argmin}}\, E_{\mathrm{init}}\left(\boldsymbol{\alpha}, \xi, \mathcal{X}, \mathcal{P}_i\right) \tag{3.28}$$

The initial value of ξ is defined by (3.26). The results of both initialization steps are presented in Fig. 3.15

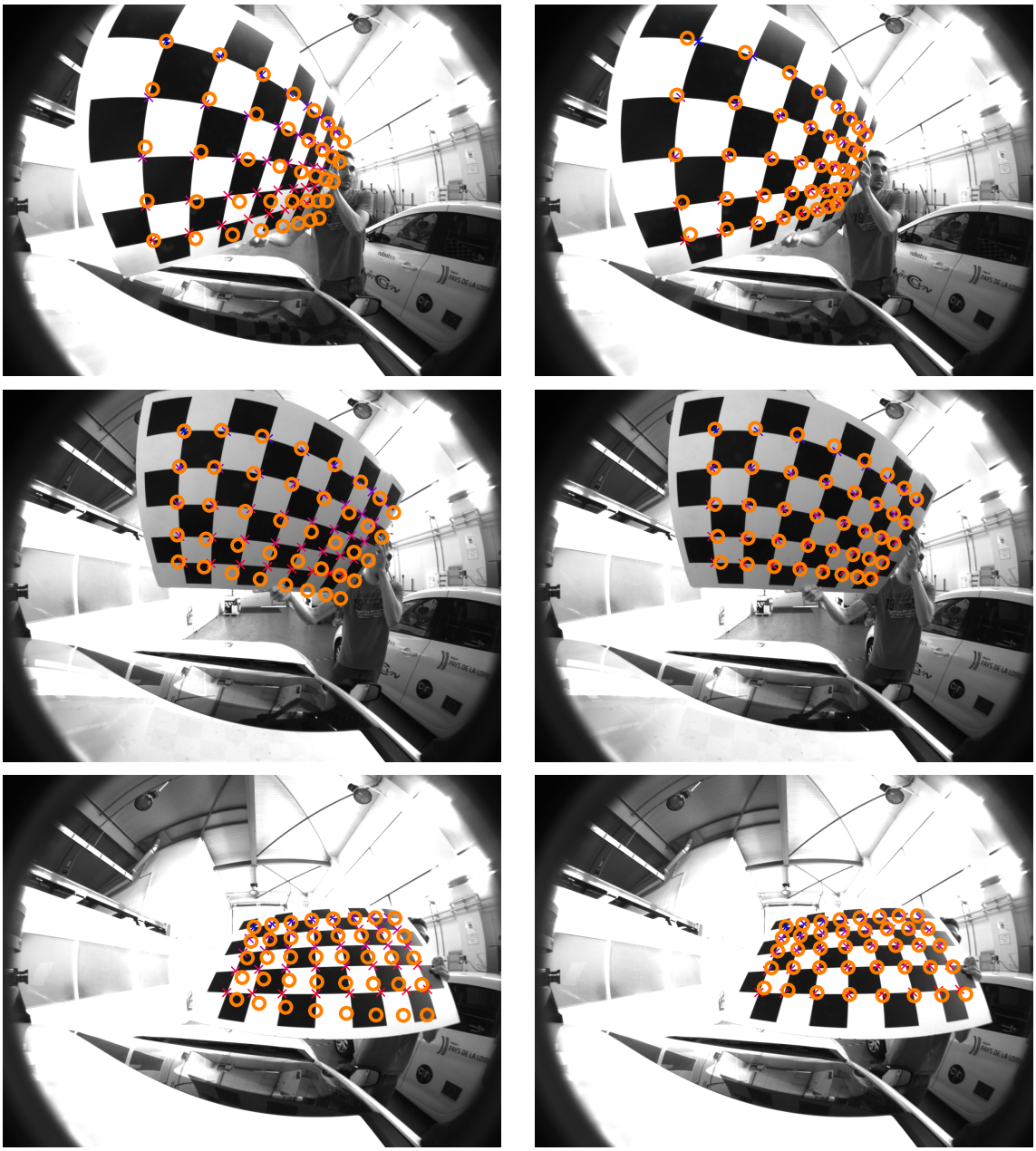


Figure 3.15: Extrinsic initialization for guessed intrinsic parameters. Left — direct pose calculation. Right — after optimization.

4 Calibration Using the Enhanced Camera Model

4.1 Monocular Calibration

Here we present some results related to monocular calibration. First, a comparison of board detectors is done. Then a comparison between different fisheye camera models is presented. For solving we use a C++ library called *Ceres-Solver* from Agarwal *et al.* (2010).

4.1.1 Detection Comparison

To compare different corner detectors, the same raw dataset of 436 images has been used for calibration. The OpenCV detector detected the pattern in 142 of them, whereas the proposed detector detected the board in 320 of them. Concerning the images in which the pattern has not been detected by the latter detector, the board there was either partially invisible or contained strong reflections. The resulting residual distribution is presented in Fig. 3.16. The figure shows that, for the OpenCV detector the error spread is much wider, which means that the proposed detector is more robust.

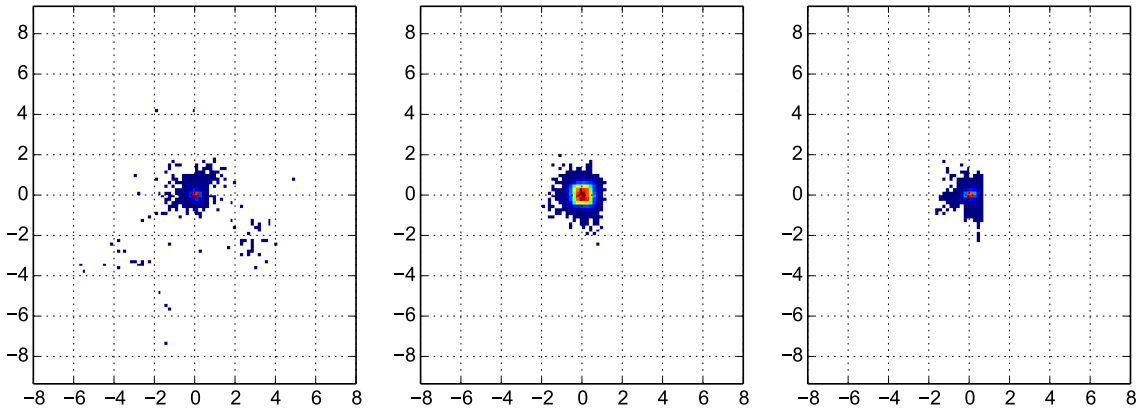


Figure 3.16: Calibration results for different detectors — final reprojection error distribution. From left to right: OpenCV detector, the proposed detector, and the proposed detector with subpixel refinement. The novel detector detects the board in more images with no evident outliers.

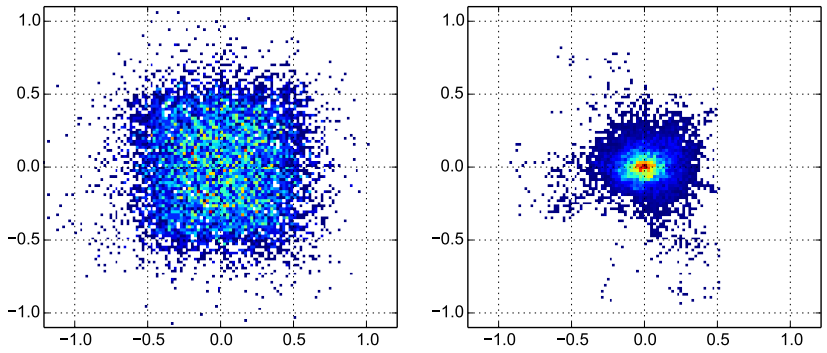


Figure 3.17: Subpixel detection, the two right plots from Fig. 3.16. Left — pixel detector; right — subpixel refinement. Notice the squarish shape of the left distribution which fits a 1×1 px area.

If we actually discard images which the OpenCV detector fails to process accurately, and which generate large errors, the error distribution will be narrower than the one, given by the

new detector (0.12 vs 0.14 px standard deviation). But we should remember the fact that OpenCV discarded right away the most challenging distorted images while processing only relatively "easy" cases.

After we have discarded problematic images, which are slightly blurred or distorted because of the motion or, perhaps, because of the calibration board slight bending, the resulting error distribution became even more compact (Fig. 3.17). The number of images used for this calibration is 260. You can see on the left that the noise pattern is squarish. It happens because the measurement noise in the absence of subpixel refinement has a uniform distribution with a square support 1×1 px. This distribution disappears once the subpixel detector is applied, and the standard deviation of the distribution goes down from 0.30 to 0.14 px.

4.1.2 Model Comparison

Six lenses have been calibrated using the Enhanced Unified Model: four fisheye lenses of two different models and two perspective lenses. Their aliases, models, focal lengths and fields of view are given in Table 3.2.

Table 3.2: Monocular calibration results. Second and third lines represent different lenses of the same model. σ_x and σ_y represent the reprojection error distribution after the calibration.

Lens alias	Lens model	focal length, mm	horizontal field of view, deg
fisheye 1	CF5M1414	1.4	182
fisheye 2			
fisheye 3	FE185C057HA-1	1.8	185
fisheye 4			
perspective 1	DF6HA-1B	6	56
perspective 2	COMP-M0814-MP	8	42

The image resolution in all cases is 1296×966. The standard deviations of the reprojection errors along x and y axes as well as the number of images used in the calibration are given in Table 3.3. The first four fisheye lenses have large α. Lenses of the same model demonstrate the same intrinsic parameters. The last two lenses are low-distortion, narrow-angle, and their values of α are significantly smaller than for fisheye. Fig. 3.18 shows the projection curves corresponding to the calibrated lenses.

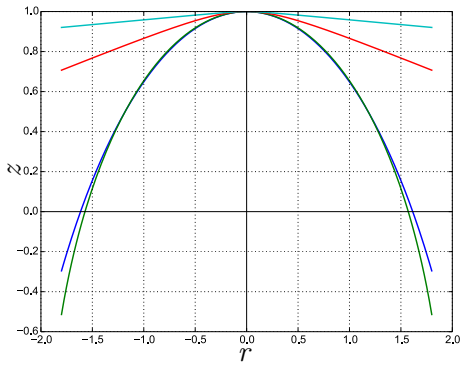


Figure 3.18: Projection curves for the calibrated lenses. Blue — fisheye 1 and 2, green — fisheye 3 and 4, red — perspective 1, cyan — perspective 2.

Fig. 3.19 is the reprojection of the grid after the calibration. Fig. 3.20–3.21 show the undistortion using the model. Fig. 3.21 shows the undistortion of a region on the border of the image: lines that are straight on the calibration board appear straight. To perform this

undistortion we have to rotate the virtual camera with respect to the real camera. Otherwise the region would not be in the field of view of the virtual pinhole camera.

The Enhanced Model (EUCM) has been compared to the Unified Model, with (UCM-D) and without (UCM) the distortion layer described in Mei and Rives (2007). The resulting reprojection error values are given in Table C.3. The first thing that we see is that the change in σ from EUCM to UCM-D is negligible. On the other hand the computation time increases by an order of magnitude. If we compare UCM and EUCM, then we see that there is no significant change in computation time. But there is a significant improvement of precision for fisheye 1 and 2, while the performance of either model is almost the same for fisheye 3 and 4. It becomes clear by looking at Table 3.3 where for fisheye 3 and 4 β is close to 1.0, while for fisheye 1 and 2 it is not the case. And β is the parameter that makes the difference between two models. We can see a certain improvement for perspective 1 (σ changes from 0.24 to 0.11).

Another criterion that we can use to compare different lenses is the computed pose. We took the pose computed with UCM-D as a reference and compared it to poses computed using UCM and EUCM. The result is presented in Fig. C.4. The error is measured in millimeters and milliradians. The average distance camera-board and rotation angle for poses computed using UCM-D are about 1 m and 1 rad respectively. For fisheye 1 and 2 we can see that the difference between UCM and UCM-D is significantly larger than between EUCM and UCM-D. For fisheye 3, 4 and perspective 1 all three models give close results with little difference. For perspective 2, however, we see that both UCM and EUCM are quite far from UCM-D. Moreover, if we compute the error between UCM and EUCM we get 3.0 mm and 1.6 mrad. In the absence of ground truth, we cannot make an ultimate conclusion, but it seems like the complex model overfits data, since UCM-D has 10 parameters and the data closely follows the Pinhole model. So, UCM and EUCM give better precision.

Table 3.3: Monocular calibration results. σ_x and σ_y represent the reprojection error distribution after the calibration.

Lens	Images	σ_x, px	σ_y, px	α	β
fisheye 1	142	0.128	0.121	0.569	1.19
fisheye 2	105	0.114	0.118	0.571	1.18
fisheye 3	143	0.079	0.064	0.629	1.02
fisheye 4	87	0.084	0.075	0.626	1.02
perspective 1	50	0.111	0.078	0.076	6.08
perspective 2	82	0.120	0.142	0.019	10.0

4.2 Stereo Calibration

4.2.1 Extrinsic Calibration Approach

Using a calibration board, we can calibrate not only intrinsic parameters of cameras, but extrinsic parameters of stereo systems as well. For that we need to couple the cameras' positions via images of the calibration board taken simultaneously.

We can decouple the extrinsic and intrinsic calibrations, that is calibrate the cameras' intrinsics separately, and then calibrate the stereo transformation. But a better way is to calibrate the whole system simultaneously and unify three calibration problems (two intrinsic and one extrinsic) into one.

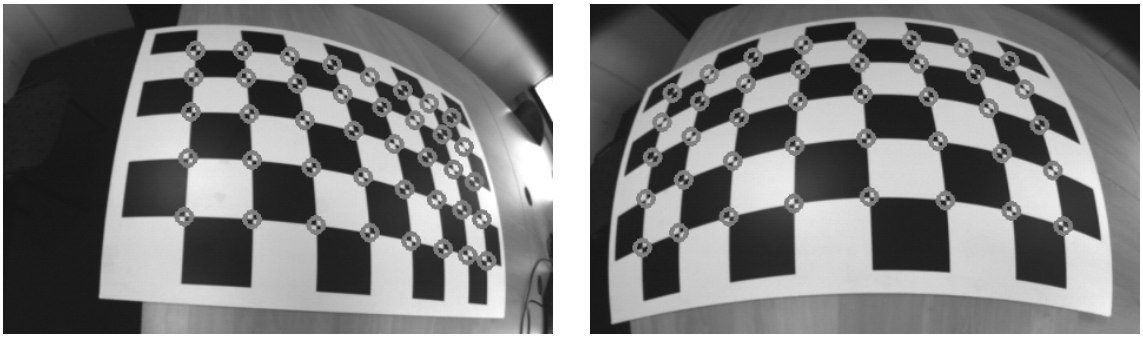


Figure 3.19: Grid projection after calibration.

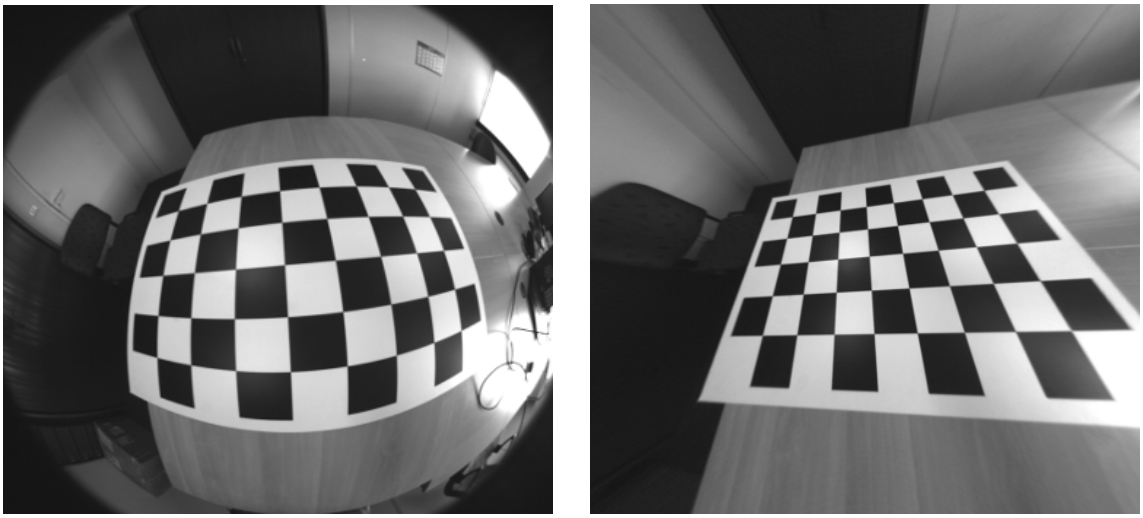


Figure 3.20: Undistortion using the calibrated model — all the straight lines after undistortion become straight. Here the board is in the middle of the image.

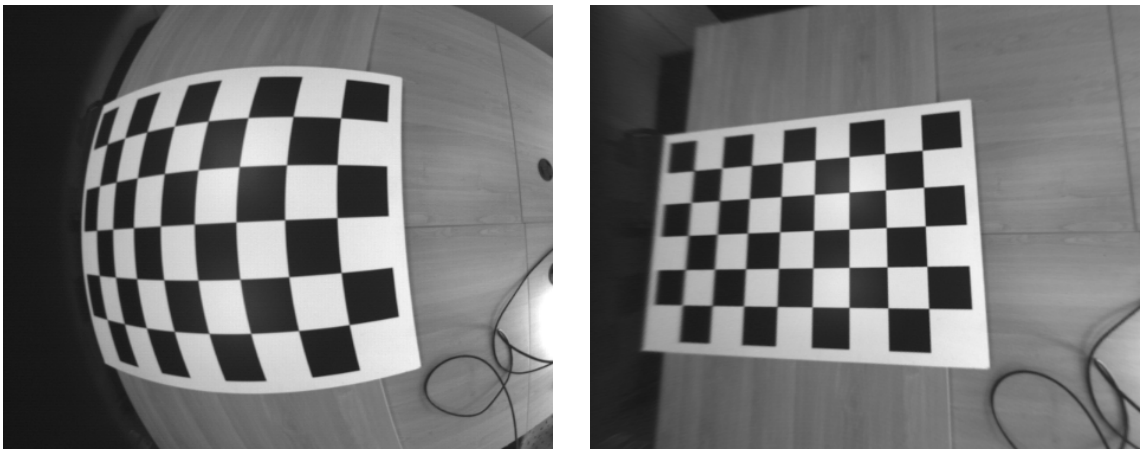


Figure 3.21: Undistortion using the calibrated model. The virtual camera is rotated by about 60° . The model works well even on the very border of projection.

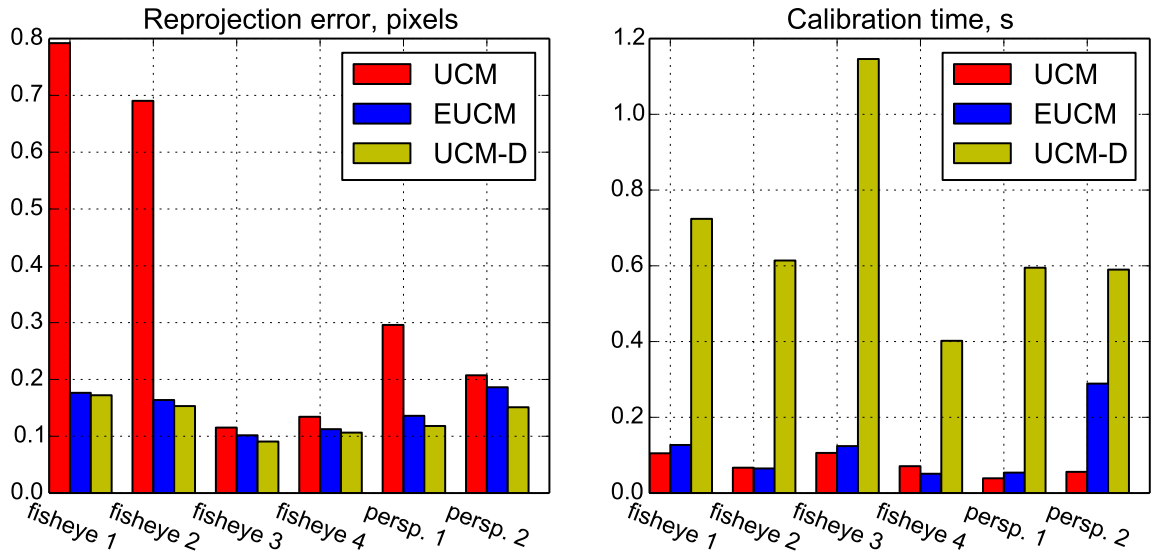


Figure 3.22: Monocular calibration results — model comparison.

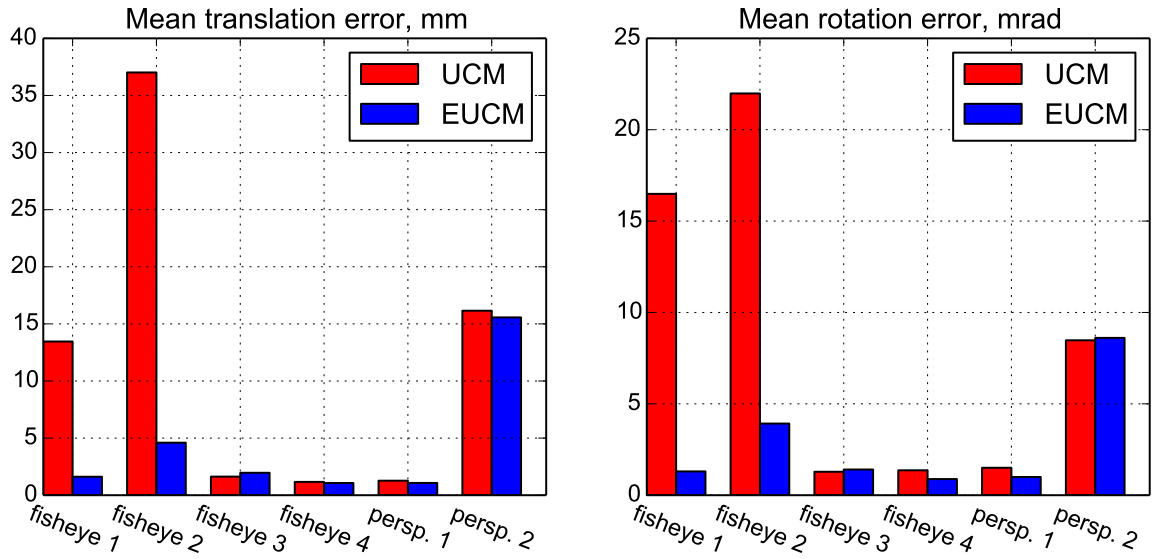


Figure 3.23: Monocular calibration results — pose reconstruction comparison.The reference model is the Unified Camera Model with Distortion (10 parameters).

Since there is a visibility constraint on stereo images (that is, the board should be completely visible in both images) it is better to acquire two additional monocular datasets, which are coupled with the stereo calibration only via the cameras' intrinsics. We can represent this scheme via the diagram from Fig. 3.24.

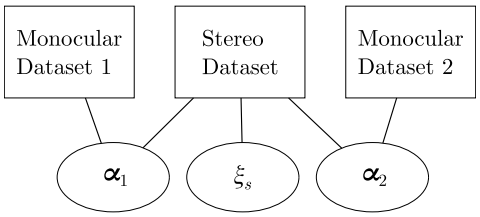


Figure 3.24: Stereo calibration coupling graph.$\boldsymbol{\alpha}_{1,2}$ are the intrinsic paramters; ξ_s is the stereo transformation.

For the particular case of stereo calibration, the following formal definition can be used. We have the following data and variables:

- Let $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathbb{R}^K$ be the intrinsic parameters for the 1st and 2nd cameras.
- Let $\Xi_1, \Xi_2 \subset \mathrm{SE}(3)$ be the sets of board positions for the first and the second camera's monocular calibration datasets.
- Let $\Xi_s \subset \mathrm{SE}(3)$ be the set of board positions in the first camera frame for the stereo calibration dataset.
- Let ξ_s define the 2nd camera frame in the 1st camera frame.
- Let $\boldsymbol{\mathcal{P}}_1, \boldsymbol{\mathcal{P}}_2$ be the detected board corner data for the monocular calibration.
- Let $\boldsymbol{\mathcal{P}}_{s,1}, \boldsymbol{\mathcal{P}}_{s,2}$ be the detected board corner data for the stereo calibration.

Let us define one more cost function for the stereo coupling:

$$E_s\left(\boldsymbol{\alpha}, \Xi, \xi_s, \mathcal{X}, \boldsymbol{\mathcal{P}}\right) = \sum_{\substack{\xi \in \Xi \\ \mathcal{P} \in \boldsymbol{\mathcal{P}}}} \sum_{\substack{\boldsymbol{p} \in \mathcal{P} \\ \boldsymbol{X} \in \mathcal{X}}} \left\| \boldsymbol{p} - \mathfrak{f}(\xi_s^{-1} \circ \xi(\boldsymbol{X}), \boldsymbol{\alpha}) \right\|^2 \tag{3.29}$$

Then the optimization problem will be defined as follows using both (3.3) and (3.29):

$$\begin{aligned} \{\boldsymbol{\alpha}_1^*, \boldsymbol{\alpha}_2^*, \Xi_1^*, \Xi_2^*, \Xi_s^*, \xi_s^*\} = \underset{\substack{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \\ \Xi_1, \Xi_2, \\ \Xi_s, \xi_s}}{\operatorname{argmin}} & E\left(\boldsymbol{\alpha}_1, \Xi_1, \mathcal{X}, \boldsymbol{\mathcal{P}}_1\right) + E\left(\boldsymbol{\alpha}_2, \Xi_2, \mathcal{X}, \boldsymbol{\mathcal{P}}_2\right) + \\ & E\left(\boldsymbol{\alpha}_1, \Xi_s, \mathcal{X}, \boldsymbol{\mathcal{P}}_{s,1}\right) + E_s\left(\boldsymbol{\alpha}_2, \Xi_s, \xi_s, \mathcal{X}, \boldsymbol{\mathcal{P}}_{s,2}\right) \end{aligned} \tag{3.30}$$

In fact this method can be extended to any number of cameras and transformations, under the condition that all the transformations can be actually identified. For example, instead of $\xi_s^{-1} \circ \xi$ in (3.29) it can be any other transformation combination. The calibration toolbox developed in this work can handle arbitrary transformation chains.

4.2.2 Results

A vertical stereo system has been calibrated using the described method. The first camera is at the bottom. The error distribution is shown in Fig. 3.25. The distribution standard deviation is given in Table 3.4.

A smaller reprojection error for stereo dataset can be explained by the fact that the board is generally farther away from the cameras to be completely visible, hence the distortion is smaller and it is easier to fit.

Such a small reprojection error for the stereo data means that the stereo calibration is actually precise, because both cameras use the same board transformation and, since they fit almost perfectly all the images from the dataset, this is the best extrinsic calibration quality assessment.

Table 3.4: Calibration results for a stereo system — standard deviation of the error distribution.

Camera/Dataset	Num. images	Num. detections	σ_x, px	σ_y, px
bottom/mono	284	246	0.16	0.14
top/mono	207	179	0.20	0.19
bottom/stereo	319	314	0.083	0.086
top/stereo	319	313	0.093	0.088

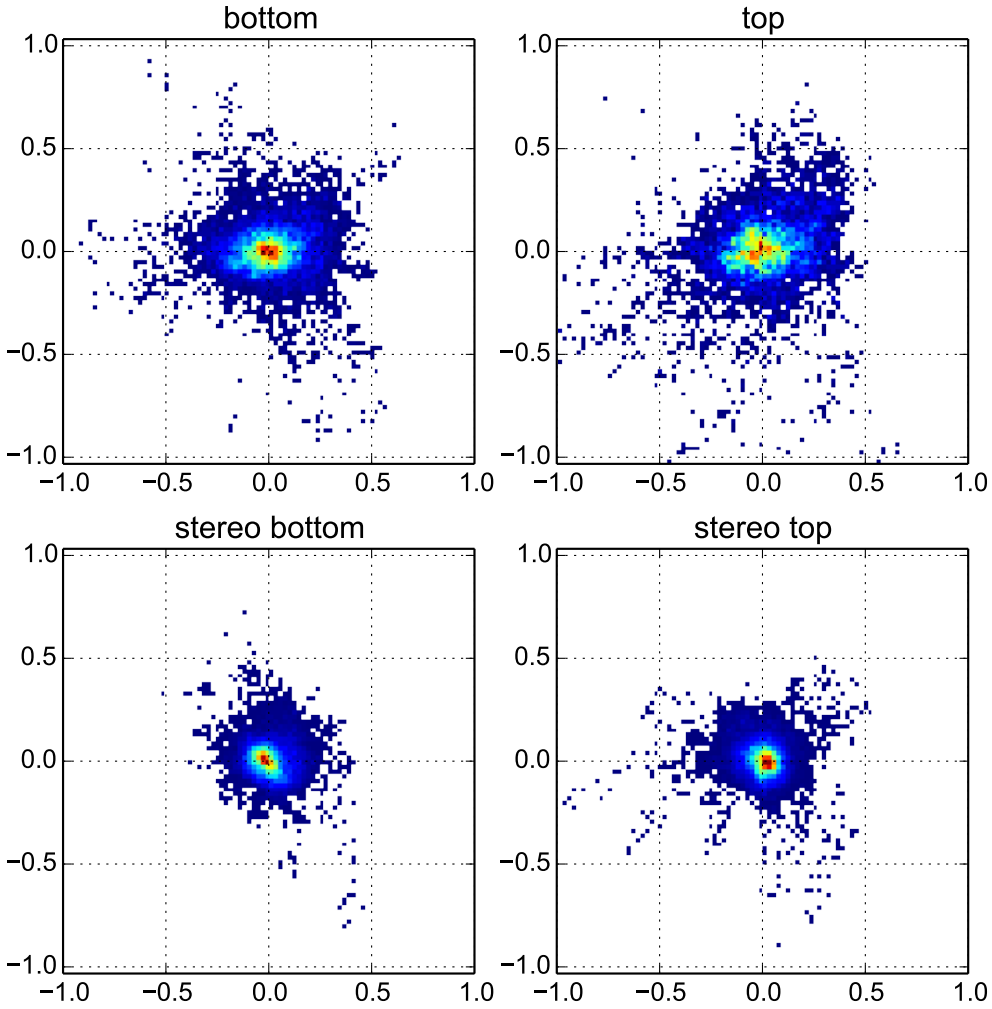


Figure 3.25: Error distribution for calibration result of a vertical stereo system.

5 Odometry Extrinsic Calibration

The problem of extrinsic calibration has been addressed in multiple papers; first for manipulators equipped with eye-in-hand cameras (for example, Daniilidis (1998)), and more recently for mobile robots by Antonelli *et al.* (2010); Censi *et al.* (2013); Heng *et al.* (2013). Censi *et al.* (2013) mentions an impact of the trajectories on the calibration quality, and a method to compute optimal trajectories for solving the linear part of the calibration, the wheel odometry intrinsic calibration, is proposed. Concerning the nonlinear part (which includes the camera extrinsic parameters) the authors give some intuitive guidelines. In particular, a straight trajectory does not give any information on the relative position of the camera with respect to the base, whereas a pure rotation makes the orientation of the sensor not fully observable.

We suggest a method to improve the calibration quality and its robustness with respect to the noise by computing the optimal trajectories. They provide the best possible definition of the calibration problem, which is formulated, in most cases, as a nonlinear optimization problem. The idea of generating exciting trajectories comes from the robotic arm community. It has been successfully applied to perform dynamic identification Armstrong (1989); Gautier and Khalil (1991). In Renaud *et al.* (2003), trajectory optimization for the extrinsic calibration of an eye-in-hand camera-manipulator system has been done. We present a framework which allows us to evaluate the quality of a given trajectory or of a set of trajectories for the extrinsic calibration. In this work the odometry is supposed to be calibrated and its output to be the integrated trajectory with the uncertainty.

The demonstration goes as follows. First, we formulate the calibration problem in the case of a mobile robot equipped with a single camera, without addressing trajectory properties. Second, we discuss how to choose trajectories to make the problem better defined. In order to do that, a criterion is proposed. Finally, we show the impact of the trajectory choice on the calibration quality.

5.1 Odometry Calibration Formulation

The extrinsic transformation between the odometry origin and the camera frame can be found as follows:

1. Collect the data, that is odometry measurements and calibration board images for, generally speaking, several trajectories.

2. Fit all the unknowns which are the extrinsic camera parameters, the board position with respect to the world origin for each trajectory, and the exact trajectories.

Here we assume that the camera is calibrated and intrinsic parameters $\boldsymbol{\alpha}$ are known and fixed. It is possible to integrate camera intrinsic calibration into this process, but we don't do it here for the sake of simplicity. Since $\boldsymbol{\alpha}$ is not involved in the demonstrations, we omit it everywhere. As in the case of stereo calibration we have to redefine the cost functions here.

- Let O be the world frame origin.
- Let O_{o} be the odometry frame origin.
- Let $\Xi_k \subset \mathrm{SE}(3)$ be the k-th set of robot base poses in O. Generally there are more than one of them.
- Let $\xi_i \in \Xi$ be the i-th base pose in O.
- Let $\xi_{\mathrm{b},k} \in \mathrm{SE}(3)$ define the board frame $O_{\mathrm{b},k}$ in O. Each $\xi_{\mathrm{b},k}$ is associated with Ξ_k.

- Let $\xi_{\mathrm{c}} \in \mathrm{SE}(3)$ define the camera frame O_{c} in O_{o}.
- Let $\zeta \in \mathrm{SE}(3)$ be a wheel odometry increment.
- Let $Z_k = \{\zeta_1..\zeta_{N_k}\} \subset \mathrm{SE}(3)$ be the set of wheel odometry increments.
- Let $\boldsymbol{\mathcal{C}}_{\zeta,k} = \{\mathcal{C}_{\zeta,1}..\mathcal{C}_{\zeta,N_k}\} \subset \mathbb{R}^{6\times 6}$ be a set of covariance matrices, defining the uncertainties of Z_k.

The reprojection cost function is defined as follows:

$$E_{\mathrm{b}}\left(\Xi, \xi_{\mathrm{c}}, \xi_{\mathrm{b}}, \mathcal{X}, \boldsymbol{\mathcal{P}}\right) = \sum_{\substack{\xi\in\Xi \\ \mathcal{P}\in\boldsymbol{\mathcal{P}}}} \sum_{\substack{\boldsymbol{p}\in\mathcal{P} \\ \boldsymbol{X}\in\mathcal{X}}} \left\| \boldsymbol{p} - \mathfrak{f}(\xi_{\mathrm{c}}^{-1} \circ \xi \circ \xi_{\mathrm{b}}(\boldsymbol{X})) \right\|^2 \tag{3.31}$$

This cost function is similar to (3.3) and (3.29), except that in this case the transformation chain is $\xi_{\mathrm{c}}^{-1} \circ \xi \circ \xi_{\mathrm{b}}$

In the case of odometry prior, we have to treat robot poses by pairs $\xi_i, \xi i + 1$. To do so we use the following notation:

$$\sum_{a_i, a_{i+1}\in A} \phi a_i, a_{i+1} \triangleq \sum_{i=1}^{\|A\|} \phi a_i, a_{i+1} \tag{3.32}$$

Where ϕ is a function of two elements from A. Odometry prior cost function is defined in the following way:

$$E_{\mathrm{o}}\left(\Xi, Z, \boldsymbol{\mathcal{C}}_{\zeta}\right) = \sum_{\substack{\xi_i, \xi_{i+1}\in\Xi \\ \zeta_i\in Z \\ \mathcal{C}_{\zeta,i}\in\boldsymbol{\mathcal{C}}_{\zeta}}} \left\| \zeta_i^{-1} \circ \xi_i^{-1} \circ \xi_{i+1} \right\|^2_{\mathcal{C}_{\zeta,i}} \tag{3.33}$$

It penalizes the difference between the odometry measurement and the computed trajectory increment. In other words, it tends to close the loop $\zeta_i^{-1} \circ \xi_i^{-1} \circ \xi_{i+1}$ (Fig. 3.26). The "flexibility" of the trajectory depends on the odometry prior precision defined by $\boldsymbol{\mathcal{C}}_{\zeta}$.

The setup is illustrated in Fig. 3.26. Let us use k for indexing datasets, corresponding to different trajectories. The problem is formulated as follows:

$$\left\{\xi_{\mathrm{c}}^*, \{\xi_{\mathrm{b},k}^*\}, \{\Xi_k^*\}\right\} = \underset{\xi_{\mathrm{c}}, \{\xi_{\mathrm{b},k}\}, \{\Xi_k\}}{\operatorname{argmin}} \sum_k E_{\mathrm{b}}\left(\Xi_k, \xi_{\mathrm{c}}, \xi_{\mathrm{b},k}, \mathcal{X}, \boldsymbol{\mathcal{P}}_k\right) + E_{\mathrm{o}}\left(\Xi_k, Z_k, \boldsymbol{\mathcal{C}}_{\zeta,k}\right) \tag{3.34}$$

5.2 Optimal Trajectory for the Extrinsic Calibration

To assess how well the optimization problem defined in (3.34) is determined, we reformulate the problem.

- Let $\hat{\xi}_{\mathrm{c}} \in \mathrm{SE}(3)$ be the base-camera transformation prior.
- Let $\mathcal{C}_{\hat{\xi}_{\mathrm{c}}} \in \mathbb{R}^{6\times 6}$ be the covariance of $\hat{\xi}_{\mathrm{c}}$.
- Let $\Xi_{\mathrm{o}} \subset \mathrm{SE}(3)$ represent odometry base poses obtained by odometry integration.
- Let $\boldsymbol{\mathcal{C}}_{\mathrm{o}} \subset \mathbb{R}^{6\times 6}$ be the set of covariance matrices which define uncertainties of camera position, induced by errors in Ξ_{o}.
- Let $\Xi_{\mathrm{v}} \subset \mathrm{SE}(3)$ be the set of camera positions computed via visual localization.
- Let $\boldsymbol{\mathcal{C}}_{\mathrm{v}} \subset \mathbb{R}^{6\times 6}$ be the set of covariance matrices which define uncertainties of Ξ_{v}.

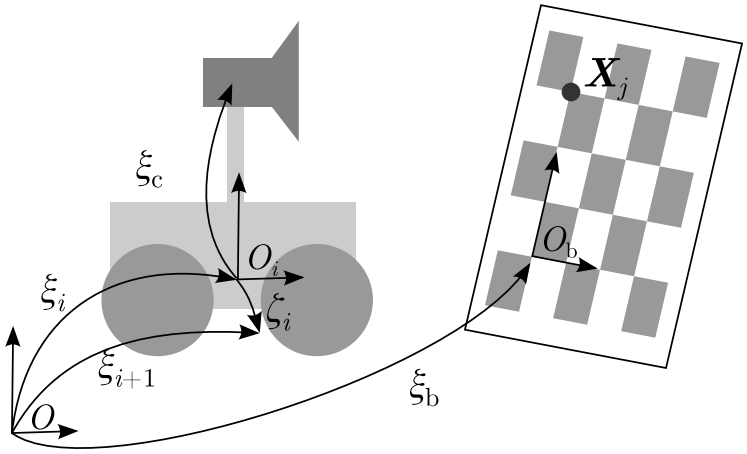


Figure 3.26: A scheme of the calibration process. The unknowns are $\{\xi_i\}$, ξ_{c}, ξ_{b}.

Instead of considering the coupled solution of the visual localization and odometry integration, which can be viewed as a SLAM problem, we say that the integrated odometry Ξ_{o} and the visual localization Ξ_{v} are given, as well as their covariance matrices. It reduces the number of variables to ξ_{c}, but the uncertainty introduced by the calculation of the other variables is taken into account. The reformulated problem is as follows:

$$\xi_{\mathrm{c}}^* = \operatorname*{argmin}_{\xi_{\mathrm{c}}} E^*(\xi_{\mathrm{c}}) \tag{3.35}$$

$$E^*(\xi_{\mathrm{c}}) = \left\| \hat{\xi}_{\mathrm{c}}^{-1} \circ \xi_{\mathrm{c}} \right\|^2_{\mathcal{C}_{\hat{\xi}_{\mathrm{c}}}} + \sum_{\substack{\xi_{\mathrm{o}} \in \Xi_{\mathrm{o}} \\ \xi_{\mathrm{v}} \in \Xi_{\mathrm{v}}}} \sum_{\substack{\mathcal{C}_{\mathrm{o}} \in \boldsymbol{\mathcal{C}}_{\mathrm{o}} \\ \mathcal{C}_{\mathrm{v}} \in \boldsymbol{\mathcal{C}}_{\mathrm{v}}}} \left\| \xi_{\mathrm{c}}^{-1} \circ \xi_{\mathrm{o}}^{-1} \circ \xi_{\mathrm{c}} \circ \xi_{\mathrm{v}} \right\|_{\mathcal{C}_{\mathrm{v}} + \mathcal{C}_{\mathrm{o}}} \tag{3.36}$$

The second term means that, as shown in Fig. 3.27, we want:

$$\xi_{\mathrm{o}} \circ \xi_{\mathrm{c}} = \xi_{\mathrm{c}} \circ \xi_{\mathrm{v}} \tag{3.37}$$

We want to analyze the properties of the energy function E^* from (3.36) in the neighborhood of the solution. To improve the problem stability properties we can minimize the following criterion:

$$F = -\sum_{i=1}^{6} \log \sigma_i \left(\frac{\partial^2}{\partial \xi_{\mathrm{c}}^2} E \right) \tag{3.38}$$

where $\sigma_i(\cdot)$ is the i^{th} singular value of a matrix. Basically, it is equivalent to the maximization of the Hessian matrix determinant $\det(H) = \prod_{i=1}^{6} \sigma_i(H)$. In Renaud *et al.* (2003) they suggest the matrix condition number as a criterion instead of the determinant, in Armstrong (1989) the inverse the minimum singular values is proposed. But in our case one degree of freedom is not observable, hence the corresponding singular value is defined only via the prior transformation estimation and does not depend on the trajectory. In fact, the condition number will be just proportional to the maximum singular value. Yet, there might be a better criterion than (3.38), like a different function of the singular values; it is still an open question.

The Hessian matrix itself $\frac{\partial^2}{\partial \xi_{\mathrm{c}}^2} E$ can be used to analyze the observability of different degrees of freedom. Its singular decomposition and singular vectors corresponding to small or zero singular values tell us which degrees of freedom of the extrinsic transformation cannot be observed.

In our case (3 DoF mobile robot) the only rotation is about z-axis, hence z-component of the translation of ξ_{c} is not observable (as it is mentioned in Heng *et al.* (2013)), and its value

is defined by the prior transformation estimation. But x, y, and $\boldsymbol{u}\vartheta$ are observable, and the larger the singular values, the better the final numerical estimation.

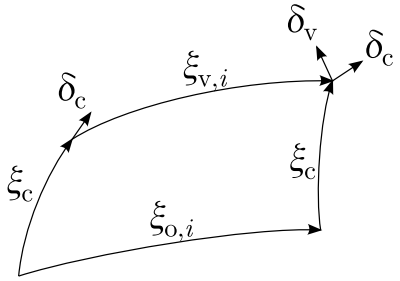


Figure 3.27: δ_{c} is an infinitesimal increment of ξ_{c}, notice that δ_{c} appears twice since ξ_{c} is present twice in the transformation chain; δ_{v} is increment δ_{c} transformed into the terminal frame; $\delta = \delta_{\mathrm{v}} - \delta_{\mathrm{c}}$ would be the residual which appears when we modify the solution ξ_{c} by δ_{c}.

If covariances are defined for the infinitesimal motions of the terminal frame, then we need to transform the infinitesimal increments δ_{c} of ξ_{c} into the terminal frame (Fig. 3.27):

$$\delta_{\mathrm{v}} = {}^{\mathrm{v},i}\boldsymbol{T}_{\mathrm{v},1}\delta_{\mathrm{c}} \tag{3.39}$$

$${}^{\mathrm{v},i}\boldsymbol{T}_{\mathrm{v},1} = \begin{pmatrix} R_{\mathrm{v},i}^{-1} & -R_{\mathrm{v},i}^{-1}[\boldsymbol{t}_{\mathrm{v},i}]_{\times} \\ 0 & R_{\mathrm{v},i}^{-1} \end{pmatrix} \tag{3.40}$$

where ${}^{i}\boldsymbol{T}_{j}$ is a twist transformation matrix, $R_{\mathrm{v},i}$ and $\boldsymbol{t}_{\mathrm{v},i}$ is another representation of $\xi_{\mathrm{v},i}$. Another way of writing the residual in energy expression (3.36) is:

$$\delta = \delta_{\mathrm{v}} - \delta_{\mathrm{c}} \approx \xi_{\mathrm{c}}^{-1} \circ \xi_{\mathrm{o},i}^{-1} \circ \xi_{\mathrm{c}} \circ \xi_{\mathrm{v},i} \tag{3.41}$$

5.2.1 Covariance for the Visual Localization

Here we assume that the visual localization is done using a calibration board. Any other visual localization can be used to compute the trajectory of the camera, but similar precision analysis must be done in this case. The covariance of the visual localization, assuming the camera intrinsic parameters exactly known, comes from the uncertainty in the corner detections of the calibration board. The localization is computed as non-linear least squares. The cost function is defined similar to (3.3):

$$E_{\mathrm{v}}(\xi) = \sum_{\substack{\boldsymbol{p}\in\mathcal{P} \\ \boldsymbol{X}\in\mathcal{X}}} \|\boldsymbol{p} - \mathfrak{f}(\xi(\boldsymbol{X}))\|_{\mathcal{C}_{\boldsymbol{p}}}^{2} \tag{3.42}$$

Here $\mathcal{C}_{\boldsymbol{p}}$ is a 2×2 corner detection covariance matrix. We assume that u and v of detected feature points are decorrelated and have equal uncertainties:

$$\mathcal{C}_{\boldsymbol{p}} = \begin{pmatrix} \sigma_{\boldsymbol{p}}^{2} & 0 \\ 0 & \sigma_{\boldsymbol{p}}^{2} \end{pmatrix} = \sigma_{\boldsymbol{p}}^{2} I \tag{3.43}$$

For practical purposes we can assume that $\sigma_{\boldsymbol{p}} = 1$, unless we have a better prior estimation of corner detector precision.

The whole least-squares problem for all points together is defined as follows:

$$J^{T}\mathcal{C}_{\Delta\boldsymbol{p}}^{-1}\Delta\boldsymbol{p} = 0 \tag{3.44}$$

where J is a concatenation of Jacobian matrices for individual points:

$$J = \begin{pmatrix} \dfrac{\partial \mathfrak{f}(\xi(\boldsymbol{X}_1))}{\partial \xi} \\ \vdots \\ \dfrac{\partial \mathfrak{f}(\xi(\boldsymbol{X}_M))}{\partial \xi} \end{pmatrix} = \begin{pmatrix} J_1 \\ \vdots \\ J_M \end{pmatrix} \tag{3.45}$$

$\Delta \boldsymbol{p}$ is the concatenation of reprojection errors:

$$\Delta \boldsymbol{p} = \begin{pmatrix} \boldsymbol{p}_1 - \mathfrak{f}(\boldsymbol{X}_1, \xi) \\ \vdots \\ \boldsymbol{p}_M - \mathfrak{f}(\boldsymbol{X}_M, \xi) \end{pmatrix} \tag{3.46}$$

$\mathcal{C}_{\Delta \boldsymbol{p}}^{-1}$ is the covariance matrix of $\Delta \boldsymbol{p}$:

$$\mathcal{C}_{\Delta \boldsymbol{p}}^{-1} = \begin{pmatrix} \mathcal{C}_{\boldsymbol{p}}^{-1} & 0 & \dots & 0 \\ 0 & \mathcal{C}_{\boldsymbol{p}}^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathcal{C}_{\boldsymbol{p}}^{-1} \end{pmatrix} \tag{3.47}$$

Let us consider that the solution for (3.44) is found by using the iterative Newton-Gauss optimization. The solution looks as follows (for the last iteration, Lourakis and Argyros (2004)):

$$\Delta \xi = \underbrace{(J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J)^{-1} J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1}}_{J_{\mathcal{C}}^{+}} \Delta \boldsymbol{p} \tag{3.48}$$

Here, $J_{\mathcal{C}}^{+}$ is the weighted generalized inverse of J. We know that if a linear mapping is applied to a random Gaussian vector:

$$\boldsymbol{y} = A\boldsymbol{x} \tag{3.49}$$

Then the covariance of the result is computed as:

$$\mathcal{C}_{\boldsymbol{y}} = A \mathcal{C}_{\boldsymbol{x}} A^T \tag{3.50}$$

In our case:

$$\begin{aligned} \mathcal{C}_{\mathrm{v}} &= J_{\mathcal{C}}^{+} \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J_{\mathcal{C}}^{+T} \\ &= (J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J)^{-1} J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} \mathcal{C}_{\Delta \boldsymbol{p}} \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J (J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J)^{-1} \\ &= (J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J)^{-1} J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J (J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J)^{-1} \\ &= (J^T \mathcal{C}_{\Delta \boldsymbol{p}}^{-1} J)^{-1} \end{aligned} \tag{3.51}$$

This general form, which is an interesting theoretical result by itself, can be simplified even further, if we recall that $\mathcal{C}_{\Delta \boldsymbol{p}}^{-1} = \sigma_{\boldsymbol{p}}^{-2} I$:

$$\mathcal{C}_{\mathrm{v}} = \sigma_{\boldsymbol{p}}^2 (J^T J)^{-1} \tag{3.52}$$

5.2.2 Odometry Covariance Matrix

We are interested in $\mathcal{C}_{\zeta,i}$, the covariance which characterizes the error distribution of the odometry increment measurement in its local frame. It can be computed as follows:

$$\mathcal{C}_{\zeta,i} = \frac{\partial \zeta_{i,\text{local}}}{\partial \zeta_i} \frac{\partial \zeta_i}{\partial \boldsymbol{u}} \mathcal{C}_{\boldsymbol{u},i} \left(\frac{\partial \zeta_{i,\text{local}}}{\partial \zeta_i} \frac{\partial \zeta_i}{\partial \boldsymbol{u}} \right)^T \tag{3.53}$$

Its expression has been obtained in Chapter 5 and is given in (5.33).

We need to integrate the covariance matrix along the trajectory to get its actual value at a given point:

$${}^{\text{o},i}\mathcal{C}_{\text{o},i} = \mathcal{C}_{\zeta,i} + {}^{i}\boldsymbol{T}_{i-1}\, {}^{\text{o},i-1}\mathcal{C}_{\text{o},i-1}\, {}^{i}\boldsymbol{T}_{i-1}^{T} \tag{3.54}$$

$\boldsymbol{T}$ is a twist transformation matrix, like in (3.40). The given expression gives a 3×3 matrix; three more rows and columns are to be added to make it 6×6. We also add a small constant covariance matrix $\tilde{\mathcal{C}}$ to it, to introduce some uncertainty in the other 3 directions, then the frame must be changed from i-th odometry frame to i-th camera frame:

$$\mathcal{C}_{\text{o},i} = {}^{\text{v},i}\boldsymbol{T}_{\text{o},i} ({}^{\text{o},i}\mathcal{C}_{\text{o},i} + \tilde{\mathcal{C}}) {}^{\text{v},i}\boldsymbol{T}_{\text{o},i}^{T} \tag{3.55}$$

5.3 Practical constraints and details

As it is mentioned in Censi *et al.* (2013), in any experimental setup there are some practical constraints which are not necessarily represented by the cost function (3.38). It is possible to augment it by adding terms to find the best of practical and feasible trajectories.

Visibility Constraint To make sure that the images contain the calibration board along the trajectory, we reinforce the visibility constraint by adding a term to (3.38):

$$F = -\sum_{i=1}^{6} \log \sigma_i \left(\frac{\partial^2}{\partial \xi_{\text{c}}^2} E \right) + \sum_{\xi \in \Xi} \sum_{\boldsymbol{X} \in \mathcal{X}} \rho_r(\mathfrak{f}(\xi_{\text{c}}^{-1} \circ \xi^{-1} \circ \xi_{\text{b}}(\boldsymbol{X}))) \tag{3.56}$$

here $\rho_r : \mathbb{R}^2 \to \mathbb{R}$ is the penalty function, defined as follows:

$$\rho_r(\boldsymbol{p}) = \begin{cases} 0 \quad \text{if} \quad \|\boldsymbol{p} - \boldsymbol{p}_0\| < r \\ \lambda(\|\boldsymbol{p} - \boldsymbol{p}_0\| - r)^2 \text{ otherwise} \end{cases} \tag{3.57}$$

where $\boldsymbol{p}_0$ is the image center; r is a certain radius. We can divide the image into three zones, as shown in Fig. 3.28. All the feature points of the calibration board in the first zone are not penalized (and its radius is r), but once a point approaches zone 2, the additional cost starts growing quadratically, which means, considering that the data term is a logarithm of singular values, that all the points will rather stay in zone 1 after the optimization.

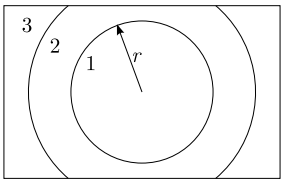


Figure 3.28: Every image is divided into three concentric zones: 1) the additional cost for the points there is 0; 2) the additional cost for points grows quadratically with respect to the distance from the projection center; 3) there are no points projected into this zone because of the fact that we are using fisheye cameras.

This method is adapted for fisheye optics. For pinhole cameras it may be interesting to use a different regularization term.

Constraints on the Trajectory Curvature For certain types of mobile robots (such as ones with steerable wheels) there are limits on the trajectory curvature. This constraint can be taken into account by adding the following term to the cost function:

$$E_{\text{curvature}} = \lambda_\kappa \sum_{i=1}^{N-1} \max\left(\frac{\varphi_i}{l_i} - \kappa_{\max}, 0\right)^2 \tag{3.58}$$

Where φ is the elementary rotation, l is the elementary translation length.

Trajectory Parametrization It appears that a trajectory which is a single arc of a constant curvature is not enough to do the extrinsic identification. It has a simple explanation: for such a trajectory the camera's trajectory will also be an arc of a different radius. If we rotate the camera's initial position and the calibration board about the center of the circular trajectory, then the camera's motion will be the same with respect to the calibration board. It means that such a trajectory will define a circle with the center at the trajectory's center, on which lies the camera. If we take two distinct arcs with different curvature as a trajectory, it will define two circles, and the camera has to be at one of the intersections, since two circles generate two intersections. If the camera's prior position is accurate enough, the optimization will converge to its real position, that is why the trajectory is represented by two arcs. It is parametrized by their initial point along x, initial orientation, linear and angular speed of the robot. It makes 8 optimization parameters in total. "The optimal trajectory" refers to the solution of the optimization problem with such a parametrization.

Optimization Initialization A good initialization is important to make the problem converge to the global optimum. In order to do that, the initialization is carried out as follows:

1. The prior on the camera extrinsic ξ_c must be provided by the user. Generally, its position can be measured with reasonable precision, the orientation prior can be approximate.
2. Transformations $\{\Xi_i\}$ are initialized using the odometry measurements, which is straightforward.
3. $\{\xi_{b,i}\}$ are initialized in a few steps:
   a) Using first image of a sequence, compute ${}^c\xi_b$ as it has been explained before.
   b) Compute $\hat{\xi}_b = \xi_1 \circ \xi_c \circ {}^c\xi_b$.
   c) Using all the images of the sequence, minimize the reprojection residual over ξ_b, using $\hat{\xi}_b$ as the initialization. Basically we minimize (3.34) fixing all the variables, except for ξ_b.

5.4 Results

The concept has been tested using synthetic data [1]. It allows us to compare the results to the true values. For a generated trajectory, a set of images of the calibration board has been generated, using the camera model (30 images per trajectory piece). These images have been used to perform the calibration. Even though the images are synthetic, the image processing part still introduces a certain error in the calibration due to the imperfection of the pattern detection. The extrinsic calibration has been done twice: once using the optimal trajectory, and one more time using a suboptimal one. The suboptimal trajectory has been obtained by stopping the optimization process before convergence, but after the visibility and curvature constraints have been satisfied. Both trajectories are given in Fig. C.5.

[1] Technical details of image rendering are given in Appendix B

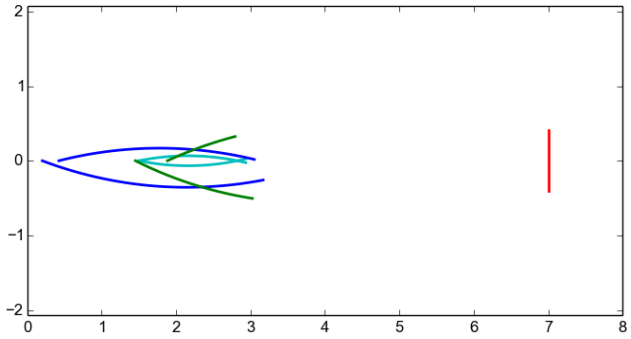


Figure 3.29: Generated trajectories: Cyan – the initial trajectory, $E = 688$, the curvature constraint is not satisfied; green – a suboptimal trajectory, $E = -29.6$; Blue – the optimal trajectory, $E = -37.4$. E is the cost function.

To check the extrinsic calibration robustness, some noise has been added to the odometry measurements. The absolute value of the noise and the resulting calibration error are summarized in Table 3.5. The results are also visualized in Fig. C.6. The true values of the extrinsic parameters are $\xi_{\mathrm{c}} = [3.5, 0.35, 0, -1.2092, 1.2092, -1.2092]$. The errors are computed in the following way:

$$\begin{aligned} \delta &= \xi_{\mathrm{c}}^{-1} \circ \hat{\xi}_{\mathrm{c}} \\ e_{\boldsymbol{t}} &= \|\boldsymbol{t}_{\delta}\| \\ e_{\vartheta} &= \vartheta_{\delta} \end{aligned} \tag{3.59}$$

where $\hat{\xi}_{\mathrm{c}}$ is the estimated extrinsic parameters; $\boldsymbol{t}_{\delta}$ is their translational part; ϑ is the norm of the rotation vector of δ.

The results show that the optimal trajectory is significantly more resistant to noise in the odometry measurements. The noise in the test has zero mean, and in our strong belief odometry bias would have a negative effect on this calibration. It must be eliminated during the preceding odometry calibration.

We can perform simultaneous calibration of extrinsic parameters for a stereo system. For that we define the transformation between the board and the cameras as follows:

$$\begin{aligned} {}^{\mathrm{c}_1}\xi_{\mathrm{b}} &= (\xi_{\mathrm{o},i} \circ \xi_{\mathrm{c}})^{-1} \circ \xi_{\mathrm{b}} \\ {}^{\mathrm{c}_2}\xi_{\mathrm{b}} &= (\xi_{\mathrm{o},i} \circ \xi_{\mathrm{c}} \circ {}^{\mathrm{c}_1}\xi_{\mathrm{c}_2})^{-1} \circ \xi_{\mathrm{b}} \end{aligned} \tag{3.60}$$

The extrinsic parameters for the first camera are the same. For the second:

$$\xi_{\mathrm{c}_2} = [3.5, -0.35, 0, -1.2092, 1.2092, -1.2092]$$

Since extrinsic transformations for the two cameras are different, different trajectories have to be used for the optimal calibration. The trajectories are represented in Fig. 3.31. In the optimization problem we have the following cost function blocks:

1. The first camera optimal trajectory — 60 images.
2. The second camera optimal trajectory — 60 images.
3. The stereo calibration — two times 27 images.

The results are presented in Table 3.6. For stronger noise, the stereo version gives a slightly better orientation estimation, but generally speaking, the order of precision is the same as for the monocular calibration. It should be noted that we generated the calibration trajectories regardless of the fact that the two cameras' extrinsics are coupled via the stereo calibration.

Table 3.5: Comparison between extrinsic calibration quality for two trajectories. The numbers in the last four columns represent the error in the extrinsic transformation estimation.

Odometry noise, absolute value		Optimal trajectory		Suboptimal trajectory	
$\boldsymbol{t}$, m	$\boldsymbol{u}\vartheta$, rad	$\boldsymbol{e_t}$, m	$\boldsymbol{e}_\vartheta$, rad	$\boldsymbol{e_t}$, m	$\boldsymbol{e}_\vartheta$, rad
0	0	0.00935	0.000665	0.0166	0.00181
0.002	0.01	0.0101	0.00302	0.0431	0.0143
0.004	0.02	0.0164	0.00727	0.199	0.0507
0.006	0.03	0.0628	0.0149	0.436	0.105
0.008	0.04	0.13	0.0238	0.739	0.17

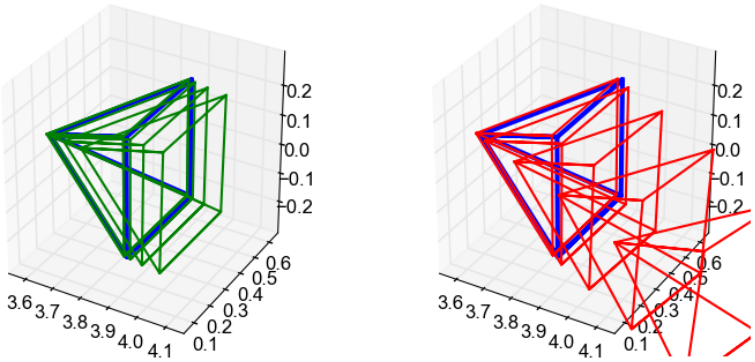


Figure 3.30: Visualization of the calibration results as a function of the odometry noise. Blue – ground truth, green – optimal trajectory, red – suboptimal trajectory. The camera wire model is a pyramid $0.4 \times 0.4 \times 0.4$m. The stronger the noise, the farther away the estimated extrinsics from the real values. For the optimal trajectory (on the left) the sensitivity to the noise is significantly smaller than for the suboptimal one.

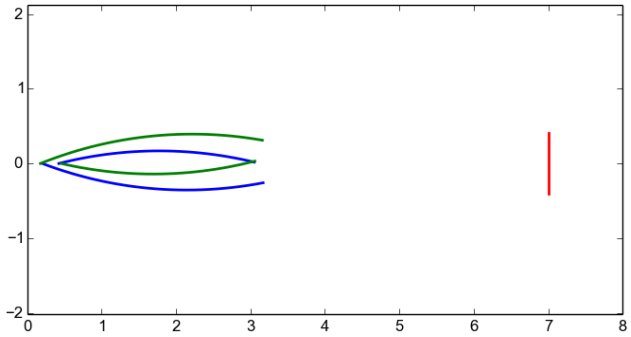


Figure 3.31: Generated trajectories for the stereo extrinsic calibration. Blue – the optimal trajectory for the left camera, green – the optimal trajectory for the right camera, red – the calibration board position. Considering the symmetry of the stereo system, it is logical that the two trajectory sets are symmetrical.

Table 3.6: Comparison between extrinsic calibration quality for monocular and stereo systems. The numbers in the last four columns represent the error in the extrinsic transformation estimation. The transformation is the same in both cases.

Odometry noise, absolute value		Monocular		Stereo	
$\boldsymbol{t}$, m	$\boldsymbol{u}\vartheta$, rad	$\boldsymbol{e_t}$, m	$\boldsymbol{e}_\vartheta$, rad	$\boldsymbol{e_t}$, m	$\boldsymbol{e}_\vartheta$, rad
0	0	0.00935	0.000665	0.0172	0.00123
0.002	0.01	0.0101	0.00302	0.0102	0.00331
0.004	0.02	0.0164	0.00727	0.0173	0.00506
0.006	0.03	0.0628	0.0149	0.0565	0.00804
0.008	0.04	0.13	0.0238	0.116	0.0131

Table 3.7: Calibration results for the same system for two datasets — extrinsic parameters of the camera (transformation ξ_c).

Dataset	$\boldsymbol{t}_x$, m	$\boldsymbol{t}_y$, m	$\boldsymbol{t}_z$, m	$\boldsymbol{r}_x$, rad	$\boldsymbol{r}_y$, rad	$\boldsymbol{r}_z$, rad
1	0.391324	1.57821	1.16095	-1.6013	-0.003233	0.017913
2	0.388454	1.59595	1.16045	-1.6035	-0.004753	0.018170

Real Data Test This method has been successfully used to calibrate a real car equipped with cameras. The calibration has been done twice using two similar, but distinct, datasets. The reprojection error distributions are shown in Fig. 3.32–3.33. The calibrated extrinsic parameters are given in Table 3.7. The results of both calibrations are quite close, which means that the experiment is repeatable. A good way to validate the calibration would be to use the system for visual odometry.

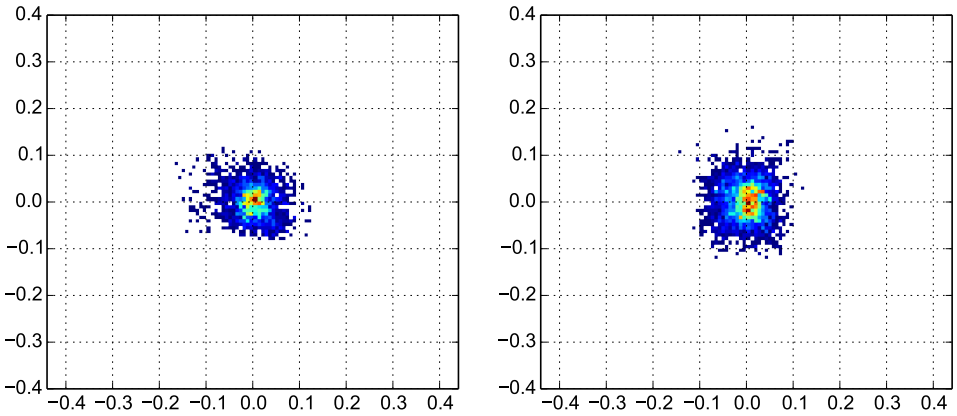


Figure 3.32: Error distribution for the bottom and the top camera — dataset 1.

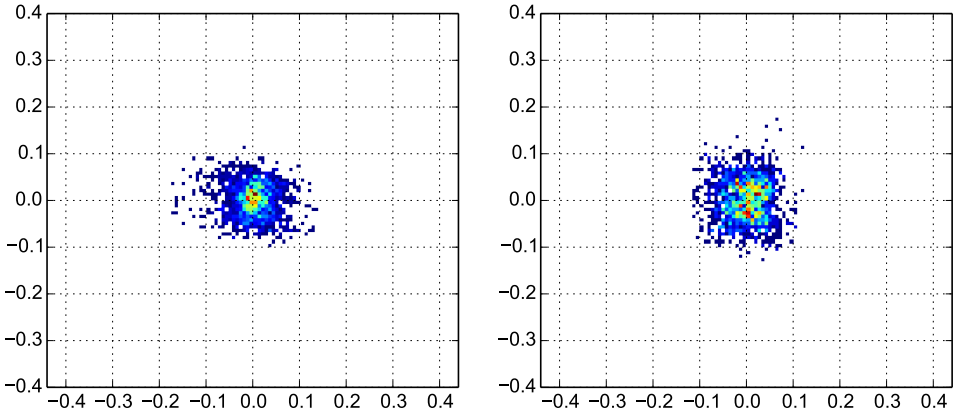


Figure 3.33: Error distribution for the bottom and the top camera — dataset 2.

6 Conclusions

An efficient and flexible calibration toolbox for multi-camera systems has been developed. It has a novel fully-automated subpixel detector of calibration boards, which is significantly faster than that of OpenCV. It allows us to use large datasets for calibration purposes while keeping the computation time reasonably short.

The Enhanced Unified Camera Model has been intensively tested. The model's precision, evaluated using the reprojection error after the calibration, is close to the one provided by the model described in Mei and Rives (2007), The time needed for solving the calibration problem for the Enhanced Unified Model is close to the one for classical Spherical Model. Moreover, only two distortion parameters eliminate any possibility of overfit, which makes the model suitable for both high- and low-distortion lenses.

A method of calculating optimal trajectories for extrinsic calibration of mobile robots has been proposed and tested. Its effectiveness to reduce the noise impact has been demonstrated on simulated data. It has been applied to calibrate a complete system Camera-Odometry, which has been used for localization experiments, described in Chapter 5.

Intermediate monocular calibration results have been published in Khomutenko *et al.* (2016a). Extrinsic calibration technique has been published in Khomutenko *et al.* (2017). Below we summarize the main contributions and novelties described in this chapter and propose some ideas on how to continue and extend the present work.

6.1 Contributions

Board Detector As it has been stated, a reliable and robust board detector is one of the most important calibration tools. Any limitations imposed by it, like necessity to do interactive detection with the user involved in the process, or inability to detect the board in certain conditions for high contrast background and high distortion, seriously complicates the calibration process and makes it more time-consuming.

The developed board detector has proven to be robust and to efficiently detect the board in virtually all images where a human could do it. It works both for images where the board is close to the camera and strongly distorted, and for images where the board is quite far away and its squares have apparent size of 6-10 px. This detector has been compared to the one from OpenCV library and demonstrates both better detection rate (that is, number of images where the calibration board has been detected) and accuracy.

Subpixel Refinement Since the corner detector is a part of the system to be calibrated, we want its impact be as small as possible, even negligible. It is almost the case with the proposed subpixel detector, which uses the most valuable gradient information along calibration pattern edges to estimate the corner position as accurately as possible. The detector takes into account the overexposure effect, which keeps the corner measurements precise even if the lighting conditions are not perfect.

Generic Calibration Framework The calibration toolbox developed during this work is a flexible tool which allows us to calibrate different assemblies involving cameras and wheel odometry. The whole calibration problem and calibration data are defined in a separate .json file. Different types of cameras, as well as arbitrary transformation chains are allowed. The source code of the project can be found at `https://github.com/BKhomutenko/visgeom`.

Optimal Trajectories for Camera-Odometry Calibration A planned data acquisition improves the calibration precision. The developed methodology lets us evaluate a trajectory's quality for the extrinsic calibration, taking into account the odometry noise properties and image-based localization uncertainty. The global optimization problem allows us to include additional constraints, such as space limitations, size of the calibration board in the image and so on. The synthetic data have allowed us to test the concept in a fully controlled environment. Calibration using real data has been done and has given consistent repeatable results. Also, the computed extrinsic parameters are used in visual localization applications described in Chapter 5.

Covariance of a Least-Squares Solution Another small and yet interesting theoretical result is (3.51). It is the expression for covariance matrix of an output of a weighted least-square problem.

6.2 Future Work

Extrinsic Calibration Validation Accurate extrinsic calibration validation is not a trivial problem since it cannot be verified by directly re-using a calibration dataset for validation, since the trajectory of the robot is not precisely known and the odometry is noisy and biased. Moreover, because of the suspension, the transformation between the odometry frame and the camera is not constant and changes while the car accelerates, brakes, turns.

One way to validate the parameter values is to use them for visual mapping and localization. If such a system outputs precise localization, then the calibration is rather accurate. On the other hand, poor localization may be related to the internal localization system problems.

Camera-Odometry-GPS-IMU Calibration An interesting extension of the calibration toolbox would be the complete sensor calibration for a mobile robot. It would include also GPS sensor position and IMU pose with respect to the odometry frame. Intrinsic odometry calibration also can be done.

Accurate Suspension Modeling To improve the calibration quality, the suspension can be modeled explicitly. The car body changes its orientation because of accelerations. The stiffness of this system can be estimated. And since IMUs give us accurate acceleration information, we can predict the instant car body position and not treat this transformation as noise.

Trajectory Planning for Localization The criterion on the trajectory quality for the extrinsic calibration can be applied for any kind of visual localization, whether they use the calibration board or not. Here we suggest a methodology to estimate the localization precision using the calibration board, but any visual odometry system can replace it. The only requirement is to be able to estimate the localization covariance matrix. For natural features, doing it precisely is a challenging problem. A possible research track would be to estimate the average visual localization precision statistically. Another option is to do it in two steps. First use any kind of trajectory to build a metric map, then use similar calculations as for the calibration board model to get the localization covariance estimation, based on the environmental structure. Yet, feature visibility has to be treated and analyzed. The method can also be used to analyze the observability properties of extrinsic parameters for a given trajectory.

Chapter 4

Direct Stereo Correspondence for Fisheye Cameras

1 Introduction

Human perception proves that two passive visual sensors are enough for a robust and accurate 3-dimensional reconstruction of the environment. But still, the stereo correspondence and vision-based 3D reconstruction are not considered as "solved" problems. This chapter is dedicated to an algorithm of direct fisheye stereo correspondence, as well as to a number of techniques which improve the stereo correspondence results in the context of fisheye distortion.

First, let's have a look at the concept of dense stereo correspondence. Reconstruction using two views is called *triangulation*. The intuitive understanding of the triangulation is the following: as a camera moves left, all the object in the image move right; the farther the spatial point from the camera, the smaller its apparent motion in the image. Hence, by measuring the apparent motion of the point we can deduce its distance to the camera. To be able to measure a point's motion and triangulate it we have to find it in both images. That is to say, to reconstruct a point in one image, we need to find the point, to which it corresponds in the other image. For both correspondence search and matching we need to know the geometric transformation between the two camera positions with a high precision, as well as the intrinsic camera parameters. In other words, the stereo system has to be calibrated. Throughout this chapter, we assume that it is the case.

The transformation is needed for the matching process, because it introduces geometric constraints reducing the search space a lot. That is, instead of looking for the correspondence across the whole image, we search a small area. More precisely, the search is to be done along epipolar lines (which become curves in the case of fisheye projection). Intuitively, the concept of epipolar lines can be explained as follows. If we take a sequence of points, aligned along a straight line which passes through the center of projection of a camera, and observe them with another camera, they will lie on a line, called epipolar, or on a curve in the case of cameras with strong distortion (Fig. 4.1). Since all the points are projected to the same 2D point on the first image, we know that the corresponding 2D point on the second image necessarily lies on this line. The way to compute the epipolar curve equations for the Enhanced Unified Model is given in Chapter 2.

Once correspondence between pixels from both images is established, the reconstruction is a relatively simple step, even though it might be quite technical. That is why *stereo correspondence* and *stereo reconstruction* are usually synonyms. Another terminological remark, in the context of perspective projection and pinhole camera, the output of 3D reconstruction is usually a depth value associated with each reconstructed pixel. By depth we mean its z coordinate, as it allows to reconstruct 3D coordinates from image coordinates. In the case

of fisheye camera, instead of z coordinate, the output is the distance between the projection origin and the 3D point, since for more than 180° field of view, z might be 0 all along the direction, associated with a pixel to be reconstructed. In Caruso *et al.* (2015), it is suggested that the term "depth map" should be replaced by "distance map". Throughout this text, we still use *depth* to refer, depending on the context, either to z or to the distance between the projection origin and the reconstruction point, since just *distance* might be ambiguous. Consequently, the stereo reconstruction algorithm's output is called *depth map*.

Generally, correspondence search, even boosted up by the epipolar constraints, is subject to mismatches. Independent pixel matching generally gives noisy results. There exist a number of different regularization techniques to reduce this effect of mismatching. For example, in Geiger *et al.* (2011) it is proposed to compute first so-called reliable matches, which are correct with a high probability, and then use these values to reconstruct the depth around. In this work we use an efficient depth regularization technique, called the Semi-Global Matching algorithm Hirschmuller (2005). Regularization consists in introducing some prior knowledge about the environment to improve the matching. The classical model for the stereo correspondence problem is that the depth is a piecewise continuous function, hence neighbor pixels are likely to have similar depth value.

1.1 Pixel Sweep vs Space Sweep

We can distinguish two main dense stereo correspondence paradigms : pixel-sweep and space-sweep. By pixel-sweep we mean that the method is looking for the correspondence by iterating over pixels, like classical rectified pinhole stereo, where the iteration is done along the image rows or columns Geiger *et al.* (2011); Hirschmuller (2005) or, for example, the semi-dense odometry Engel *et al.* (2013), where it is done along epipolar lines, which are still straight lines. Hereafter we will use the term *disparity* to refer to the displacement of the corresponding pixels along the epipolar line, measured in pixel steps.

A stereo correspondence algorithm, which is called *Plane Sweep* and represents the other paradigm, originally proposed in Collins (1996), has been applied to fisheye cameras in Häne *et al.* (2014) to get the direct stereo correspondence (Fig. 4.1). The difference is that instead of iterating over pixels, we iterate over a finite set of spatial planes. That is, to find a potential correspondence on the second image, we reproject a point from the first image onto one of the planes, and then project it onto the second image.

Another example of a space-sweep approach is the one presented in Caruso *et al.* (2015), where the sampling is performed along the direction corresponding to a pixel of the reference image. But the step along the spatial line in this method is chosen in a way such that it approximates as closely as possible the pixel based method of Engel *et al.* (2013). To do so the authors use the projection Jacobian matrix of the camera model to adapt the spatial step. They mention that this approach is computationally expensive.

We suggest that the pixel-sweep approach has the following advantages over the space sweep in the context of fisheye cameras:

1. Iterating over pixels of an image is computationally more efficient than projecting 3D points, since there is no need in computing a non-linear projection model and interpolating image values.

2. For fisheye cameras, space-sweep methods don't give a uniform sampling along epipolar curves. But in the case when no prior information about the 3D scene is provided, there is no reason to use different sampling steps across the image.

3. Pixel-wise sampling is a natural solution for image processing which also allows us to apply directly the Semi-Global Matching as a regularizer.

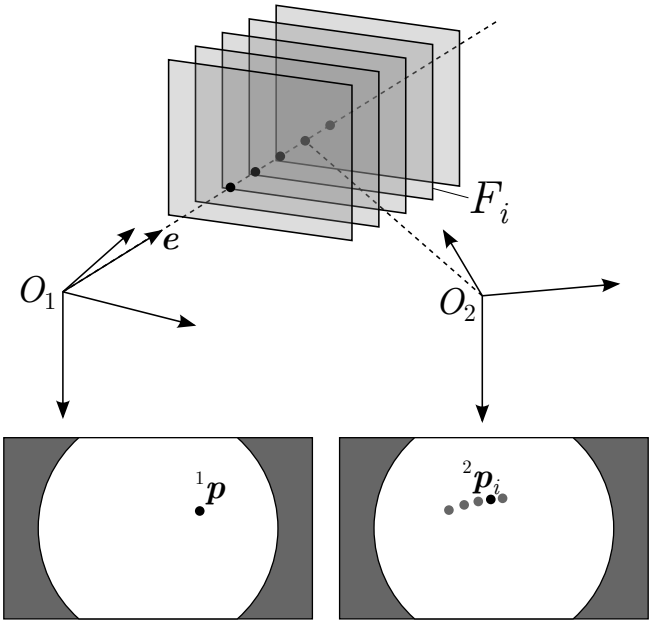


Figure 4.1: Plane-sweep stereo. To find potential matching points in the second image for point ${}^1\boldsymbol{p}$, first it is reconstructed into direction $\boldsymbol{e}$; then intersections between the ray and a set of planes $\{F_i\}$ is found. The last step is to project each intersection into the second image to get a correspondence candidate ${}^2\boldsymbol{p}_i$. It is also depicted here that a uniform space sampling (the distance between the planes is the same) leads to a non-uniform image sampling (the distance between projected points is different). So the space sampling has to be adapted to achieve a more uniform sampling in the image space.

The pixel-sweep technique aligns with the philosophy of direct stereo correspondence and direct image processing in general. Image resampling, as well as any other image filtering, leads to information loss and distortion.

1.2 Direct vs Rectified

In Häne *et al.* (2014) the advantages of the direct fisheye correspondence over rectified, pinhole-based version are mentioned. Applying the distortion-rectification mapping inevitably reduces the field of view, and the image part further away from the projection center gets stretched, while the middle part usually gets compressed (Fig. 4.2). All these effects distort the original information provided by the sensors. In contrast, the direct methods use raw images, and by doing so the field of view is not reduced, neither is the uniformity of the original information broken. The latter matters when it comes to the matching cost computation. The stretched regions have spectra with a lower frequency content than the rest, hence we have to use a larger neighborhood to extract pixel descriptors. On the other hand, certain works show that it is possible to adapt the matching methods to curved geometry of omnidirectional images. For example in Radgui *et al.* (2011), in order to compute optic flow in fisheye images, the shape of patch descriptors and geometric constraints have been adapted to the geometry of spherical projection.

Since fisheye stereo is such an attractive tool for 3D perception, there have been several techniques of fisheye rectification proposed. In Caruso *et al.* (2015) an *Array of Pinhole Models* has been used to represent wide-angle images. The fisheye image is projected onto a cube, each face of which corresponds to a pinhole camera model. The advantage is that the epipolar lines remain straight, yet the projection mapping in this case is only piecewise differentiable (if an epipolar line crosses a border between two pinhole projections, it changes its direction). This approach is convenient and efficient when the displacement between the two points of view which form the stereo system is not known, which is the case in the mentioned article.

Figure 4.2: From left to right: Original fisheye image and two undistorted versions. In the second image the scale of the neighborhood of the projection center is preserved, in the third one it is compressed in order to capture a wider field of view.

If the transformation between the two cameras is known and fixed, a more efficient technique can be applied. That is, the rectification is not based on a projection plane, like in the pinhole model. In Roy *et al.* (1997), the rectified images are reprojected onto cylinders, aligned with the stereo base, to minimize the rectification stretching effect. In Arican and Frossard (2007), a method to rectify spherical images has been proposed. In order to do that, the image sphere must be rotated so that the epipole coincide with the coordinate pole. In Abraham and Förstner (2005), each pixel on the rectified images corresponds to two angles, one of which defines the orientation of a corresponding epipolar plane, while the other defines the ray's direction within this plane. With this modeling, all epipolar lines are straight and horizontal, even though a projection of a general straight line is a curve.

The notion of *Projection Surface*, presented earlier, allows us to find a convenient way to sample fisheye images along epipolar curves, which in turn enables using a pixel-sweep technique without any rectification. Geyer and Daniilidis (2000) have showed that the Unified Camera Model projects straight lines into conics. In Chapter 2, we demonstrated that the same is true for the Enhanced Unified Camera Model, and then suggested a method to compute implicit equations of epipolar curves. It allows us to efficiently sample an image along the curve, using an implicit curve rasterization algorithm, which is a modification of one suggested in Hobby (1990).

1.3 Stereo Matching Regularization

Let us assume that for each image point we have a range of possible correspondences, and moreover for each correspondence we can compute a certain matching cost. The simplest similarity measure can be a simple difference in brightness between two pixels. But usually there are multiple pixels on the second image which satisfy the epipolar constraint and which have the same brightness or a brightness which is close to that of the original pixel. To make the matching more robust, we can use so-called descriptors to compute the matching cost, which makes points more unique. Usually, the descriptor of a point is computed using a small image patch around it. The simplest descriptor is just the patch itself (for example 3×3 pixels large).

The results of stereo matching using the simplest strategy "winner takes all" are shown in Fig. 4.3. Even though the descriptor-based method gives a better result, it is still not good enough. There is a fact which we are not using in this approach: the disparity of neighbor pixels are strongly correlated. The world we live in has a certain structure. Usually what we observe is not a random chaos of small particles floating in space, but a set of "large" objects, usually smooth. It means that two neighbor pixels are likely to belong to the same object, hence they are likely to have the same or at least similar disparity. So we can inject this prior knowledge into the correspondence problem by penalizing the difference in disparity for neighbor pixels.

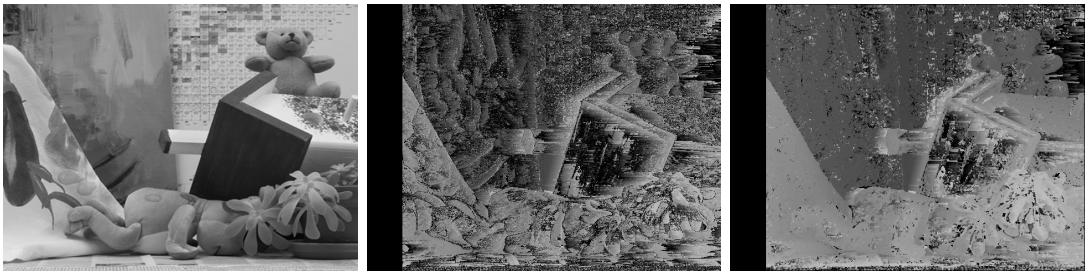


Figure 4.3: An example of a naive approach to stereo correspondence. From left to right: reference image, disparity based on a single pixel brightness, disparity computed using 3×3 pixel descriptors. The stereo image is taken from Scharstein and Szeliski (2003).

We suggest that the technique, proposed in Hirschmuller (2005), which is called *Semi-Global Matching* (SGM), is suitable for our case and gives good results (for example, Fig. 4.4). It is based on the dynamic-programming algorithm for the approximate string matching. The assumption is made that along any continuous path the disparity mostly does not change. Sometimes it makes "steps", that is changes by one, and even more rarely it makes "jumps", that is, changes by more than one. With a complexity $O(NMD_{\max})$, where N and M are the disparity map dimensions, and $D_{\max}$ is the maximum disparity value, that is being as time-consuming as a simple brute-force matching with the winner-takes-all strategy, SGM gives some of the best dense stereo reconstruction results. In should be noted that this regularizer can be naturally applied to compute the disparity map, not the depth map, because it assumes a finite (and rather small) number of possible values. On the other hand, if any kind of space-sweep technique is applied and the depth is computed directly, then it is still possible to apply it, even though a certain number of additional assumptions have to be made and it will not be as natural as in the case of a disparity map. We consider that the main virtue of the present work is that it enables the natural application of such an efficient regularizer to the fisheye stereo reconstruction problem. Let us describe the Semi-Global Matching algorithm.

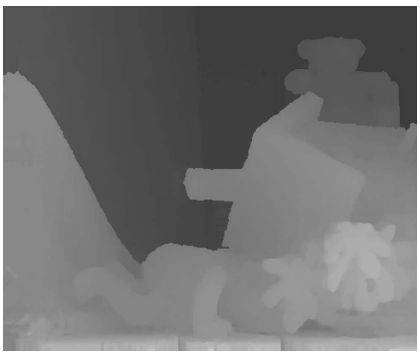


Figure 4.4: Semi-Global Matching (image from Hirschmuller (2008)). The result is much more regular than the ones presented in Fig. 4.3.

Error Accumulation The regularization algorithm requires a table of matching costs for all pixels and all possible disparity values. It means that we need to fill up a table E of size $N \times M \times D$, where N and M are the image dimensions, and $D_{\max}$ is the maximum allowed disparity. For a rectified pair of images the equation is written as:

$$E[u, v, k] = \|I[u, v] - J[u - d, v]\| \qquad v \in 1..N, u \in D_{\max}..M, d \in 0..D_{\max} \tag{4.1}$$

Here I is the left image, J is the right image. The left image is usually used as the reference image. As the second camera position is on the right of the first one, everything in the image

moves left. That is why we subtract disparity d from the column index u. Another comment: for $u < D_{\max}$ the disparity cannot be computed for all values of d, because $u - d$ becomes negative. so usually a band of $D_{\max}$ columns is skipped. Typical values of $D_{\max}$ lie between 30 and 100 depending on the stereo base (that is, the distance between cameras), the image resolution (the higher it is, the larger $D_{\max}$ may get), and the distance to the nearest objects.

Dynamic Programming The error buffer E has two spatial coordinates u, v and one disparity coordinate d. To regularize the disparity map we compute the final cost of a pixel (u, v) having a disparity value d as a sum of costs of cheapest paths through uvd space, whose footprint in uv is a straight line and passing through (u, v, d). To ensure the latter, we compute the path cost as a sum of costs of two parts: one on either side of the point. Hence the algorithm is applied twice to each path orientation: forward and backward (see Fig. 4.5).

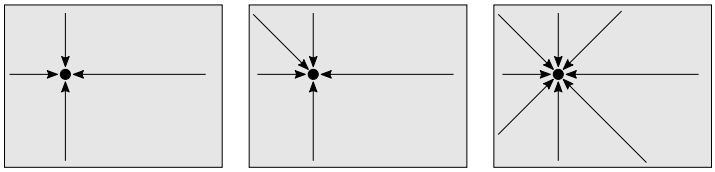


Figure 4.5: Semi-Global Matching. (a) the regularization is computed for two paths, or 4 different directions. In order to compute the cost for every point and every disparity value, the algorithm has to be applied four times: twice for every path. (b) Five directions: two complete paths and one diagonal path processed only in one sense. Even though the total length of regularization paths is different for different pixels is does not break the algorithm, since the cost is compared for different disparity along paths for the same pixel. Comparing the cost of two different pixels in this case would be meaningless. (c) four regularization paths for each pixel. The same as for (b), the total length of paths depends on pixels. The time and space complexity of the algorithm grows linearly with the number of directions.

A path's cost is defined as a sum of all errors along the path plus λ_S times the number of disparity changes by 1 (number of steps) plus λ_J times the number of disparity changes by more than 1 (number of jumps); λ_S and λ_J are the algorithm's parameters, and generally the output is stable with respect to them. The number of paths defines how regular we want the result to be. In the OpenCV library, two-and-a-half (Fig. 4.5b) and four (Fig. 4.5c) paths regularizations are implemented. In the original paper, it has been proposed to use 8 paths.

Let K be the number of directions (which are twice as many as the number of regularization paths). We need to solve the dynamic programming problem for each line in each of K directions. Let C_i be a table $N \times M \times D_{\max}$ which represents the cost of paths in ith direction. Let $c[t, d]$ be an iterator for a certain line in C_i along the ith direction, parametrized with t, that is, $c[t, d] = C_i[u_t, v_t, d]$. For example, for a horizontal direction from left to right, $u_t = t$, $v_t = v_0$. First, we need to initialize the first value:

$$c[0, d] = E[u_0, v_0, d] \tag{4.2}$$

Suppose that for a certain t, $c[t, d]$ contains the cost of the optimal path from (u_0, v_0) to (u_t, v_t), then let $c^*[t]$ be $\min_d c[t, d]$, then the following recursive relation is true (Hirschmuller (2005)):

$$\begin{aligned} c[t+1, d] = E[u_{t+1}, v_{t+1}, d] + \min(&c[t, d], c[t, d-1] + \lambda_S, \\ &c[t, d+1] + \lambda_S, c^*[t] + \lambda_J) \end{aligned} \tag{4.3}$$

Once all C_i are filled up, the final matching cost F can be computed:

$$F[u, v, d] = \sum_{i=1}^{K} C_i[u, v, d] \tag{4.4}$$

The disparity map D is defined as an image $N \times M$ with integer values in $0..D_{\max} - 1$. It is filled up as follows:

$$D[u, v] = \underset{d}{\operatorname{argmin}} F[u, v, d] \tag{4.5}$$

2 Direct Fisheye Stereo Matching

The first stage of most matching algorithms is the matching cost computation. That is for every pixel and every possible matching candidate the cost has to be computed. In this approach, candidates are the pixels which lie on the epipolar curve within a certain range. Then the regularization strategy comes in to select the final stereo matches.

Here we adapt the technique used in Engel *et al.* (2013); Caruso *et al.* (2015), which consists in describing a pixel by a sequence of samples along the epipolar line passing through it on the first image. For a particular pixel the pipeline is as follows:

1. Using the inverse camera model, reconstruct the direction corresponding to the pixel.
2. Compute the epipolar curve equation for both first and second cameras.
3. By sampling the first image along the epipolar curve, compute a pixel's descriptor $\mathfrak{D} = \{\mathfrak{D}_1..\mathfrak{D}_M\}$ (Gothic D).
4. By sampling the second image along the epipolar curve, compute a sample sequence $\mathfrak{S} = \{\mathfrak{S}_1..\mathfrak{S}_N\}$ (Gothic S).
5. For each disparity value compute the matching cost.

Here the curve rasterization is done with a step of one pixel (in 8-neighbor system), and the disparity here means the number of steps along the curve. The descriptor $\mathfrak{D}$ is a small sampling sequence along the epipolar curve, centered at the pixel in question.

Disparity Scale In order to accelerate the algorithm, we don't compute the disparity for every pixel. Since we assume that the objects on the image are bigger than one pixel, using the same sampling frequency for the depth map is redundant. On the other hand the semi-global matching algorithm requires a regular lattice to apply the dynamic-programming based regularization. We choose to reduce the depth estimation frequency by an integer factor, typically 2 or 3. Since we use pixel descriptors, which somehow encode the information about its neighborhood, the photometric information is not lost, but rather integrated in the computed disparity value. Yet this algorithm can work when the disparity scale and the image scale are the same.

2.1 Curve Rasterization

To be able to use the pixel-sweep approach, the first step is to rasterize epipolar curves for a stereo system, which have been computed in Chapter 2. Following a curve pixel by pixel as closely and accurately as possible is equivalent to its rasterization. The following rasterization algorithm is inspired by Hobby (1990). Let $\phi : \mathbb{R}^2 \to \mathbb{R}$ be a level function. The algorithm requires that the curve be defined as the zero-level of the level function:

$$\phi(u, v) = 0 \tag{4.6}$$

Let $\nabla\phi = (\phi_u, \phi_v)^T$ be the level function gradient; $\nabla\phi$ evaluated on the curve defined by ϕ and treated as a vector corresponds to its normal. Also the algorithm requires a starting pixel $\boldsymbol{p}_\mathrm{s} = (u_\mathrm{s}, v_\mathrm{s})^T$ and a goal $\boldsymbol{p}_\mathrm{g} = (u_\mathrm{g}, v_\mathrm{g})^T$ (see Fig. 4.6). Since the image is discrete, we

cannot require that $\phi(\boldsymbol{p}_\mathrm{s}) = 0$. But at least we can assume that the starting point and the goal are close to the curve. That is, the distance from the nearest solution of (4.6) to either point is less than 1. We assume that $\boldsymbol{p}_\mathrm{s} \in \mathbb{Z}^2$ and it might be the case that $\boldsymbol{p}_\mathrm{g} \in \mathbb{Z}^2$, but it is not necessary. Among the two possible directions of the curve starting at $\boldsymbol{p}_\mathrm{s}$, the algorithm chooses the one which leads towards the goal $\boldsymbol{p}_\mathrm{g}$. More formally, if the chosen direction is defined by a tangent vector $\boldsymbol{\tau}$, then:

$$\boldsymbol{\tau} \cdot (\boldsymbol{p}_\mathrm{g} - \boldsymbol{p}_\mathrm{s}) > 0 \tag{4.7}$$

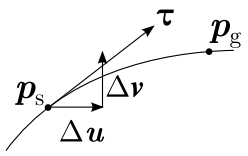


Figure 4.6: A curve rasterization process. We start from $\boldsymbol{p}_\mathrm{s}$ and make step $\Delta\boldsymbol{u}$ towards $\boldsymbol{p}_\mathrm{g}$, because tangential vector $\boldsymbol{\tau}$ is closer to u-axis, than to v-axis. Then we use the perpendicular direction to compensate the error by making step $\Delta\boldsymbol{v}$.

We assume that the angle between $\boldsymbol{\tau}$ and the desired direction is small, so the singular configuration when $\boldsymbol{\tau} \cdot (\boldsymbol{p}_\mathrm{g} - \boldsymbol{p}_\mathrm{s}) = 0$ is supposed to never happen. $\boldsymbol{\tau}$ is calculated by rotating gradient vector $\nabla\phi$ by $\pi/2$ clockwise or counterclockwise depending on the goal condition. Then we retain the rotation direction in a variable ε and use it until end of rasterization process. $\varepsilon = 1$ means counterclockwise, while $\varepsilon = -1$ means clockwise. It shall be noted that:

$$\forall \boldsymbol{q} \in \mathbb{R}^2 \quad \boldsymbol{\tau} \cdot \boldsymbol{q} \propto \varepsilon \nabla\phi \times \boldsymbol{q} \tag{4.8}$$

Now we can describe Algorithm 1. The goal is to get N_max samples of a rasterized algebraic curve, starting from a certain initial point $\boldsymbol{p}_\mathrm{s}$, which we know belongs to the curve, towards another curve point $\boldsymbol{p}_\mathrm{g}$ (let us call it the goal point).

The idea is that, in order to follow the curve as closely as possible, at every step, we check which direction, u or v, is closer to the actual tangential direction of the curve. To do it, we compare components of $\nabla\phi$ by absolute value. Let us say, $|\phi_v| > |\phi_u|$. It means that $\boldsymbol{\tau}$ is closer to u-direction. So we change u by ± 1 depending on the sign of ϕ_v and the rotation direction that has been defined. Then we check what is the actual error of (4.6) and apply a simple proportional control law to decrease it:

$$\Delta v = -\mathrm{round}\left(\frac{\phi(\boldsymbol{p})}{\phi_v(\boldsymbol{p})}\right) \tag{4.9}$$

The described algorithm is general and works for any ϕ given that it does not contain features too fine to be properly depicted with the image's sampling step.

Algorithm 1 Algebraic curve rasterization

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be an implicit curve
Let $\nabla\phi = (\phi_u, \phi_v)^T$ be its gradient
Let $\boldsymbol{p} = (u, v)^T$ be an image point
Let $\boldsymbol{p}_\mathrm{s}$ be the initial point
Let $\boldsymbol{p}_\mathrm{g}$ be the goal point
$\boldsymbol{p} \leftarrow \boldsymbol{p}_\mathrm{s}$
if $\det\left[\nabla\phi(\boldsymbol{p}_\mathrm{s}) \quad (\boldsymbol{p}_\mathrm{g} - \boldsymbol{p}_\mathrm{s})\right] > 0$ **then**
    $\varepsilon \leftarrow 1$ ▷ The rotation is counterclockwise
else
    $\varepsilon \leftarrow -1$ ▷ The rotation is clockwise
end if
for $n = 1..N_\mathrm{max}$ **do** ▷ We need only N_max samples
    if $|\phi_u(\boldsymbol{p})| > |\phi_v(\boldsymbol{p})|$ **then** ▷ $\boldsymbol{\tau}$ is closer to v
        $v \leftarrow v + \varepsilon\mathrm{sign}\,(\phi_u(\boldsymbol{p}))$
        $u \leftarrow u - \mathrm{round}\,(\phi(\boldsymbol{p})/\phi_u(\boldsymbol{p}))$
    else ▷ $\boldsymbol{\tau}$ is closer to $\boldsymbol{u}$
        $u \leftarrow u - \varepsilon\mathrm{sign}\,(\phi_v(\boldsymbol{p}))$
        $v \leftarrow v - \mathrm{round}\,(\phi(\boldsymbol{p})/\phi_v(\boldsymbol{p}))$
    end if
    PROCESS($\boldsymbol{p}$)
end for

Second Order Curves In our case $\phi(u, v)$ is a polynomial function of degree 2. It means that:

- ϕ_u and ϕ_v are easier to calculate than ϕ itself
- $\phi(u+1, v) = \phi(u, v) + \dfrac{\phi_u(u, v) + \phi_u(u+1, v)}{2}$
- $\phi(u, v+1) = \phi(u, v) + \dfrac{\phi_v(u, v) + \phi_v(u, v+1)}{2}$

These facts lead us to a better version of the algorithm. Instead of calculating ϕ on each step we can calculate it once at the beginning of the process, and then using only ϕ_u and ϕ_v to refresh the value. The general structure of this version of the algorithm is the same as before, but every time after we change u or v we have to perform the additional sequence (see Algorithm 2).

An implementation of both algorithms has been tested and the second one runs faster by 30%. But it is exact only when ϕ is a polynomial function of degree 2. In any other case, it will slowly diverge. Again, in case of stereo correspondence the curve part to be rasterized is rarely longer then 100 px, and for smooth curves, the divergence can be neglected. On the other hand the speedup is based on the fact that ϕ_u is easier to evaluate than ϕ. If it is not the case, the second version will actually run slower because it evaluates ϕ_u and ϕ_v in total 4 times per iteration, while the simpler version calls ϕ once per iteration and ϕ_u and/or ϕ_v 3 times. So the difference is exactly one ϕ against one ϕ_u.

2.2 Epipolar Curve Precomputation

The naive way of computing correspondence cost would be to compute the epipolar curve for every pixel. But since the same epipolar curve passes through multiple pixels, that approach

Algorithm 2 Algebraic curve rasterization — computation optimization by incremental error estimation

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be an implicit curve
Let $\nabla\phi = (\phi_u, \phi_v)^T$ be its gradient
Let $\boldsymbol{p} = (u, v)^T$ be an image point
Let $\boldsymbol{p}_\mathrm{s}$ be the initial point
Let $\boldsymbol{p}_\mathrm{g}$ be the final point
$\delta \leftarrow \phi(\boldsymbol{p}_\mathrm{s})$ ▷ Initial error
$(g_u, g_v)^T \leftarrow \nabla\phi(\boldsymbol{p}_\mathrm{s})$ ▷ Initial gradient
$\boldsymbol{p} \leftarrow \boldsymbol{p}_\mathrm{s}$
if $\det\left[\nabla\phi(\boldsymbol{p}_\mathrm{s}) \quad (\boldsymbol{p}_\mathrm{g} - \boldsymbol{p}_\mathrm{s})\right] > 0$ **then**
    $\varepsilon \leftarrow 1$ ▷ The rotation is counterclockwise
else
    $\varepsilon \leftarrow -1$ ▷ The rotation is clockwise
end if
for $n = 1..N_{max}$ **do**
    if $|g_u| > |g_v|$ **then** ▷ $\boldsymbol{\tau}$ is closer to v
        CHANGEV$(\varepsilon \mathrm{sign}\,(g_u))$
        CHANGEU$(-\mathrm{round}\,(\delta / g_u))$
    else ▷ $\boldsymbol{\tau}$ is closer to $\boldsymbol{u}$
        CHANGEU$(-\varepsilon \mathrm{sign}\,(g_v))$
        CHANGEV$(-\mathrm{round}\,(\delta / g_v))$
    end if
    PROCESS$(\boldsymbol{p})$
end for
procedure CHANGEU(s)
    $u \leftarrow u + s$
    $h_u \leftarrow \phi_u(\boldsymbol{p})$
    $\delta \leftarrow \delta + 0.5s(g_u + h_u)$
    $g_u = h_u$
    $g_v \leftarrow \phi_v(\boldsymbol{p})$
end procedure
procedure CHANGEV(s)
    $v \leftarrow v + s$
    $h_v \leftarrow \phi_v(\boldsymbol{p})$
    $\delta \leftarrow \delta + 0.5s(g_v + h_v)$
    $g_v = h_v$
    $g_u \leftarrow \phi_u(\boldsymbol{p})$
end procedure

would be redundant. Moreover, computing an epipolar curve equation for every pixel takes significant time. We suggest to precompute epipolar curves for a certain number of angles, and then for any given pixel just fetch the corresponding curve coefficients. To make fetching curves efficient we should avoid calling computationally intensive functions like arcsin or arctan.

Another thing to notice is that a plane, rotated about the stereo base by π, defines the same epipolar equation. So only planes within π radians are to be computed. This range can be divided into two sectors: $-\pi/2$ to $\pi/2$ and $\pi/2$ to $3\pi/2$. The equations are precomputed uniformly with respect to the tangent within the first sector, and to the cotangent within the second one.

To compute tangent or cotangent we need to compute unnormalized cosine $\tilde{c}$ and sine $\tilde{s}$. In order to do it, we project a point $\boldsymbol{X}$ that belongs to the plane in question onto x- and y-axes respectively. For that, we need to construct a stereo basis $\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z$. Its z-axis must be aligned with the translation part of the transformation between the two cameras. Then the orientation of the plane containing a given direction from the first camera can be defined as the angle between x-axis of the stereo frame and the projection of the direction vector onto the xy-plane.

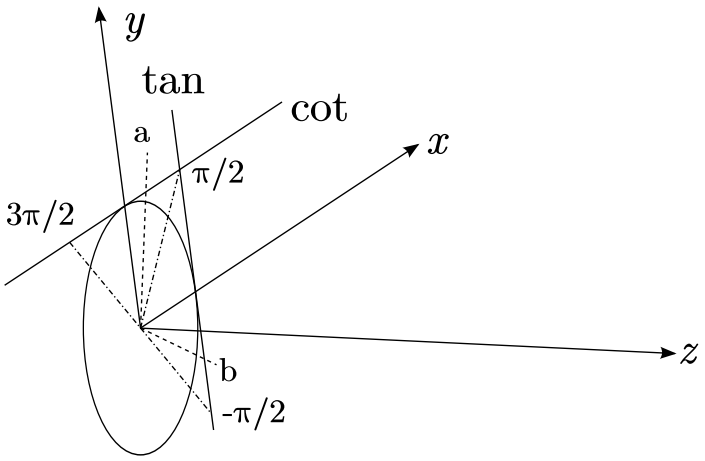


Figure 4.7: The stereo frame. z-axis passes through the other camera's projection center. The plane that intersects the xy-plane along (a) is indexed according to the cotangent, or the intersection between cot-axis and (a), the plane corresponding (b) is indexed according to the tangent.

The algorithm to query the plane data structure is:

1. Compute $\tilde{c} = \boldsymbol{X} \cdot \boldsymbol{e}_x$ and $\tilde{s} = \boldsymbol{X} \cdot \boldsymbol{e}_y$.
2. If $|\tilde{c}| + |\tilde{s}| < \varepsilon$, then the direction is almost aligned with the epipole, and the plane is ill-defined. Return any plane.
3. If $|\tilde{c}| > |\tilde{s}|$, then we can compute the plane index based on $\tan(\vartheta) = s/c$
4. If $|\tilde{s}| > |\tilde{c}|$, then we can compute the plane index based on $\cot(\vartheta) = c/s$
5. The index is computed from the previously computed value v (either tan or cot) as round $\left(\lambda^{-1}(1 - v)\right)$. Where λ^{-1} is the inverted sampling step, precomputed to speedup the computation.

Even though the distribution of the curves will not be uniform with respect to angles, by generating 2000 curves we are almost sure that every pixel has its own curve for 1 Mpx images. It takes little time (much less than computing a new curve for every pixel), and the query

procedure creates little overhead: two dot-products, one division and one multiplication. It is much more important for monocular applications, where we have to recompute the epipolar geometry at every time step, than for a calibrated stereo system.

2.3 Point Descriptor Scale

Another way of speeding up the algorithm is to vary the descriptor scale as a function of its content. That is, if after computing the descriptor we see that the value variation that it contains is of the same order as the expected image noise, then we can increase the sampling step without loosing much information, since the frequency content of the image in the pixel's neighborhood is poor. Lower frequency of the descriptor sampling has the following effects:

1. The second image can be sampled with the same frequency change, which leads to a lower computation time.
2. Since the semi-global matching requires a matching cost for every disparity value, the matching cost must be interpolated to "fill the gaps".
3. The estimated depth uncertainty will be larger, because it is related to the sampling step. But it does not mean that this approach makes the precision worse. It rather estimates the uncertainty in a more fair way, because we cannot expect a pixel-level precision in a textureless region.

To do that for a given pixel a basic descriptor $\mathfrak{D}$ with 1 px sampling step is extracted. Then its total variation is computed:

$$TV_{\mathfrak{D}} = \sum_{i=1}^{M-1} |\mathfrak{D}_{i+1} - \mathfrak{D}_i| \tag{4.10}$$

Here, M is the size of $\mathfrak{D}$. If $TV_{\mathfrak{D}}$ is lower than a certain threshold (which should depend on the noise level of the image), then the descriptor is recomputed with a larger sampling step. You can see an example of this approach in Fig. 4.8. If we know that the patch around the pixel is "flat", then we can assume that it will be the case on the other image as well, so there is no need for fine sampling. It means that, for the same disparity range fewer samples are drawn from the second image. For example, if the descriptor with sampling step 3 is good enough, then the number of samples is 3 times less. Then, depending on the stereo correspondence mode, one can either take the best correspondence as the disparity estimation, but with a larger uncertainty number; or another option (which is used in our SGM implementation) is to linearly interpolate the matching cost for missing disparity values.

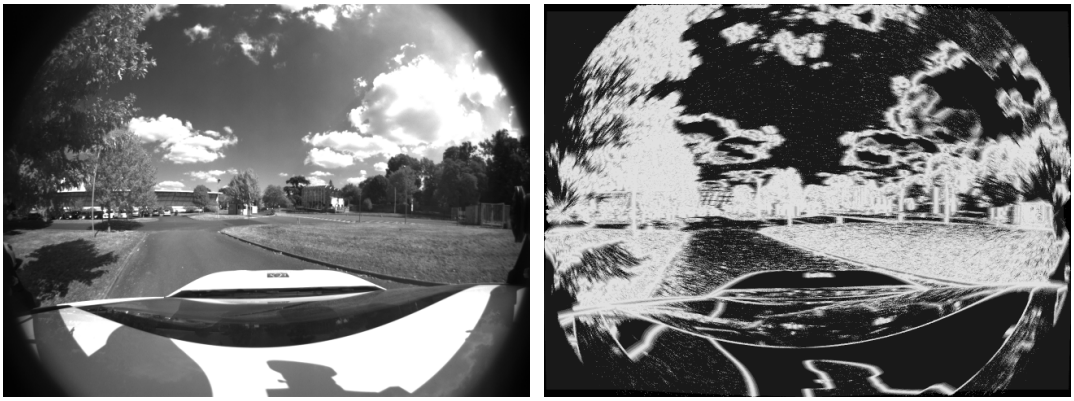


Figure 4.8: Left image of a stereo image pair (left) with its saliency estimation (right). The brighter a pixel on the saliency image, the smaller the sampling step for its descriptor. As shown by the right image, a significant part of the image has low saliency.

3 Reconstruction and Uncertainty Estimation

After the stereo correspondence has been established, the next step is to reconstruct the depth and to estimate the uncertainty of the reconstruction. A proper uncertainty estimation is crucial for any metrological application. And we consider this 3D reconstruction as a measurement tool which can be reliably used for real world application in future developments.

3.1 Reconstruction

A scheme of input data for the triangulation is shown in Fig. 4.9. Intuitively, the triangulated point lies on the intersection of the two rays. But generally, in the real world, two rays represented by a pair of corresponding pixels and the transformation between cameras don't intersect because of discretization errors and uncertainties in the extrinsic and intrinsic parameters. One approach to this problem is to find the point nearest to both rays. Such a point lies in the middle of the common perpendicular. Then, if the real spatial point is far away in comparison to the distance between cameras, it might be the case that the rays actually diverge, so the nearest point is behind the camera. In this work we propose an alternative way of triangulation, the so-called *regularized triangulation*, which is described in Chapter 5. Here, we just mention that the method reconstructs the point in front of the camera even for diverging rays. And this method is used for the stereo reconstruction.

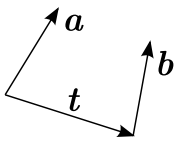


Figure 4.9: The triangulation consists in finding a spatial point which best fits both observation directions $\boldsymbol{a}$ and $\boldsymbol{b}$ for a given stereo base $\boldsymbol{t}$. In the perfect scenario, both rays intersect. In the real world, because of the noise, they do not.

Let us introduce some formalism. Let $\mathfrak{h} : \Omega \times \Omega \to \mathbb{R}$ be the reconstruction function. It takes a point $\boldsymbol{p}_1$ from the base image and corresponding point $\boldsymbol{p}_2$ from the second image, which is computed using its disparity value d computed during the matching step, and outputs the distance λ from the base camera's projection center to the presumed 3D point. Let $\boldsymbol{a} = \text{normalize}(\mathfrak{f}^{-1}(\boldsymbol{p}_1))$ be the normalized reconstructed direction vector corresponding to

$\boldsymbol{p}_1$. then the reconstructed point is defined as follows:

$$\boldsymbol{X} = \lambda \boldsymbol{a} \tag{4.11}$$

Then λ is computed for each $\boldsymbol{p} \in \Omega_1$ and stored in $\mathcal{D}$, the *depth map*, also defined on Ω.

3.2 Uncertainty Estimation

We base the uncertainty estimation on the following assumption. If the correspondence is correct, then the actual correspondence lies on the epipolar line within ± 1 px from the found position, because of the discretization error. Let $\mathfrak{s} : \Omega \to \Omega$ be a shift function, which shifts an image point along its epipolar curve by one step towards the epipole. The first try to estimate the uncertainty was to reconstruct the point twice (Fig. 4.10a):

$$\begin{aligned} \lambda_0 &= \mathfrak{h}(\boldsymbol{p}_1, \boldsymbol{p}_2) \\ \lambda_1 &= \mathfrak{h}\left(\boldsymbol{p}, \mathfrak{s}_2\left(\boldsymbol{p}_2\right)\right) \\ \sigma &= \lambda_0 - \lambda_1 \end{aligned} \tag{4.12}$$

λ_1 is less than λ_0, as it is computed for a greater disparity. Then we assumed that λ_0 was the depth estimation which was stored in $\mathcal{D}(\boldsymbol{p}_1)$. σ is stored in uncertainty map $\Sigma(\boldsymbol{p}_1)$.

Then we found out that this way of uncertainty estimation tends to underestimate the uncertainty for certain configurations (see Fig. 4.10b). For example when the patch around the pixel gets stretched or compressed because of the change in the angle between the object's surface and the view line, one pixel on the first image may correspond to 3-4 pixels on the other. In order to avoid this underestimation, we assume that there is a correspondence uncertainty which is the maximum between the error induced by an error in disparity of one pixel on either image. That is, if a step of one pixel in the left image changes the depth estimation as much as a step of three pixels in the right image, then that is the best precision we can expect from the system. The final uncertainty estimation method is illustrated in Fig. 4.10c. Formally, it is:

$$\begin{aligned} \lambda_0 &= \mathfrak{h}(\boldsymbol{p}_1, \boldsymbol{p}_2) \\ \lambda_1 &= \mathfrak{h}\left(\mathfrak{s}_1\left(\boldsymbol{p}_1\right), \mathfrak{s}_2\left(\boldsymbol{p}_2\right)\right) \\ \sigma &= \lambda_0 - \lambda_1 \end{aligned} \tag{4.13}$$

Here $\mathfrak{s}_1$ and $\mathfrak{s}_2$ are the shift functions for the first and second camera respectively.

There is another source of uncertainty, related to the discretization error in the direction perpendicular to the epipolar curves. A way to approximate this uncertainty has been described in Engel *et al.* (2013). Even though the underlying error source for that demonstration is the error in the extrinsic parameters, the concept can be applied to our algorithm with a constant error of ± 1 px in the epipolar line position. The complication comes from the fact, that in order to estimate this uncertainty, we have to use image data, not only the geometric properties of the system, as the error depends on the gradient in the neighborhood of the point under consideration. We don't take into account this source of uncertainty, yet it is a feasible task.

3.3 Reconstruction Using Tracking

In case of a sequence of images of the same scene taken from known positions, we can achieve a significant speedup by using the semi-global matching regularization only once, for the shortest stereo base. A small base, combined with SGM, gives us high accuracy. The precision can be improved by using other images from the sequence and predictive search. The reconstruction

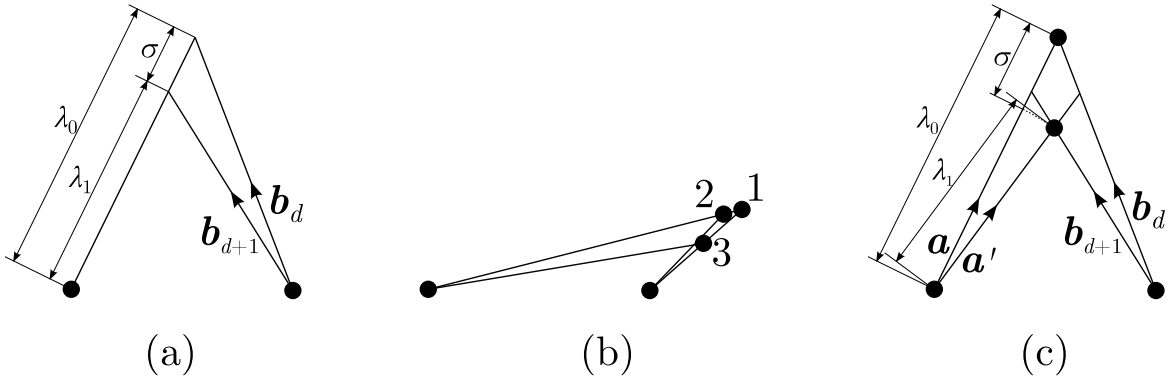


Figure 4.10: The uncertainty estimation: (a) the first try to estimate the uncertainty, after triangulating the point twice, for the corresponding direction $\boldsymbol{b}_d$ and for the direction $\boldsymbol{b}_{d+1}$, shifted along its epipolar curve towards the epipole by one pixel. (b) illustration of a case when the described method fails. Point 1 is the triangulation using the corresponding directions. Then, if the left camera is the base one, the uncertainty is estimated as the distance between points 1 and 2. On the other hand if the right camera were the base one, the uncertainty would be the distance between points 1 and 3. The angles between two rays are the same for both viewpoints. As illustrated, this approach to uncertainty estimation is not symmetric. (c) A symmetric version of the method. the point reconstructed first for the corresponding directions $\boldsymbol{a}$ and $\boldsymbol{b}_d$, then for the directions $\boldsymbol{a}'$ and $\boldsymbol{b}_{d+1}$ shifted along the epipolar curve towards the epipole by 1 px. The uncertainty σ is the difference in the distances for both reconstructions

from the first two images gives us the estimation of the epipolar curve span, where to look for the correspondence. And the search span for larger bases is small enough to avoid most ambiguous matches. In addition, due to the increase in the stereo base, it improves the depth estimation precision. The method is illustrated in Fig. 4.11.

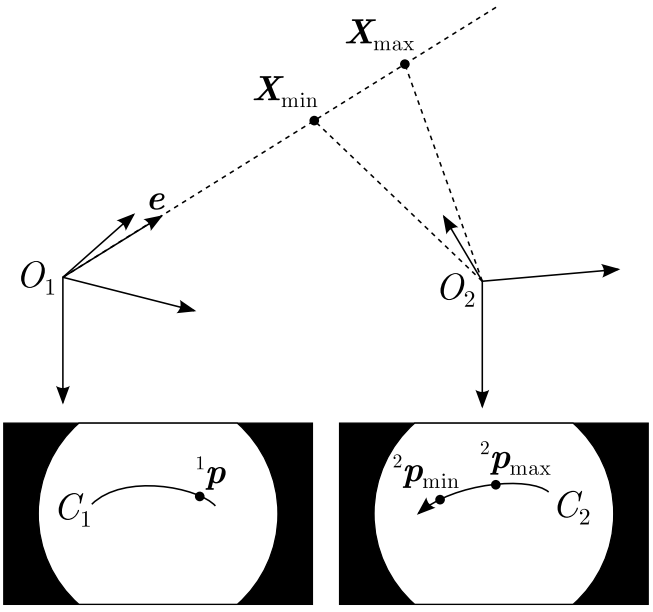


Figure 4.11: Predictive stereo matching using a prior depth estimation. If we want to match pixel ${}^1\boldsymbol{p}$, then first we compute the direction vector $\boldsymbol{e}$, then find the corresponding epipolar curves C_1 and C_2. We use C_1 to compute the descriptor. Then we compute the uncertainty interval on the second image by reconstructing the 3D point twice using the lower and upper bounds on the depth uncertainty, which gives us $\boldsymbol{X}_{\min}$ and $\boldsymbol{X}_{\max}$. By projecting them we get the limits on the search interval on the second image (${}^2\boldsymbol{p}_{\min}$ and ${}^2\boldsymbol{p}_{\max}$). Disparity is counted starting from ${}^2\boldsymbol{p}_{\max}$.

Regularization This basic strategy treats each depth pixel independently without using the fact that neighbor pixels are correlated. To make use of this fact, we can apply a basic regularization technique. For each pixel $\boldsymbol{p} \in \Omega$ a subset of neighbors which match it is selected:

$$\boldsymbol{Q} = \{\boldsymbol{q} \mid \boldsymbol{q} \in \mathcal{N}_{\boldsymbol{p}} \quad \text{and} \quad \mathcal{D}(\boldsymbol{q}) \in [\mathcal{D}(\boldsymbol{p}) - 2\Sigma(\boldsymbol{p}), \mathcal{D}(\boldsymbol{p}) + 2\Sigma(\boldsymbol{p})]\} \tag{4.14}$$

where $\mathcal{N}_{\boldsymbol{p}}$ is the neighborhood of $\boldsymbol{p}$; Ω is the domain of depth value definition. Then if $|\boldsymbol{Q}| < t_{\min}$, then $\boldsymbol{p}$ is rejected as an outlier and $\mathcal{D}(\boldsymbol{p})$ is set to 0. Otherwise the value is filtered:

$$\mathcal{D}'(\boldsymbol{p}) = \frac{\lambda \mathcal{D}(\boldsymbol{p}) + \sum_{\boldsymbol{q} \in \boldsymbol{Q}} \mathcal{D}(\boldsymbol{q})}{\lambda + |\boldsymbol{Q}|} \tag{4.15}$$

This simple regularization strategy, similar to one used in Engel *et al.* (2013), is efficient, as the experimental results show.

Pixel Reinitialization If a pixel does not have a prior depth estimation and yet it is classified as salient, that is its descriptor has a lot of variation, the algorithm will try to initialize through a brute-force search along the whole range of possible disparity values. If a mismatch occurs, it is likely that it will be filtered out by the regularizer. But if the measurement is consistent with its neighbor values, it will be used on the next iteration, and so the depth map is slowly re-populated with depth measurements.

4 Improvement Techniques

Here we propose two techniques to improve the stereo matching. Both of them are related to the matching cost computation during the search along epipolar lines. The first one allows us to take into account the descriptor deformation because of the viewpoint change. The second one aims to treat more precisely sharp brightness transitions in the context of spatial discrete sampling. An experimental validation of the proposed techniques is given in the following section "Experimental Results".

4.1 Dynamic Programming Matching Cost

We know that in the absence of occlusions all the points will appear on the other image on the epipolar line. In case of pinhole camera and narrow-angle fields of view, the distance between the points remains close to the one on the original image. Hence, just a simple comparison of the original descriptor with each subsequence of the same length sampled along the epipolar line allows us to do the matching with good precision, except for vertical surfaces which are observed at sharp angles. But in case of fisheye optics, the motion of objects far from the optical axis causes significant distortions and direct comparison of patches of two images, sampled with the same step, is not accurate anymore. A solution would be to look for an affine transformation of the descriptor, as has been done in Shi and Tomasi (1993). But doing so directly would significantly increase the algorithm's complexity and time constants, because the transformation has to be found for every pixel and for every disparity value.

We propose a dynamic-programming algorithm to approach this problem efficiently. The algorithm is based on the optimal sequence alignment algorithm Needleman and Wunsch (1970), a slight modification of which is used in the Semi-Global Matching algorithm. The problem formulation is the following: for each disparity value, compute the matching cost given that the points of the descriptor on the first image might correspond to the same point on the second image, or might have an additional unmatched pixel in between (see Fig. 4.12). This formulation includes most of the nonlinear deformation of the descriptor starting from extension by a factor of 2 to the complete collapse of the descriptor into a single pixel.

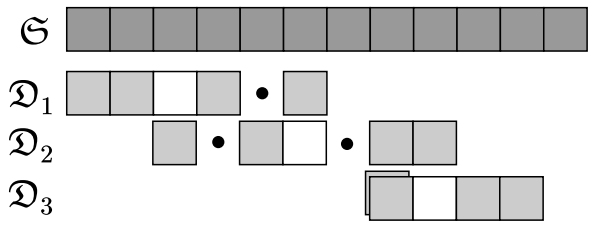


Figure 4.12: Examples of possible alignments of descriptors $\mathfrak{D}_1$, $\mathfrak{D}_2$, $\mathfrak{D}_3$ with sample sequence $\mathfrak{S}$. A circle represents a gap, the white cells are the middles of the descriptors, their position in the alignment defines the disparity value which the alignment cost is associated to.

The algorithm's complexity remains the same as that of brute-force matching of a descriptor against a sequence of samples for different disparity values, that is $O(NM)$, but with a slightly worth constant. Here N is the sample sequence length, M is the descriptor size.

To clarify the algorithm, the main three operations are illustrated on Fig. 4.13.

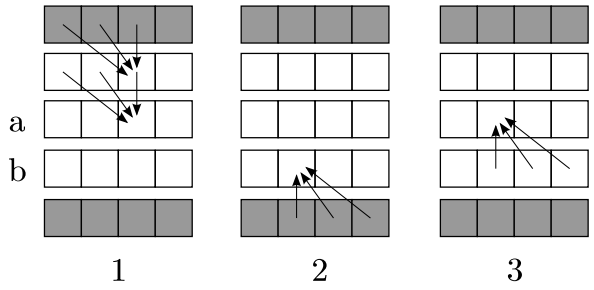


Figure 4.13: An illustration of the dynamic programming algorithm (here the descriptor is 5 pixels long). Each column of the table corresponds to a disparity value, each row corresponds to a pixel in the descriptor. (1) The algorithm accumulates cost for the first half of the descriptor; it goes from the first row to the middle row (a). (2) Then it computes the cost for the second half of the descriptor; it goes from the last row up to the row after the middle one (b). (3) The algorithm "stitches together" rows a and b. This is similar to an iteration of (2), except for the error for row a. After that, row a contains the optimal cost of every disparity value.

4.2 Correspondence Cost for Pixels

In Algorithm 3 the simple absolute difference in brightness is used as the cost of matching two pixels. We propose another measure, which works particularly well in case of high-contrast transitions. First let us suppose that we sample a continuous signal, which contains a smoothed step (Fig. 4.14a), two times with the same sampling rate but with a small shift. Even though the samples are close to each other, if we try to match these two sequences, using the absolute difference as the similarity measure, we will find out that the minimum error is high for any correspondence. This example is a simplified model of what happens when we are matching two real sequences if they represent a brightness transition on the image. If the sequence of samples from the second image contains multiple transitions, even though the limit brightnesses from the left and from the right might be different from the limit brightnesses of the transition represented by the descriptor, the final error can be significantly lower because the samples are "better" distributed (Fig. 4.14b).

To deal with this kind of situations, the following measure ϕ can be used:

$$\begin{aligned}
\Theta_i &= \max(\mathfrak{D}_i, \tfrac{\mathfrak{D}_i+\mathfrak{D}_{i-1}}{2}, \tfrac{\mathfrak{D}_i+\mathfrak{D}_{i+1}}{2}) \\
\vartheta_i &= \min(\mathfrak{D}_i, \tfrac{\mathfrak{D}_i+\mathfrak{D}_{i-1}}{2}, \tfrac{\mathfrak{D}_i+\mathfrak{D}_{i+1}}{2}) \\
\phi_i(\mathfrak{S}_j) &= \max(0, \vartheta_i - \mathfrak{S}_j, \mathfrak{S}_j - \Theta_i)
\end{aligned} \tag{4.16}$$

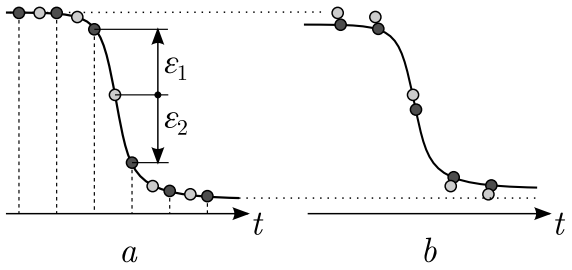


Figure 4.14: Sampling a continuous signal. Dark circles are sample sequences, light circles represent the descriptors. In case of a sharp value change, a small shift in sampling phase leads to large errors in measuring similarity using simple absolute difference. Even though on (a) both sample sequences represent exactly the same signal (assuming that it satisfies the Shannon-Nyquist frequency condition), the least error after the alignment will be greater than ε_1 or ε_2. On the other hand the same descriptor would give a smaller matching error with a sequence, representing a different signal on (b).

Here $\mathfrak{D}_i$ is a descriptor element, $\mathfrak{S}_j$ is a sample from the sequence.

Also, for the first and the last descriptor elements we have to take care of the fact that there is neither $\mathfrak{D}_0$, nor $\mathfrak{D}_{M+1}$. The cost concept is illustrated on Fig. 4.15. Let us call it *cutoff cost*. Such a strategy provides that if a sample value is in between two consecutive descriptor values, it will give a zero cost for at least one of them.

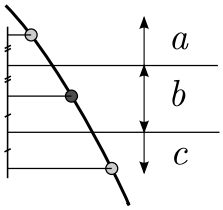


Figure 4.15: The visualization of the pixel matching cost function ϕ. The darker circle is the descriptor sample for which the matching cost is to be computed. Lighter circles are its neighbors. Value span (b) corresponds to 0 cost. Spans (a) and (c) correspond to the linear cost growth, as the value gets farther away from the descriptor sample.

Algorithm 3 Dynamic programming algorithm for the descriptor matching cost computation. For the sake of clarity, here we assume that all the values beyond the array limits are ∞

Let $\mathfrak{D} = [\mathfrak{D}_1..\mathfrak{D}_M]$ be a descriptor
Let $\mathfrak{S} = [\mathfrak{S}_1..\mathfrak{S}_N]$ be a sample sequence
Let λ be the flaw cost
$H \leftarrow \text{floor}(M/2)$ ▷ The half length
$A \leftarrow \text{Array}(N)$
$B \leftarrow \text{Array}(N)$
for $j = 1..N$ **do** ▷ Process the first half
    $A_j \leftarrow |\mathfrak{S}_j - \mathfrak{D}_1|$
end for
for $i = 2..H + 1$ **do**
    for $j = 1..N$ **do**
        $B_j \leftarrow \min(A_{j-2} + \lambda, A_{j-1}, A_j + \lambda) + |\mathfrak{S}_j - \mathfrak{D}_i|$
    end for
    $\text{Swap}(A, B)$
end for
$C \leftarrow \text{Array}(N)$
$\text{Swap}(A, C)$
for $j = 1..N$ **do** ▷ Process the second half
    $A_j \leftarrow |\mathfrak{D}_j - \mathfrak{D}_M|$
end for
for $i = N - 1..H + 2$ **do**
    for $j = 1..N$ **do**
        $B_j \leftarrow \min(A_{j+2} + \lambda, A_{j+1}, A_j + \lambda) + |\mathfrak{S}_j - \mathfrak{D}_i|$
    end for
    $\text{Swap}(A, B)$
end for
for $j = 1..N - 2$ **do** ▷ Stitch the two halves together
    $C_j \leftarrow C_j + \min(A_{j+2} + \lambda, A_{j+1}, A_j + \lambda)$
end for
$C_{N-1} \leftarrow C_{N-1} + \min(A_N, A_{N-1} + \lambda)$
$C_N \leftarrow C_N + A_N + \lambda$
return C

5 Experimental Results

A brief experimental validation for matching techniques is given in this section, to justify their use. Then a more thorough testing of the algorithm is provided.

5.1 Testing Setup

The algorithms are tested using sequences of synthetic images [1]. The ground truth depth $\mathcal{D}^*$ is available.

Quality Criteria For each image sequence the following two quantities are computed:

- Inliers rate — how many depth values actually conform to the predicted uncertainty. The criterion is simple:

$$|\mathcal{D}(\boldsymbol{p}) - \mathcal{D}^*(\boldsymbol{p})| < \Sigma(\boldsymbol{p}) \qquad \boldsymbol{p} \in \Omega \tag{4.17}$$

- The error standard deviation:

$$\sigma_{\text{err}} = \sqrt{\frac{\sum\limits_{\boldsymbol{p}\in\Omega'} \left(\mathcal{D}\left(\boldsymbol{p}\right) - \mathcal{D}^*\left(\boldsymbol{p}\right)\right)^2}{|\Omega'|}} \tag{4.18}$$

  Where Ω' is the set of all $\boldsymbol{p} \in \Omega$ for which (4.17) is satisfied.

Of course, σ_{err} is not exactly the error distribution characteristic, but the mean value of $\mathcal{D}(\boldsymbol{p}) - \mathcal{D}^*(\boldsymbol{p})$ is significantly smaller than σ_{err}, hence the latter represents well the overall reconstruction precision.

Matching Techniques Validation The two techniques are tested on sequences of images of a plane on a white background, as it is shown in Fig. 4.16. The initial distance between the plane and the camera is 90 cm, the stereo base goes up to 120 cm. Such a stereo base is exaggerated to show the algorithms limits. The algorithm has been tested for 5 different camera trajectories (including rotation about different axes and translation) and for 5 different object orientations (25 tests in total). All the tests gave consistent results, that is why we do not show all the curves, but only a few of them. The two criteria, the inlier rate and the error standard deviation, are compared for the cases when the technique is applied and when it is not.

Complete Stereo Correspondence Testing Four different short trajectories of ten frames with a distance increment of about 2 cm have been generated. The scene is the same for all the sequences. Its layout is represented in Fig. C.9a. The camera moves in different directions with different rotation. The main advantage of synthetic data is that we have the depth ground truth $\mathcal{D}^*$, and also that we can isolate the algorithm performance from calibration uncertainties, lighting conditions and other real world factors. The base image and its depth map are given in Fig. C.7. The type of motion for each sequence is illustrated by the final images of the sequences, (the first column of Fig. C.8).

SGM is tested independently for each image with the first image as the base, no information is propagated from the previous iterations. On the other hand, the tracking stereo is initialized using the second image pair, which corresponds to a stereo base of ≈ 4 cm. Then, for the following images, the predictive matching strategy is used and the simple regularizer described above is applied.

[1] Technical details of image rendering are given in Appendix B

5.2 Matching Techniques validation

Dynamic Programming Matching Cost Below, we demonstrate the improvement of the results in terms of precision and accuracy due to the dynamic programming algorithm for the pixel matching cost computation. As illustrated by Fig. 4.18, all the curves start together and then diverge. It means that when the stereo base is small, the distortion of the descriptor along the epipolar curve is not large enough to show the distinction between the algorithms. Moreover, in case of noisy images, if λ_{flaw} is not well adjusted, the dynamic-programming version may give overfit, while the simpler version of the algorithm has some kind of internal regularizer. The bottom row of Fig. 4.18 shows the result for noisy images, and to attain this result we had to increase λ_{flaw}.

Correspondence Cost for Pixels For this test we have chosen to use a texture with a higher contrast, because the method shows a lot of improvement for image regions with strong brightness transition. Fig. 4.16 shows some test images, Fig. 4.17 shows the disparity computed for them. A simple visual analysis of it already gives an idea of the effectiveness of the cutoff cost. The inlier and error deviation plots for the illustrated sequence, as well as for another one, can be seen on Fig. 4.19. There is not much change in precision, as it was the case of the dynamic-programming descriptor alignment algorithm. On the other hand, the inlier rate changes a lot. There is a logical explanation to this fact. This cost does not improve the matching locally, within few pixels, but rather helps to avoid complete mismatches.

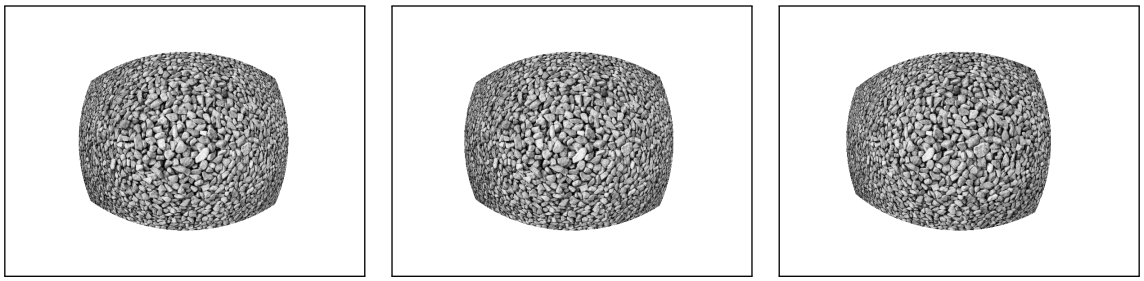


Figure 4.16: Synthetic images for basic tests of matching techniques (with photographic texture). From left to right: the base image, 0.1 m stereo base, and 0.5 m stereo base. The distance from camera to the plane is 0.9 m.

5.3 Results

Depth Map Analysis Fig. C.8 shows the last images from each sequence, as well as two reconstruction results for either method: for the stereo bases of ≈ 6 cm and ≈ 18 cm. For some depth maps certain areas are not filled up because for these areas the epipolar search for the given maximum disparity would exceed the image limits. The reconstruction succeeds no matter what the camera motion is. Yet for different kinds of motion we have different artifacts.

For example, in sequence (a) the camera moves to the right with no rotation. Because of the texture of the background wall, which has a lot of horizontal lines, which coincide with the epipolar lines, the depth is not regular for it.

In sequence (c), the camera moves forwards and turns right. The black circle in the middle corresponds to the epipole, which represents a singularity for the stereo reconstruction, so it is avoided with a certain neighborhood around it.

For SGM, the quantization effect is particularly visible. Since we have a discrete spectrum of possible disparity values, some discontinuities in the depth estimation are inevitable. For

the tracking method, the quantization effect is reduced because it integrates the information across multiple images and has a denser spectrum of possible depth values.

As the stereo base gets larger, the depth value spectrum gets denser and the reconstruction becomes smoother and more precise.

Criteria Analysis Fig. 4.23 shows the criteria as a function of the stereo base for the four data sequences. Two methods are compared: the prediction based matching and the SGM. You can see that both methods demonstrate a similar performance with no clear winner.

Apart from the values for the shortest stereo base, the curves are consistent. The inlier rate goes to a certain value and then stabilizes. It means that the uncertainty estimation approach described above is valid, given that the inlier rate stabilizes at 97–99%. As the stereo base gets wider, σ_{err} goes down, as it is approximately proportional to the inverted stereo base.

Real Data Tests The algorithm has been tested using a calibrated camera attached to a robotic arm. This setup allows us to know precisely the transformation between two camera poses. Fig. C.10 shows an example of the algorithm's result an image and its disparity map.

To quantify the result, 3D reconstruction of a planar object was performed and compared with ground truth (see Fig. C.11). To obtain the ground truth, the transformation between the robot's base and the plane was measured. Knowing the robot's pose it is possible to compute the transformation between the plane and the camera. Then, using the calibrated camera model, the ground truth for distance maps was generated for 6 different camera poses. The planar object does not cover the image entirely, so multiple datasets were acquired to test the approach in different parts of the images. To get the numeric results, disparity maps generated by the algorithm were transformed into distance maps; then the difference between the obtained maps and the ground truth was computed and analyzed. The pixels outside the planar object are ignored. All pixels, whose distance value is off by more than 100 mm, are considered as outliers and rejected.

Table 4.1 represents the error distribution of the distance reconstruction as a function of the stereo base. Results show that a larger stereo base consistently leads to a decrease in the mean and standard deviation of the error distribution. Table 4.2 shows the results for different datasets. Average distances from the camera to the plane are given to give a better idea about the reconstruction precision The error distribution is relatively narrow with respect to the actual distance values. Also, the inlier rate is high, which means that overall this algorithm is efficient and accurate in plane reconstruction.

Table 4.1: Plane reconstruction error as a function of the stereo base.

Stereo Base, mm	Mean Error, mm	σ_{error}, mm	Inliers, %
10	-7.50	23.4	97.8
15	-4.41	18.1	98.8
20	-4.07	15.4	98.7
25	-2.40	13.6	99.0
30	-1.79	12.4	99.1
35	-1.70	11.7	99.1
40	-0.98	11.2	99.0
45	0.09	10.1	98.7

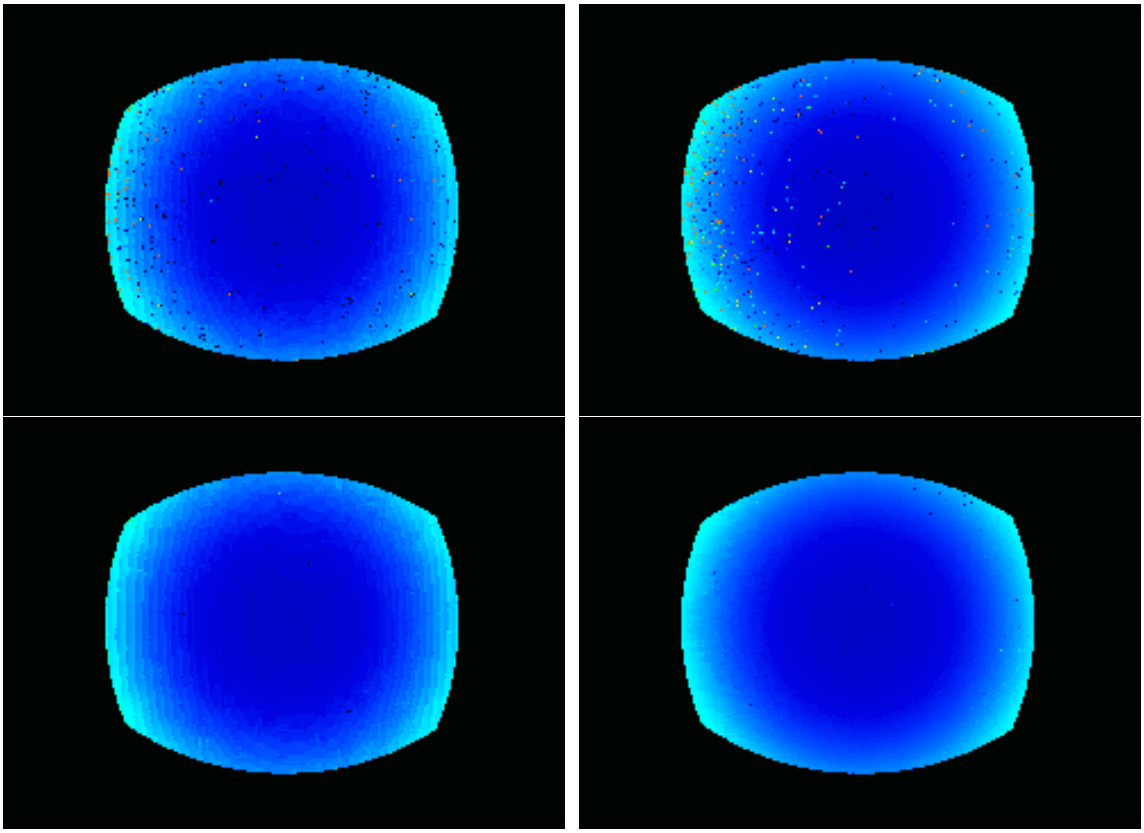


Figure 4.17: The first row — the computed depth for the simple absolute difference pixel matching cost. Second row — the depth computed using the cutoff cost. On the left the stereo base is 0.1 m, on the right 0.5 m. The cutoff cost reduces the number of strong outliers.

Table 4.2: Plane reconstruction error for different datasets. The stereo base is 35mm.

Dataset	Avg Distance, mm	Mean Error, mm	σ_{error}, mm	Inliers, %
1	552	1.00	15.0	99.3
2	381	-1.70	11.7	99.1
3	490	-0.83	15.1	99.7
4	423	-1.29	13.4	98.7
5	356	-2.42	14.1	99.5
6	433	1.61	22.0	98.4

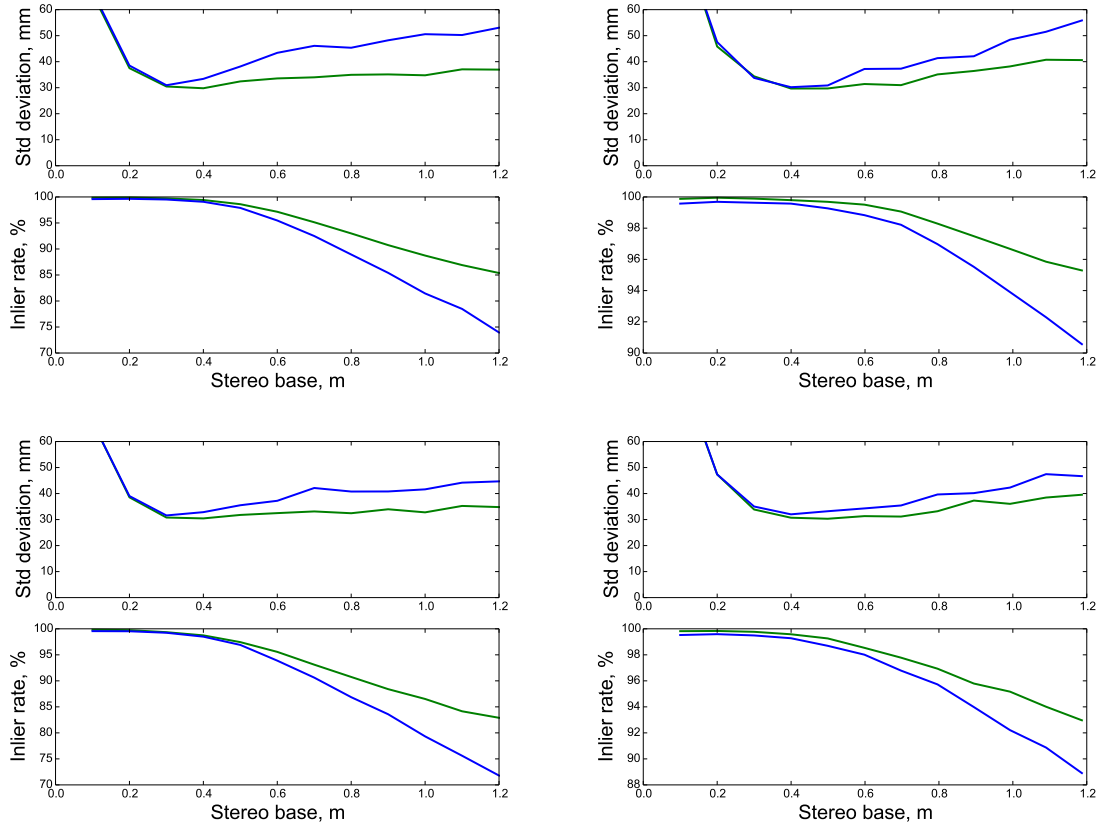


Figure 4.18: At the top: the standard deviation of the reconstruction error and the inlier rate for two different camera motions and initial object positions. The green curves represent the dynamic-programming-based matching, the blue curves represent the brute-force descriptor comparison. At the bottom: the same curves for images with uniform noise added. The dynamic programming improves both accuracy (represented by the inlier rate) and precision (represented by the error).

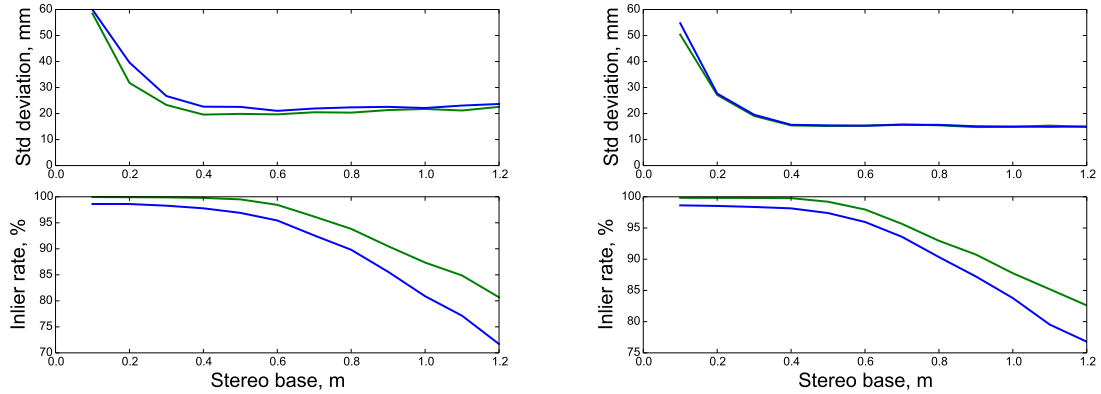


Figure 4.19: The experimental comparison of two ways of computing the pixel matching cost. The plots on the left and on the right correspond to two different camera trajectories. Green curve — cutoff cost, blue curve — absolute-difference cost. The precision remains almost the same, while the inlier rate significantly improves.

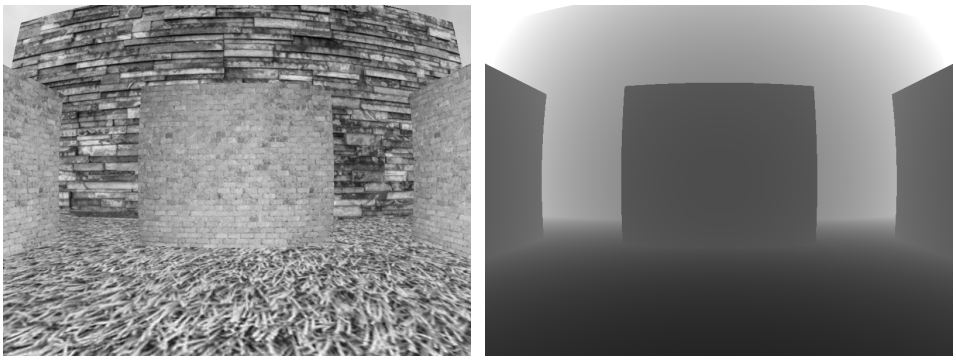


Figure 4.20: The base image and the reference depth map.

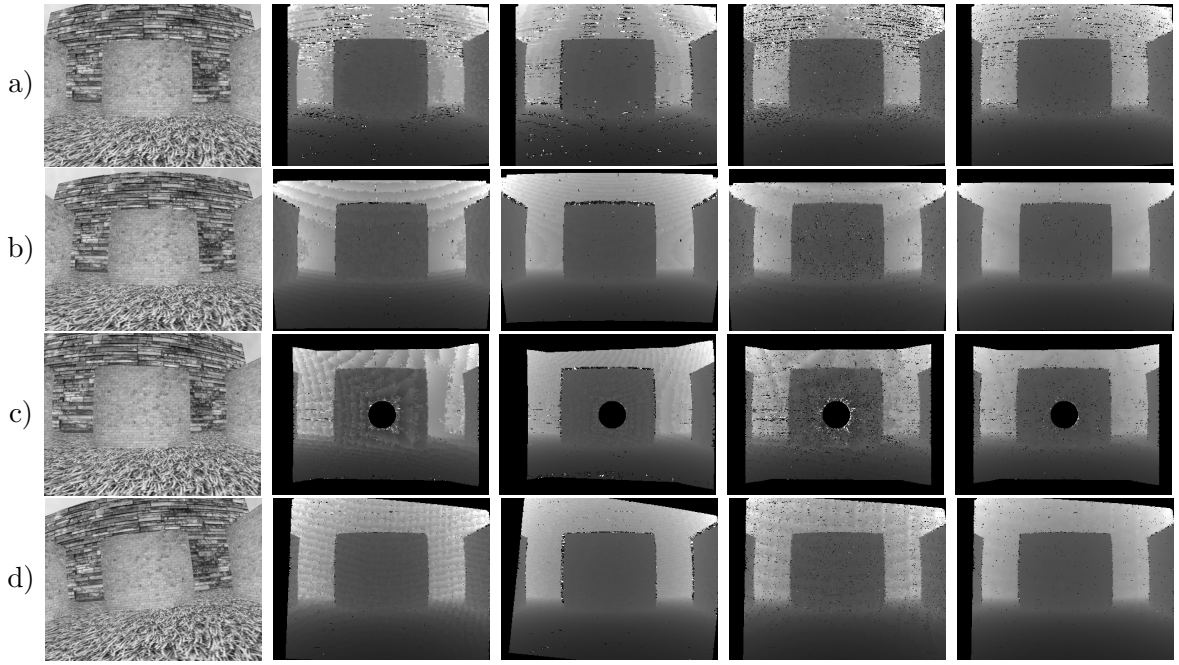


Figure 4.21: Examples of depth maps computed for this scene for different camera motions. The first column is the final image from the sequence; the second and third columns show the results of SGM; the fourth and fifth ones show the output of the predictive reconstruction algorithm. The second and fourth columns correspond to ≈ 8 cm stereo base, the third and fifth ones correspond to ≈ 20 cm stereo base. (a) camera moves to the right with no rotation. The background wall gives a lot of noise in the depth estimation because some features of its texture are parallel to the epipolar lines. (b) camera moves downwards and turns upwards, the background wall reconstruction is much more regular. (c) camera moves forwards and turns right, the black area on the border is due to the fact that, with a forward motion everything on the border disappears from the field of view, the black circle in the middle is the epipole with its neighborhood, which represent a reconstruction singularity. (d) camera moves rightwards and slightly forwards and turns about its optical axis. This is a demonstration of one of the most interesting advantages of the method, the camera rotation about the optical axis does not affect the reconstruction quality.

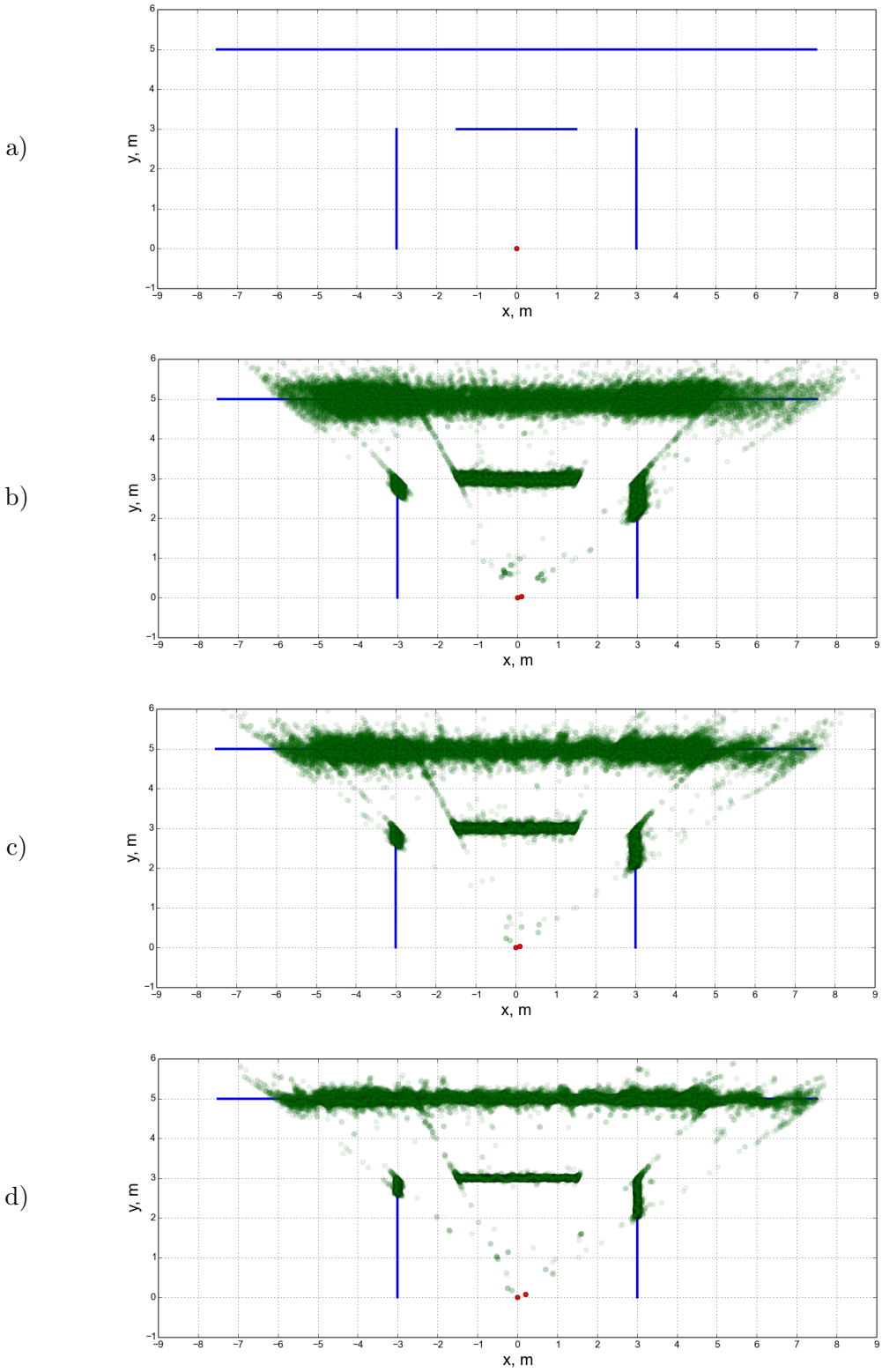


Figure 4.22: The reconstruction examples. In each case, the point cloud is truncated before being projected, so that the ground plane is not included. Sequence d from Fig. C.8 is used for the reconstruction. That is, the camera moves rightwards and slightly forwards, and rotates about its optical axis. The red dots represent the camera positions for the stereo computation. (a) The virtual scene layout with the initial camera position, the top view. (b) The reconstruction with SGM for a stereo base of about 10 cm; (c) the same configuration as (b), but this time the predictive matching is used. The result is slightly better, because the algorithm assimilates the information from all the previous steps; (d) the prediction-based reconstruction for the base of 20 cm.

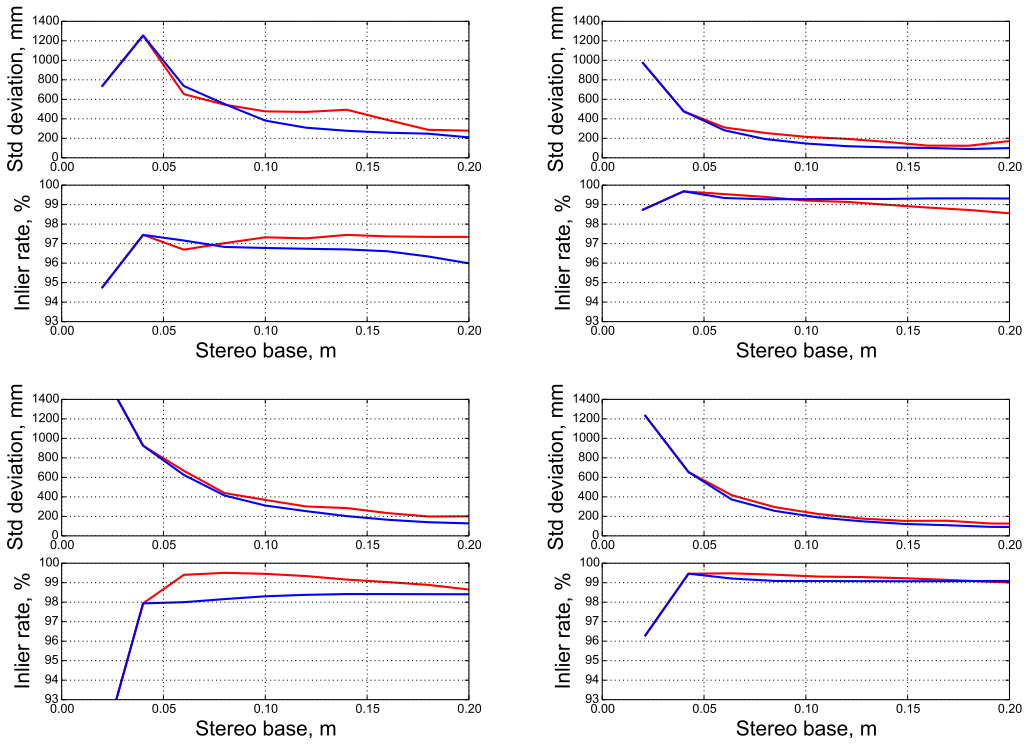


Figure 4.23: The error standard deviation and inlier's rates for four different datasets. Blue curves represent the prediction-based method, red curves represent SGM. Depending on data, the prediction based method may outperform the SGM.

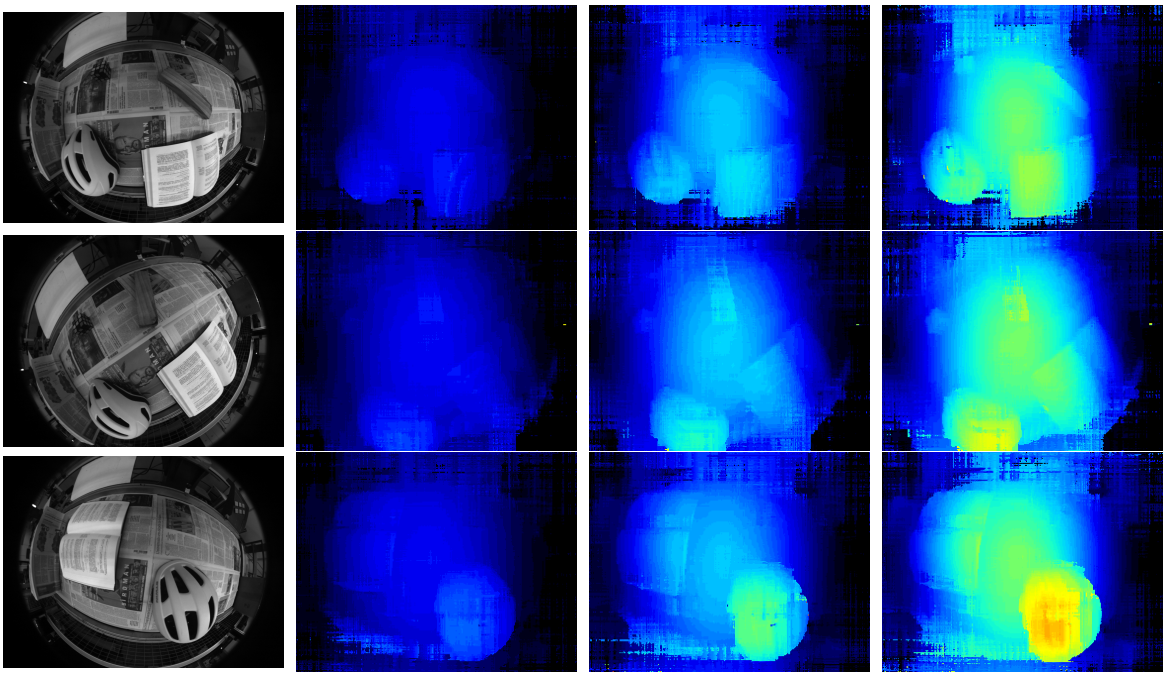


Figure 4.24: Original images (left) and the corresponding disparity maps (right) for 10, 20, 30 mm stereo bases.

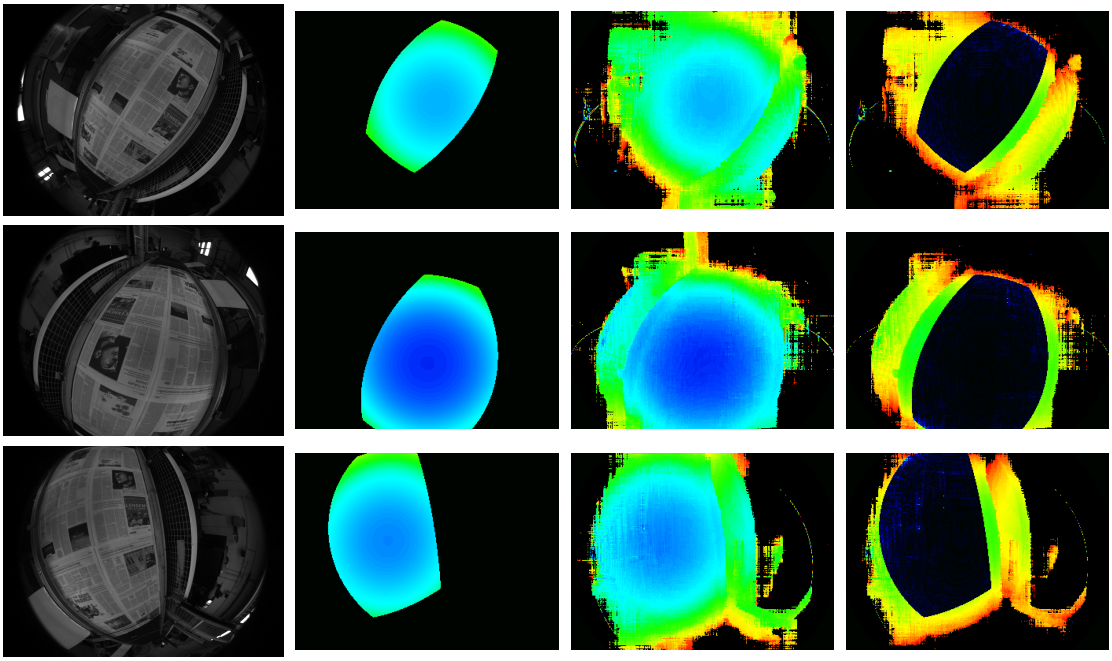


Figure 4.25: An illustration of the evaluation methodology. Each row corresponds to a different dataset (from the top: 3, 5, 6); *a* — the reference image; *b* — the ground truth of the newspaper plane distance map; *c* — the reconstruction result using disparity maps, which were computed using the camera's intrinsics and the robot poses as extrinsic parameters of stereo; *d* — the absolute difference between the ground truth and the reconstruction using the stereo.

6 Conclusions

The Enhanced Unified Camera Model has analytic properties which allow it to be used in modeling of stereo systems. The epipolar curve equations computed using the model can be used to compute the direct stereo correspondence, once they are properly rasterized. The disparity-based direct stereo correspondence gives an accurate 3D reconstruction along with a simple, and yet efficient method to estimate the reconstruction error. If we compare it to another 3D sensor — Velodyne, we see that it has some advantages and disadvantages.

The advantages of fisheye-based stereo correspondence:

- Better vertical resolution
- No temporal disparity in depth measurements caused by the sweep effects
- Much lower cost

The disadvantages of fisheye-based stereo correspondence:

- Lower precision, decreasing with distance
- Requires external light sources and textured environment

We argue that by integrating the stereo algorithm in a larger localization and mapping system we can attain the level of reconstruction precision necessary for safe and robust autonomous navigation. Some of the results presented in this chapter have been published in Khomutenko *et al.* (2016b).

6.1 Main Contributions

Semi-Global Matching Adaptation The results in geometric modeling, described in Chapter 2, allowed us to combine the Enhanced Unified Camera Model with the Semi-Global Matching. Analytic epipolar-curve equations are computed and then passed to the described rasterization algorithm with a special modification to work faster with implicit curves defined by a second-degree polynomial. The pixel-based sampling allows us to use the notion of disparity, which is the number of steps along a curve, and thus enables the Semi-Global Matching.

Matching Cost Computation We propose two methods to improve the matching results by improving the matching cost computation. The dynamic-programming algorithm for descriptor matching allows us to take into account the descriptor deformation due to the viewpoint change. The cutoff matching cost accounts for the difference in sample values due to strong gradients.

Epipolar Curve Precomputation Since one epipolar curve passes through multiple pixels, there is no need in computing one curve per pixel, but rather sample the epipolar curve space, which has only one dimension (rotation angle about the stereo base). We suggest a way to efficiently access precomputed curves without calling any non-linear functions. This operation is done for every pixel, so it is important to make it less computationally expensive.

6.2 Future Development

Algorithm Adjustment Modifying the following aspects of the algorithm may improve its performance. A different number of regularization directions should be tested. The actual implementation uses only two of them. Other numeric parameters of the algorithm, like dynamic programming step cost and jump cost must be adjusted according to the image noise. During the descriptor matching step, it may be interesting to apply certain weights to the descriptor value to make the central pixel more important for the matching. To adjust the descriptor scale, a rigorous criterion based on the frequency content of the image patch should be used.

Disparity Postprocessing So far, only a basic depth map postprocessing has been done, that is the local regularization. But the community has already developed different techniques to postprocess disparity maps. For example, removing speckle noise by detecting small "islands" of disparity which don't match their surrounding. Another aspect is treating textureless areas. Techniques for solving both of these issues have already been implemented and tested (Hirschmuller (2008)).

Parallelization The algorithm is fully parallelizable, hence an efficient GPGPU-based implementation is possible for both matching cost calculation and Semi-Global regularization (for the latter, see Zhu *et al.* (2012)).

Temporal Filtering In theory it is possible to detect mismatches based on the matching cost. We can keep pixel matching cost at the previous iteration and reject a match if its matching cost is much higher than the previous time. The underlying hypothesis is that, for a given camera, the noise level should remain the same and the matching cost for a certain pixel may change, but not too much, and at least it should do it continuously. So we can try to build an adaptive cost model for each individual pixel with a prior estimation based again on the pixel's neighborhood frequency content.

Explicit Occlusion Treatment So far the algorithm is looking for a correspondence for every pixel in the base image. It is possible to modify the Semi-Global regularizer to add the option "occluded" for pixel matching. Like this the algorithm will be able to automatically detect occluded areas. The problem here is to adjust the cost, because the "no match" option must be more expensive than actual match

Different Sampling Patterns The actual depth sampling pattern is a regular rectangular grid. It is not necessarily the best choice. A hexagonal grid may give interesting results. Another option is an adaptive pseudo-random sampling pattern, which can greatly improve the performance as it would require much fewer samples. On the other hand, the application of the dynamic-programming regularizer becomes a non-trivial problem for irregular grids.

6.3 Applications

This algorithm can be applied when we need a wide-angle 3D reconstruction. The most appealing application is visual perception systems for mobile robots and autonomous vehicles.

Localization and Navigation 3D perception allows us to build up localization systems, such as visual odometry or visual SLAM (Meilland *et al.* (2015); Caruso *et al.* (2015)). A slight extension of these localization and mapping systems can lead to a navigation system, that is a system capable of planning the path and guiding the vehicle from location A to location B.

Obstacle Avoidance and Pedestrian Detection Depth sensors are crucial in obstacle detection. In contrast with laser scanners, visual 3D perception sees obstacles at all possible heights, not only at the level of the sensor, which may be important in some cases. It can be used to detect both static and moving obstacles (for example, parked or moving cars). As this system provides a wide-angle 3D perception, it can be used to detect pedestrians which are about to cross the street.

Object Manipulation Another interesting application is a perception system for robot manipulators for object detection, manipulation and recognition. Usually robot manipulators have a highly precise relative localization of the end-effector. If a camera is attached to it, it means that the robot can do two tasks with a so-called eye-in-hand configuration: provide the stereo extrinsic calibration for a 3D reconstruction and manipulate the objects.

Chapter 5

Vision-Based Localization and Mapping

1 Introduction

This chapter is dedicated to different methods of vision-based localization for mobile robots. A fisheye camera and wheel odometry constitute the necessary setup. First some necessary general concepts of mapping and localization are reviewed. Then we consider the classical feature-based visual odometry and the bundle adjustment algorithm. Then an alternative visual odometry with wheel odometry prior is described. Finally, two methods of direct image registration analyzed. One of them, photometric-based, is used to register consecutive frames from the same video sequence. The other, based on the concept called Mutual Information, is used to register two images of the same place, taken in two different moments of time.

1.1 Different Maps and Localization Algorithms

Depending on application, available sensors and structure and properties of the environment, different kinds of maps and localization algorithms can be used. Obviously, exteroceptive sensors must be used for the mapping and localization. Let us discuss which options we have.

Topological Maps If the map is represented by a set of abstract locations connected with edges which represent a possibility to go from one location to another (Fig. 5.1), then the localization is called "topological". Examples: in which room are you? in which street are you? in which city are you? Only topological localization is not sufficient for many robotic tasks, so it has to be combined with some other localization systems. But if the robot knows in which location it currently is, it can upload a map of the location and apply a local metric localization, which we will talk about in a moment.

You may notice that the global localization and topological localization are doing somehow similar jobs. Indeed, a robot can find out in which street it is by checking its global coordinates. on the other hand, when the robot enters a building, GPS is not operational anymore and it has to rely on other sensory modalities to localize itself.

The most common topological localization approach, called FAB-MAP and described in Cummins and Newman (2008), is based on vision and visual features. The concept of features is described later on. By now, let us say that features are some, generally small, image areas which are easily detectable and recognizable. Given an image of a place to be localized, the system is looking for visual features, and then recalls which places with similar visual features it has seen so far.

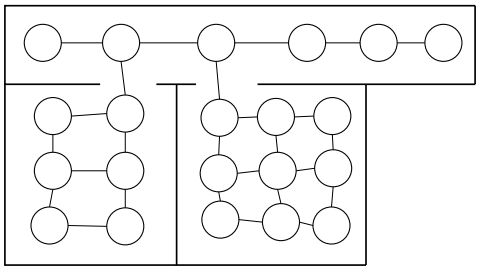


Figure 5.1: A metric map which represents walls, with a topological map overlaid, represented by circles and possible transitions. Since topological maps are usually constructed automatically, they are not always regular.

Once candidates are found, the geometric consistency can be checked (how to do it is also described later). If there is a place which has a set of features similar to the observed ones, and these features are geometrically consistent, then localization is considered to be successful.

Local Metric Maps The surroundings can be described directly in terms of 3D space. They can be represented by points or other geometric primitives, all put in the same *local map frame*. For example a map can be represented by a point cloud, which is a common choice for 3D sensors, like Velodyne. Another popular option is the so-called *occupancy grid*, a bitmap which represents free space and occupied space. The occupancy grid is usually used with 2D laser scanners (for example Wang *et al.* (2007)).

These kinds of maps are the natural choice for active metric sensors, like lidars. A single lidar scan (image) already can be considered as a local metric map, and by combining multiple scans, it is possible to construct a precise and detailed map.

An efficient algorithm to process point clouds, called ICP, is usually used to register different scans. Registration means bringing two datasets to the same reference frame. In other words registering two scans means "aligning" them, or finding a transformation between the origins of both scans. ICP works for both 2D and 3D scans. After registration, the map can be extended. It is done either by adding more primitives to the map, that is, points, lines, planes; or by changing values in the occupancy grid.

Sensory Maps Another way of representing maps is to keep the data in the sensory space. For example, in the case of laser scanners, the actual output is not directly a point cloud but rather a function which maps the angle to a distance value. This representation can be used for a map as well. In Biber and Duckett (2005) this representation has been used for an environment which is changing over time.

But for some sensors, in particular cameras, where the sensory output does not contain any geometric data, it may be the best representation. Let us talk about it in more details. An image is a mapping which associates a brightness value to each direction associated with pixels. To extract geometric data, as it has been discussed in Chapter 2, we need to process multiple images, establish correspondences and solve a reconstruction problem. Finally, we can get something similar to what laser scanners provide, that is, a correspondence direction-distance. But there is an important difference between these two geometric measures, which consists in differences in measurement noises. In the case of lidars, the distance measurement noise does not depend on the distance, but only on the sensor design. On the other hand, vision-based 3D reconstruction gives a non-uniform noise, which depends on the depth value, the precision of the transformation between the two camera positions, and even on the texture of the image region in question.

In the ideal case, when there is no noise, all the representations are equivalent. But in the presence of noise, it is generally better to keep the representation as close to the actual perception as possible.

An example of sensory localization is described in Courbon *et al.* (2009), where visual paths are represented by sequences of key images. Other examples are the recent direct visual SLAM systems Meilland *et al.* (2015); Engel *et al.* (2014), where the map is represented by a network of keyframes with depth estimation and photometric information, associated to each of them. We can go about the localization within this map as follows. First assume that we know which of the keyframes is closest to our actual position. This assumption is not too strong because either we know where we start the localization or we can try to localize ourselves globally by using, for instance, FAB-MAP (Cummins and Newman (2008)). Via the robot's motion we can observe the depth of a certain set of image points (either a dense depth estimation as it has been described in Chapter 2, or the depth for a number of keypoints). Then it is possible to compute the transformation between the keyframe and the current camera pose. The most part of this chapter is dedicated to the question "How to compute it?". Then eventually the keyframe content may be updated or the robot may switch to the following keyframe.

Odometry is a particular type of localization which is based on integration of motion increments. It can be based on either proprioceptive (IMU or wheel rotation in the case of wheeled robots) or exteroceptive sensors — cameras. In both cases the particularity of this method is that the localization it gives is consistent only locally. Since every increment has an error, integration over the whole path leads to error accumulation and divergence between the real trajectory and the estimated one.

Both wheel and visual odometries are described later in this chapter. For now, let us say that odometry, especially the proprioception-based one, is an essential perception modality for mobile robots. Since it is locally consistent and quite precise, it allows the system to accurately predict its current position since the last exteroceptive input, which, on its turn, improves the localization system robustness and accuracy. Indeed, the job of a map-based localization in this case is just to correct the small incremental error, which is usually proportional to the traveled distance and is in the order of a few percents of it, instead of looking across the whole area where the robot could have got during the elapsed time since the last localization iteration.

Mapping Mapping is an essential function of perception. As it has been stated above, a map is a representation of the environment. Whatever an agent is doing, it requires a certain kind of map. If it is navigating through space, it has to know where the obstacles are, and where the free space is. If the robot is a manipulator, it needs to know where the obstacles are and where the objects to be manipulated are. Whenever it comes to the question "where?", the map is what is needed.

The exact way of constructing maps depends on the kind of map we need. If we knew the robot's motion precisely, reconstruction of the map would be a simple task, and vice versa, knowing the map, localizing the robot is a simple task. But in the case of SLAM, we have to estimate both things simultaneously.

At the beginning of the process, the environment is completely unknown. The robot, through its motion, observes the surroundings from different positions, that is, it gets a set of noisy observations. The problem is formulated using the probabilistic framework (for example Wang *et al.* (2007)): find the map and the trajectory which together maximize the probability of observations. Usually, solving hard optimization problems requires a good initial approximation, which can be obtained using proprioceptive sensors.

Sometimes some prior knowledge about the map is added. That is, we have some models of the environment. An example of a prior model is the stereo correspondence regularization, described in Chapter 2. Another example is to assume that the environment is a set of flat surfaces, which is generally not true, but can be a valid model for some special cases. Any prior knowledge simplifies the mapping problem. Humans have a lot of it, so for us building local maps is almost trivial. We operate with such landmarks as furniture, plants, buildings and so on. But integrating such priors into robotic mapping systems is a huge research subject on its own.

1.2 Visual Localization

The mathematical aspects of visual localization are considered in the corresponding sections of this chapter. Here, let us just discuss certain conceptual aspects of visual localization. It is a hard problem, because instead of measuring the geometry of the environment directly, cameras provide data which are an entangled combination of camera parameters, environment geometry, textures of surrounding objects, light sources.

Even the direct problem, that is, given the complete environment state, simulate the image, is computationally expensive if we want to take into account all the effects, like reflections, shadows, speckle light etc. And still it is not possible to generate photo-realistic images in real time. So, in order to use vision for localization, a certain number of assumptions has to be made. Depending on exact approach, these assumptions may be different, and are described in corresponding sections.

From Brightness to Geometry Usually, the image-based localization is done in two steps:

1. Extract geometric information from the image.
2. Solve a geometric-vision problem.

We should notice that with recent, so-called direct methods, this pipeline changes. The bottleneck of this approach is the first step. What kind of geometric information do we want? Usually, it is the correspondence between image points. Suppose that we have two images. If we knew for sure what are the corresponding points in the second image for a certain number of points in the first image, then we could already estimate the camera motion between the two images with the translation known up to a scale factor.

Another kind of geometric information that we might want to get is the distance from the camera to every observed points. Given the distances, we can use another method to compute the camera motion, or some other geometric characteristics of the environment.

Short-Term Localization If we want to localize the robot in a certain place within a few minutes, then we usually make following assumptions:

- The lighting conditions do not change. That is, the sun remains in the same place, the artificial light does not go off or on, shadows do not move and so on.
- Static objects in the map likely do not move. For example chairs, tables, doors remain as they were at the beginning.

These assumptions are obviously false, the sun moves, a small cloud may change the lighting a lot within a few seconds, and the doors and other furniture change position and state. Sure, but even under these assumptions the problem is not easy to solve.

There is an even stronger set of assumptions, called *static environments*. It includes the fact that there are no moving objects around. In this case a pure vision-based localization

system can be implemented. But if not all the objects in the environment are static, then a general vision based localization may fail badly. A simple example, a train passing in front of a robot and covering almost the whole field of view may make the visual localization system think that it is actually the robot which is moving, not the train. A few more words will be said about this issue in the discussion of multi-modal perception.

Long-Term Localization What if the robot has to visit again a place it has visited before and has constructed a map of. The map has been based on a particular lighting condition and a layout of movable objects. Both may have changed since the last time. Is the map still reusable? If not all the landmarks and features in it have been destroyed, then yes. But as the changes in the environment are more significant than ones that occur within few minutes, the localization algorithm must explicitly include possible changes into the model.

One method is to construct multiple sensory maps, as has been done in Churchill and Newman (2013). Every time the robot is revisiting a place, it tries to localize itself with respect to all the maps of this place that it has constructed previously. If none of them give consistent results, a new map is instantiated. Each map is called "experience" and represents a different possible appearance of the place.

Another possible approach in the context of sensory maps is to use a similarity measure to compare and register map frames and current images, which is robust with respect to such changes. The so-called *mutual information* can be such a measure. It has been successfully used for image tracking and vision-based control of mobile robots to make them follow a path similar to one which is represented by a sensory map, that is, a sequence of images (Dame and Marchand (2012); Raj Bista *et al.* (2016b)).

1.3 Visual Features

Here we provide an overview of point-like visual features, which are a recipe for solving non-local problems of computer vision. That is, for a particular point in one image we do not have any guess where the corresponding point is in the other image.

There are other types of visual features, like lines, curves, ellipses and so on. But point features are the simplest to work with.

1.3.1 Feature Detection

A keypoint or feature point is a point in an image which is well-localized, in the sense that there exists a function of the image, such that the point coordinates correspond to an extremum of this function and small noise in the image leads to small changes in the coordinates of the extremum. Such function is called *response function*; extrema of this function correspond to the feature points. This function should be designed in a such way that if the image undergoes some transformations (for example, we take the picture of the same object but from a different prospective) extrema of this function will still correspond to the projection of the same 3D point. Usual types of well-localized points are corners and blobs (that is, small textured area different in some way from the background). In contrast, points on edges or inside uniform regions are badly localized. It is called *aperture problem*. In the first case we may have difficulty with localization along edge direction, in the second, completely uncertain position.

Blob Detectors Common blob detectors are SIFT, Lowe (2004), and its modification SURF, Bay *et al.* (2008). The concept of blob detection is that blobs are brighter or darker than the background. The response function is the difference of Gaussians, which approxi-

mates Laplacian of Gaussian, in the case of SIFT, while SURF approximates it with so-called box filters to speed up the computation.

Harris Corner Detector Here we would like to give a more detailed overview of another classical detector by Harris and Stephens (1988). It is mathematically elegant and demonstrates well the concept of feature detection. Actually this detector responds not only to corner-like features, but to any image patch which is not invariant with respect to translation. To detect a corner we must figure out how much the original patch that surrounds a point differs from a shifted version of itself, which can be measured using the sum of squared differences (SSD):

$$S(\zeta, \eta) = \sum_{u,v} w(u, v)(I(u + \zeta, v + \eta) - I(u, v))^2 \tag{5.1}$$

Where $I(u, v)$ is an image value that has coordinates u, v; w is a weight function to select just the region of interest around a point, ζ and η are shift parameters. We can approximate the shift of an image by a Taylor expansion:

$$I(u + \zeta, v + \eta) - I(u, v) = I_u(u, v)\zeta + I_v(u, v)\eta \tag{5.2}$$

where I_u denotes the partial derivative of the image with respect to u, the same for I_v. And then the SSD criterion can be written in matrix form:

$$S(\zeta, \eta) = \begin{pmatrix} \zeta & \eta \end{pmatrix} A \begin{pmatrix} \zeta \\ \eta \end{pmatrix} \tag{5.3}$$

where A is called *structure tensor*:

$$A = \sum_{u,v} w(u, v) \begin{pmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{pmatrix} \tag{5.4}$$

If both eigenvalues of matrix A are small, then the patch is almost invariant to a shift, it is almost homogeneous. If just one eigenvalue is large enough, it means that the patch is an edge and it is invariant with respect to translation along the edge direction. If both eigenvalues of A are large and positive, it means that a corner is detected. The Harris response function is:

$$M_c = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det A - \kappa \operatorname{trace}^2 A \tag{5.5}$$

Where λ_1 and λ_2 are the eigenvalues of A, and κ is a coefficient that can be tuned.

Another possible response function is :

$$M_c = \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)} = \frac{\det A}{\operatorname{trace} A} \tag{5.6}$$

It is always less than the smallest λ, so it provides information on the lower boundary of the worst direction in terms of stability.

FAST Corner Detector Another popular corner detector is called FAST (Rosten and Drummond (2006)). Its idea is to compare every pixel, a potential corner, to pixels which lie on a small circle around it (of radius about 3 px). A certain number of contiguous pixels among the pixels on the circle must be either darker than the central pixel minus threshold t, or brighter than the central pixel plus t. In this case the central pixel is classified as a corner. If there are many neighbor pixels which are classified as corners, the one which has the greatest t is kept, and the rest is discarded. It is called *non-maximum suppression*. The advantage of FAST is that there is an efficient way of rejecting candidates without checking all the 16 pixels on the circle. Modifications of this detector are used in such detectors as ORB by Rublee *et al.* (2011) and BRISK by Leutenegger *et al.* (2011).

1.3.2 Feature Descriptors and Matching

The idea behind the feature descriptors is that we want to encode the neighborhood of the feature point, in order to be able to compare it to other detected feature points and tell which one of them is the most likely match for the given point. Descriptors of point-like features may have some of the following properties:

- Scale invariance. The keypoint is detected on two images with different zoom (for example, because of changed distance to the object), but the descriptor remains the same (or nearly the same). Scale invariance is provided by looking for features not in the original image but in scale-space. Scale-space is a 3D space with coordinates (u, v, σ). σ corresponds to the degree of Gaussian blur. From the information point of view, it is the same as scaling images down.

- Rotation invariance. Rotation invariance with respect to the optical axis can be achieved by choosing the dominant orientation (or feature direction) of the feature and prerotating the feature neighborhood to align it with the dominant direction before constructing the descriptor.

- Full affine invariance. This can be done by constructing the Hessian matrix, computing its eigenvectors and rectifying the region with respect to these directions.

Actually, for the purposes of visual odometry, there is no need for rotation invariance, since we do not expect the robot to perform significant rotations about the optical axis. But even if it did, the rotation done between two consecutive images should be within robustness limits of the descriptor.

The most basic descriptor for a point $\boldsymbol{p}$ is a weighted patch. Let $\Omega_{\mathfrak{D}} \subset \mathbb{R}^2$ be a set of points which define the neighborhood, then let $\mathfrak{D} : \Omega_{\mathfrak{D}} \to \mathbb{R}$ be a descriptor Usually, $\Omega_{\mathfrak{D}}$ is just a square centered at the origin. Let $w : \mathbb{R}^2 \to \mathbb{R}_+$ be a weight function. Then the descriptor $\mathfrak{D}_{\boldsymbol{p}}$ of a point $\boldsymbol{p}$ is defined as follows:

$$\mathfrak{D}_{\boldsymbol{p}}(\boldsymbol{q}) = w(\boldsymbol{q}) I(\boldsymbol{p} + \boldsymbol{q}) \qquad \boldsymbol{q} \in \Omega_{\mathfrak{D}} \tag{5.7}$$

To provide invariance with respect to the exposure and light intensity, the descriptor can be normalized:

$$\mathfrak{D}_{\boldsymbol{p}}(\boldsymbol{q}) = w(\boldsymbol{q}) \frac{I(\boldsymbol{p} + \boldsymbol{q}) - \bar{I}(\boldsymbol{p})}{\sigma_I(\boldsymbol{p})} \qquad \boldsymbol{q} \in \Omega_{\mathfrak{D}} \tag{5.8}$$

where $\bar{I}(\boldsymbol{p})$ and $\sigma_I(\boldsymbol{p})$ are normalization terms:

$$\begin{aligned} \bar{I}(\boldsymbol{p}) &= \frac{\sum\limits_{\boldsymbol{q} \in \Omega_{\mathfrak{D}}} w(\boldsymbol{q}) I(\boldsymbol{p} + \boldsymbol{q})}{\sum\limits_{\boldsymbol{q} \in \Omega_{\mathfrak{D}}} w(\boldsymbol{q})} \\ \sigma_I(\boldsymbol{p}) &= \sqrt{\frac{\sum\limits_{\boldsymbol{q} \in \Omega_{\mathfrak{D}}} w(\boldsymbol{q}) \left(I(\boldsymbol{p} + \boldsymbol{q}) - \bar{I}(\boldsymbol{p}) \right)^2}{\sum\limits_{\boldsymbol{q} \in \Omega_{\mathfrak{D}}} w(\boldsymbol{q})}} \end{aligned} \tag{5.9}$$

Usually, w decreases rapidly with the norm of its argument. It is usually a Gaussian kernel:

$$w(\boldsymbol{q}) = \exp\left(\frac{\|\boldsymbol{q}\|^2}{2\sigma^2} \right) \tag{5.10}$$

Another type of descriptors has been proposed in Calonder *et al.* (2010) and modified in Leutenegger *et al.* (2011). The keypoint neighborhood is sampled according to a pseudo-random pattern. Samples are compared according to the same pregenerated pattern, and the result of the comparison is stored as a binary descriptor. Another kind of distance measure, called Hamming distance, must be used for these descriptors. It counts how many bits are different in two descriptors.

To match two sets of descriptors, usually for each descriptor in the first set we are looking for the closest descriptor in the other set. Then we repeat the procedure in the opposite direction. If two descriptors "choose" each other, then the match is considered as successful. Also different thresholds can be set up to automatically reject matches of descriptors which are too far away.

It is impossible to guarantee exact matching between two keypoint sets, based purely on descriptors. Firstly, because of viewpoint changes, image discretization and noise, descriptors of the same keypoint in two different images will be slightly different. Secondly, especially in artificial environment, there are plenty of repetitive structures, in which we can find multiple keypoints with similar neighborhoods. That is why geometric consistency must be checked to ensure that the matches are correct.

1.4 Feature-Based Visual Odometry

Visual odometry consists in computing a camera's motion between two frames. At first, we assume that we have N point-like features detected in both images and perfectly matched. The question is how to use this information to compute the motion. Then we take into account mismatches between features, which generate outliers, that is, data samples which do not follow the presumed noise distribution. Finally, we explain how to integrate the wheel odometry data into visual odometry to improve its accuracy and robustness.

Since cameras do not provide any information about the distance, the translation can be computed only up to a scale factor. It means that if we want to reconstruct a path based only on visual odometry, we shall use some special technique to avoid so-called scale-factor drift. In this work, the problem is addressed by simply using the wheel odometry to correct the scale.

1.4.1 Motion Estimation for Pinhole Cameras, Closed-Form Solution

Motion estimation can be done by decomposing the essential matrix E, introduced in Chapter 1. This matrix, as well as the fundamental matrix F can be estimated as described below. We would argue that this method, though mathematically elegant, has a number of drawbacks, for example it requires the camera to follow the pinhole projection model. On the other hand optimization-based techniques are well developed and contemporary non-linear solvers run fast and don't represent the algorithm bottleneck anymore.

Estimation of the Fundamental matrix The following methodology has been overviewed in Szeliski (2010). Estimation techniques for F and E are similar, but generally matrix F is more practical, because it does not require the conversion of the image points into points on the normalized plane (that is, $\boldsymbol{p}$ to $\boldsymbol{m}$). On the other hand, F has meaning only in the case of pinhole cameras. For cameras with distortion, E is the only option. To estimate F we can develop (2.76) rewriting it in the scalar form:

$$\begin{array}{ccccccc} u_0u_1f_{11} & + & u_0v_1f_{12} & + & u_0f_{13} & + & \\ v_0u_1f_{21} & + & v_0v_1f_{22} & + & u_0f_{23} & + & \\ v_1f_{31} & + & v_1f_{32} & + & f_{33} & = & 0 \end{array} \tag{5.11}$$

Hence, we have the following equation, linear with respect to the components of matrix F:

$$A\boldsymbol{f} = 0 \tag{5.12}$$

$$A = \begin{pmatrix} u_0^1 u_1^1 & u_0^1 v_1^1 & u_0^1 & v_0^1 u_1^1 & v_0^1 v_1^1 & u_0^1 & v_1^1 & v_1^1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_0^n u_1^n & u_0^n v_1^n & u_0^n & v_0^n u_1^n & v_0^n v_1^n & u_0^n & v_1^n & v_1^n & 1 \end{pmatrix} \tag{5.13}$$

$$\boldsymbol{f} = \begin{pmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{pmatrix}^T \tag{5.14}$$

where upper index of u_0^j, v_0^j stands for the feature number. Since this equation is defined up to arbitrary scale, we need $n = 8$ equations to estimate the f_{ij} coefficients (by fixing $f_{33} = 1$). If we have more points, we can estimate F in a more precise way from a statistical point of view. To do this we must construct matrix A and perform its singular value decomposition (SVD). The right singular vector that corresponds to the smallest singular value is the estimation of $\boldsymbol{f}$. Because of noises and errors, matrix A, which in theory is of rank 8, may be full column rank. If we know the uncertainties in the feature positions, we can weight the equations with respect to the reciprocal of them. That is, use $B = \mathcal{C}^{-1}A$ instead of A in (5.11), where $\mathcal{C}$ is a diagonal matrix with elements representing the variance of feature locations.

Another issue is that if u, v are large, which is usually the case since they are measured in pixels, then, for example, uncertainty in u_0 will be amplified by u_1 in term $u_0^j u_1^j$ of A, while in term u_0^1 uncertainty will stay the same. It is better to normalize all points $\boldsymbol{p}$ for each image using a linear transformation so that the mean of $\boldsymbol{p}$ be 0 and the standard deviation along each axis be 1. This transformation is:

$$T = \begin{pmatrix} \frac{1}{\sigma_u} & 0 & -\frac{\mu_u}{\sigma_u} \\ 0 & \frac{1}{\sigma_v} & -\frac{\mu_v}{\sigma_v} \\ 0 & 0 & 1 \end{pmatrix} \tag{5.15}$$

where μ_u, μ_v are the means of u and v respectively, while σ_u, σ_v are their standard deviations. If $\tilde{x}_i = T_i x_i$ then (2.76) becomes:

$$\tilde{\boldsymbol{x}}_{proj0}^T T_0^{-T} F T_1^{-1} \tilde{\boldsymbol{x}}_{proj1} = 0 \tag{5.16}$$

We can denote $\tilde{F} = T_0^{-T} F T_1^{-1}$, estimate it using normalized points and then recover original $F = T_0^T \tilde{F} T_1$

Equations (2.71) and (2.76) can be used to discard outliers. If the left part of the equation is larger than some threshold, it means that the feature is badly localized or the matching is wrong.

In case of pure rotation, E and F are impossible to estimate. If such a case may appear (for example, for some wheel configurations pure rotation is virtually impossible), it is a good idea to try to compute the rotation only, assuming that the translation is zero. Then compute the residual and only if it is non negligible, compute matrix E.

Motion Extraction from the Essential Matrix The rotation and translation part (up to a scale factor) can be estimated from matrix E using SVD. If the camera is calibrated, then E can be easily calculated from F as $E = K^T F K$, where K is the camera projection matrix. Here we assume that the camera is the same in both camera positions (generally this technique can be applied for two different cameras with different K matrices, but for visual odometry application, the case with one camera is of greater interest).

To estimate the direction $\hat{\boldsymbol{t}}$ we may notice that $E = [\hat{\boldsymbol{t}}]_\times \boldsymbol{R}$ and $\hat{\boldsymbol{t}}^T E = 0$. So, applying SVD we obtain:

$$E = [\hat{\boldsymbol{t}}]_\times \boldsymbol{R} = U\Sigma V^T = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \hat{\boldsymbol{t}} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \boldsymbol{v}_3^T \end{pmatrix} \tag{5.17}$$

The translation direction corresponds to the left singular vector with the least singular value. Two other singular values must be similar, but because E is known up to scale, they may be different from 1. Exploiting the fact that $[\hat{\boldsymbol{t}}]_\times$ projects the vector onto the plane orthogonal to $\boldsymbol{t}$ and rotates it by $\frac{\pi}{2}$ we can rewrite it as:

$$[\hat{\boldsymbol{t}}]_\times = UZR_{\pi/2}U^T = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \hat{\boldsymbol{t}} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1^T \\ \boldsymbol{u}_2^T \\ \hat{\boldsymbol{t}}^T \end{pmatrix} \tag{5.18}$$

That is, we project the vector into basis U, rotate it around $\boldsymbol{t}$ and zero this component, then we go back to the old basis. Now we can restore the $\boldsymbol{R}$ matrix:

$$E = UZR_{\pi/2}U^T\boldsymbol{R} = U\Sigma V^T \quad \Longrightarrow \quad \boldsymbol{R} = UR_{\pi/2}^T V^T \tag{5.19}$$

assuming that $Z = \Sigma$ as it should be in theory. Unfortunately we know E and $\hat{\boldsymbol{t}}$ up to sign. It means that we have to try 4 different matrices $R = \pm UR_{\pm\pi/2}^T V^T$ and keep the two that have the determinant $det(\boldsymbol{R})$ equal to 1. To choose the correct one we must try all the combinations $\pm\hat{\boldsymbol{t}}$ and $\boldsymbol{R}_{1,2}$ and take the one that provides maximum points in the field of view for both cameras.

1.4.2 Bundle Adjustment

Bundle Adjustment (BA) is a problem of reconstruction of a 3D environment and camera's positions by minimizing the error between modeled projections of 3D primitives and their detected projections. This problem is solved by means of global nonlinear optimization. It is a better solution to estimate the motion in a statistically correct way using redundant measurements, than the estimation of E and its decomposition. Also, it is a basic method to construct metric maps with point features (see for example Beall and Dellaert (2014)).

Assume that we have N 3D points and M positions, for each of one them we grab an image with a calibrated camera.

- Let $\{\boldsymbol{X}_j\} \subset \mathbb{R}^3$ be a set of observed 3D landmarks.
- Let $\{\xi_i\} \subset \mathrm{SE}(3)$ be a set of pose parametrization.
- Let $\{\boldsymbol{p}_{ij}\} \subset \mathbb{R}^2$ be the set of projections of the j-th landmark onto the i-th image.
- Let $\mathfrak{f}$ be the camera projection model.

Bundle Adjustment minimizes the *reprojection error* with respect to all the camera and 3D point positions:

$$\underset{\xi_i, \boldsymbol{X}_j}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j=1}^{m} \left\| \mathfrak{f}(\xi_i^{-1}(\boldsymbol{X}_j)) - \boldsymbol{p}_{ij} \right\|_2 \tag{5.20}$$

This optimization problem is solved using the Levenberg-Marquardt algorithm. The Jacobian which appears in the problem of bundle adjustment is sparse and has regular structure. Exploiting this fact, an efficient sparse bundle adjustment algorithm is proposed in Lourakis and Argyros (2004) as well as open-source implementation.

1.4.3 Outliers and Random Sample Consensus

Inliers and Outliers The term *outlier* often appears model fitting problems. Suppose that we have a set of data samples which follow a certain model but have some additive noise. For example, a set of detected feature points on two images with correspondences is generated by a certain geometric model (camera projection model, coordinate transformation, and so on) and a detection noise. We can reasonably assume that the detection noise is of a few pixels. More generally, we assume that the measurement noise follows a certain Gaussian distribution with known σ (which is, of course, an approximation) and also that there are no samples with noise stronger than a certain threshold, which depends on data dimension. For example, for a one-dimensional space traditionally this threshold is set to 3σ.

Let us suppose that we know the model parameters and find that some samples have fitting error much larger than 3σ, for example, in the case of visual features, an error of 20 pixels cannot be explained by the detection noise. A better explanation is that the matching process failed to find a proper correspondence. Let's define the notion of outliers formally.

- Let $\{x_i\}$ and $\{y_i\}$ be two observation datasets.
- Let λ be a set of numeric parameters.
- Let $y = f(x, \lambda)$ be the prediction model.
- Let $\mathcal{N}(0, \sigma)$ be a normal distribution with zero mean and σ standard deviation.
- Let $\tilde{y}_i = f(x_i, \lambda) + \mathcal{N}(0, \sigma)$

Inliers are data points for which the following is true:

$$y_i - \tilde{y}_i \sim \mathcal{N}(0, \sigma) \tag{5.21}$$

An *outlier* is a data sample which violates this distribution law. Usually, to state that a data sample is an outlier a threshold θ is chosen:

$$P(\|\mathcal{N}(0, \sigma)\| > \theta) < \epsilon \quad 0 < \epsilon \ll 1 \tag{5.22}$$

And if $\|y_i - \tilde{y}_i\| > \theta$, then an outlier is detected. $\epsilon = 0.05$ is the most common choice.

Outliers, if they are used as data samples in model fitting, may lead to significant parameter estimation errors. A standard pipeline for registration of a pair of datasets is as follows:

1. Detect features for both datasets and match them against each other. A certain number of mismatches will inevitably occur.
2. Discard outliers and get an initial approximation using robust transformation estimation.
3. Refine the transformation using local search techniques

Random Sample Consensus (RANSAC) An algorithm has been proposed by Fischler and Bolles (1981) to fit a model in the presence of outliers. The algorithm repetitively takes random set of s points, where s is the minimum number of data points necessary to fit the model. Then it fits the model and counts *inliers*. After several iterations it takes the largest set of inliers among all the tries and fits the model using all of them.

The needed number of iterations to fit the model with inliers only at least once with probability P is:

$$N \geq \frac{\log(1 - P)}{\log(1 - (1 - \epsilon)^s)} \tag{5.23}$$

Where ϵ is the fraction of outliers in the dataset. So we can see that a small s leads to a high algorithm efficiency. In the case of the fundamental matrix estimation, $s = 8$. The minimum number of keypoints needed to estimate the motion by means of BA is 5. Actually, the fact that the model is fitted with inliers only does not guarantee that all the other inliers will be detected. For example, if we use 5 points which are close on the image, just because of the noise, the inliers from the other parts of the image might produce significant reprojection errors.

If there is a prior estimation of the parameters, or any other additional information source, like current position prediction given by a filter or a wheel odometry measurement in the case of a localization problem, it is possible to use less points, and so perform less RANSAC iterations. The so-called 1-point RANSAC has been proposed by Civera *et al.* (2009).

2 Wheel and Visual Odometry Fusion

2.1 Multimodal Perception for Mobile Robotics

Let us recall what are the main features of proprioceptive and exteroceptive sensors. *Proprioceptive sensors* are locally consistent and generate a local prediction, which can be used later to facilitate the registration and matching process for other modalities. They require integration, hence the position estimation continuously diverges and another modality is required to correct the accumulated error.

Exteroceptive sensors allow the robot to construct a map and localize itself with respect to it later on. Such sensors provide the necessary connection between the robot and the environment and thus are absolutely necessary.

The concept of Multimodal Perception is to combine different modalities to attain a better performance. Human perception is based on local predictions and anticipation. We model the world around us as a continuous flow of states where the next state is determined by the previous one, and the multimodal perception allows us to predict the next state better. Intuitively, when we move, we know what kind of motion we perform, so we anticipate how the world perception image will change because of this motion.

In this context, two important terms must be defined. *Detection* is selection of information that belongs to an object pertinent to a given task, based on models and prior knowledge about the object class, the task and environment. An example of detection of keypoints is described above. Other examples are detecting straight lines, circles, pedestrians, cars in a street, buildings, parking lots and so on. It means select a set of pixels which presumably picture an object to be detected. The complexity of this task is rather obvious. Pedestrians is a vast class and its instances in images can be of different size, shape, and color.

Tracking is re-selection of information which belongs to an object previously detected in data of the same origin. This task is intrinsically easier than detection because here we have to find a concrete object with concrete properties, which narrows the search space. Moreover, due to the world continuity, we can reduce the search space even more by predicting the object's position and orientation based on previous observations.

We can go even further into abstractions and talk about *local* and *global* problems. In the case of a global problem, one has to scan the whole search space to find the exact solution. Local problems imply that the search space is significantly reduced and there are efficient techniques of local optimization which solve this problem. And our goal is to make as many problems as possible local.

Some of the most challenging problems which arise during a mapping process are *matching* and *registration*. By matching we mean finding one-to-one correspondence for two sets of features. Registration is a process of transforming data into the same coordinate system. And

generally these are global problems. Since each newly arrived dataset needs to be registered, detection and matching have to be done every time. There is an alternative: use another modality to get an approximate transformation and, as a result, reduce search complexity and avoid the detection part of the process.

To be more concrete, let's take vision as an exteroceptive modality, and wheel odometry as a proprioceptive one. Pure vision-based mapping systems exist (for example, Raúl *et al.* (2015); Engel *et al.* (2014)) and they solve the following problem: using only a video sequence filmed with a moving calibrated camera, reconstruct the camera's trajectory and build an environmental map. The problem becomes significantly simpler once we introduce another source of information, which directly gives us a motion estimation. Not only improves it precision by giving presumably decorrelated measurements but also it facilitates the registration process by providing an initial motion approximation.

Another issue addressed by a multimodal system is a non-static surrounding. The main assumption used in the design of robust estimators is that outliers don't follow any particular distribution and are decorrelated from one another. This assumption may be false since any moving object generates a particular set of outliers which gives an estimation of relative motion between the camera and the object, while we need an estimation of camera motion with respect to the ground. In some cases, such estimations can be rejected based on dynamic modeling and a hypothesis of limited acceleration. But it does not guarantee a correct estimation, if we consider the following example. The system observes a parked car that starting moving. There is no way for the system to know whether the robot itself is decelerating or the observed car is accelerating since both objects have the same acceleration limits. That is the system might consider that the accelerating car, not the rest of the objects around, represents the static environment.

On the other hand, proprioceptive sensors are much less influenced by the surrounding. In the case of wheel odometry, if the ground is not whet or dirty and its contact with the wheels closely follows the non-slipping and non-skidding model, the odometry gives very accurate local motion estimations. In this way we can choose among all the consistent sets of inliers the one which is in the best consensus with the odometry and make the system even more robust with respect to possible outliers.

2.2 Formal Definition

Let us define the visual odometry as an optimization problem with only 6 parameters: the unknown transformation. The feature point reconstruction is done implicitly, that is there are no parameters for 3D point reconstruction, as in typical bundle adjustment. For that, a special triangulation algorithm has been developed, as the classical midpoint triangulation has a singularity in whose neighborhood the most points actually lie.

Frame Configuration Let us first discuss the frame layout and transformation notation used throughout this section (see the illustration in Fig. 5.2). Frames are referred to by their origin, which is in fact just a 3D point. But the frame is also attributed a certain orientation. Frames are defined via transformations from SE(3). In the problem of visual odometry we consider only two frames at a time. Let us call them frames 1 and 2. To make the indexing general we just need to replace 1 by $i-1$ and 2 by i. The following frames are relevant for the localization purposes:

- O_0 — local map origin. The localization is done with respect to it.
- $O_{\mathrm{b},1}$ and $O_{\mathrm{b},2}$ — mobile base frames, or odometry frames defined via ξ_1 and ξ_2.
- O_{c} — camera frame, defined in O_{b} via ξ_{c}.

- ζ_b, or ζ — transformation between two consecutive poses of the mobile base frame. It is equivalent to ${}^1\xi_2$.
- ζ_o — odometry measurement between frames $O_{\mathrm{b},1}$ and $O_{\mathrm{b},2}$.
- ζ_v — camera motion.

ζ_v is not an independent variable, since the camera is rigidly linked to the base. It is defined as follows:

$$\zeta_\mathrm{v} = \xi_\mathrm{c}^{-1} \circ \zeta \circ \xi_\mathrm{c} \tag{5.24}$$

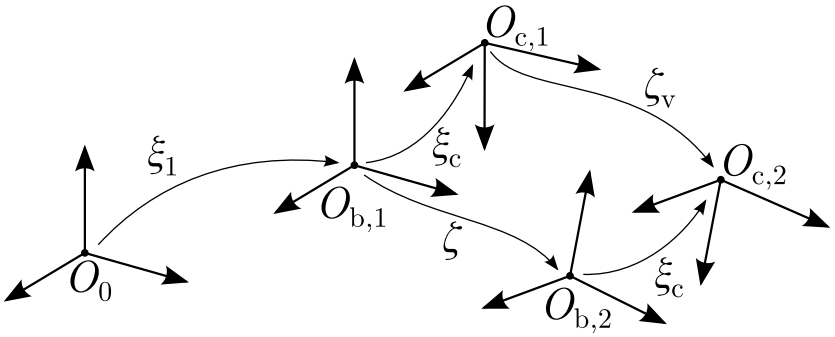


Figure 5.2: Frame configuration. O_0 is the local map origin; O_b the mobile base frames, or odometry frames; O_c is the camera frame; ξ define base frames in the map frame; ξ_c represents the camera's extrinsic parameters; ζ_b, ζ is the transformation between two consecutive poses of the base frame; ζ_v is the camera motion.

Optimization Problem First, let us introduce necessary notations:

- Let $\mathcal{P}_1 \subset \mathbb{R}^2$ be a set of detected features on the first image.
- Let $\mathcal{P}_2 \subset \mathbb{R}^2$ be a set of corresponding features on the second image. It is homologous to $\mathcal{P}_1$.
- Let $\mathfrak{h} : \mathbb{R}^2 \times \mathbb{R}^2 \times \mathrm{SE}(3) \to \mathbb{R}^2$ be a reprojection function.

$\mathfrak{h}$ takes both detected points and the transformation, triangulates and projects the triangulated point from the first camera onto the second image. The odometry problem is formulated as a non-linear optimization problem:

$$\zeta^* = \underset{\zeta}{\operatorname{argmin}} \sum_{\substack{\boldsymbol{p}\in\mathcal{P}_1 \\ \boldsymbol{q}\in\mathcal{P}_2}} \|\boldsymbol{q} - \mathfrak{h}(\boldsymbol{p}, \boldsymbol{q}, \zeta_\mathrm{v})\|_2 + (\zeta_\mathrm{o}^{-1} \circ \zeta)^T \frac{\mathcal{C}_\mathrm{o}^{-1}}{2} \zeta_\mathrm{o}^{-1} \circ \zeta \tag{5.25}$$

The first term contains visual data. Because of the feature detection noise and errors in ζ (remember that ζ_v is a function of ζ), the triangulation is not perfect and the reprojection error is not zero. Assuming that we use wheel odometry ζ_o to initialize ζ, the triangulation error should not be large and in this case $\mathfrak{h}(\boldsymbol{p}, \boldsymbol{q}, \zeta_\mathrm{o})$ should output a point relatively close to $\boldsymbol{q}$. We want $\mathfrak{h}$ to be *continuous* and *differentiable*, in order to apply a second-order non-linear optimization algorithm and minimize this reprojection error. In the following subsections we figure out what kind of triangulation can be used for this purpose.

The second term in (5.25) is a wheel odometry prior. $\mathcal{C}_\mathrm{o}$ is a covariance matrix, which represents the wheel odometry uncertainty. There are two advantages of such a formulation. Firstly, the wheel and visual odometries are tightly integrated; secondly, the dimension of the problem is 6, and it means that it is efficiently solvable using non-linear least squares, no

matter how many feature points are used. Also, it is easy to change the number of points in RANSAC and use, for example, one-point RANSAC algorithm Civera *et al.* (2009), or any other number (3 must be sufficient to fully constrain the 6 DoF transformation).

Odometry Covariance Matrix We assume the unicycle kinematic model and locally circular motion. It means that we assume non-slipping and non-skidding constraints. The uncertainty in odometry measurements is mainly due to:

- Violation of these constraints
- Errors in odometry calibration

For a different kind of kinematic scheme, strictly speaking, it may be necessary to do a similar analysis. But in our belief, it will not make much difference. Let ζ be a 3 DoF motion increment.

$$\zeta_i = \begin{pmatrix} x \\ y \\ \varphi \end{pmatrix} = \begin{pmatrix} l\cos\frac{\varphi}{2} \\ l\sin\frac{\varphi}{2} \\ \varphi \end{pmatrix} \tag{5.26}$$

where l is the elementary distance and φ is the elementary rotation (Fig. 5.3). Together they form a control vector $\boldsymbol{u} = (l, \varphi)^T$.

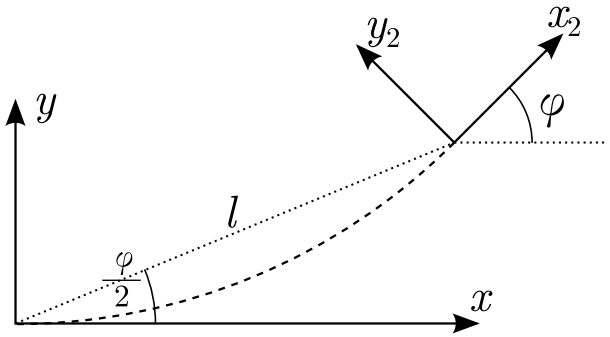


Figure 5.3: The circular motion model. φ is the turning angle; l is the distance. There are two frames: xy — the starting frame, x_2y_2 — the terminal local frame. The motion here is exaggerated to illustrate the geometry, but in reality the model is applied for small translations and rotations.

We are interested in $\mathcal{C}_\zeta$, the covariance which characterizes the error distribution of the odometry increment measurement in its local frame. It can be computed as follows:

$$\mathcal{C}_\zeta = \frac{\partial \zeta_{\text{local}}}{\partial \zeta}\frac{\partial \zeta}{\partial \boldsymbol{u}}\mathcal{C}_{\boldsymbol{u}}\left(\frac{\partial \zeta_{\text{local}}}{\partial \zeta}\frac{\partial \zeta}{\partial \boldsymbol{u}}\right)^T \tag{5.27}$$

Here ζ_{local} is the elementary motion expressed in the terminal frame. $\mathcal{C}_{\boldsymbol{u}}$ is the covariance of the control vector:

$$\mathcal{C}_{\boldsymbol{u}} = \begin{pmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_\varphi^2 \end{pmatrix} \tag{5.28}$$

We assume that the uncertainty, which is usually generated by small odometry intrinsic parameter errors, grows linearly with traveled distance. So, we propose to compute it as:

$$\begin{aligned} \sigma_l &= \lambda_l l \\ \sigma_\varphi &= \lambda_\varphi l \end{aligned} \tag{5.29}$$

The transformation between the root frame and the local frame is:

$$\frac{\partial \zeta_{\text{local}}}{\partial \zeta} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.30}$$

The Jacobian matrix of the motion:

$$\frac{\partial \zeta}{\partial \boldsymbol{u}} = \begin{pmatrix} \cos\frac{\varphi}{2} & -\frac{l}{2}\sin\frac{\varphi}{2} \\ \sin\frac{\varphi}{2} & \frac{l}{2}\cos\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix} \tag{5.31}$$

Multiplying the two matrices yields:

$$\frac{\partial \zeta_{\text{local}}}{\partial \boldsymbol{u}} = \begin{pmatrix} \cos\frac{\varphi}{2} & \frac{l}{2}\sin\frac{\varphi}{2} \\ -\sin\frac{\varphi}{2} & \frac{l}{2}\cos\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix} \tag{5.32}$$

And the final covariance matrix has the following form:

$$\mathcal{C}_\zeta = \frac{\partial \zeta_{\text{local}}}{\partial \boldsymbol{u}} \mathcal{C}_u \left(\frac{\partial \zeta_{\text{local}}}{\partial \boldsymbol{u}}\right)^T = \begin{pmatrix} \sigma_l^2 c_{\varphi/2}^2 + \sigma_\varphi^2 \frac{l^2}{4} s_{\varphi/2}^2 & \left(\frac{l^2\sigma_\varphi^2}{4} - \sigma_l^2\right) s_{\varphi/2} c_{\varphi/2} & \sigma_\varphi^2 \frac{l}{2} s_{\varphi/2} \\ \left(\frac{l^2\sigma_\varphi^2}{4} - \sigma_l^2\right) s_{\varphi/2} c_{\varphi/2} & \sigma_l^2 s_{\varphi/2}^2 + \sigma_\varphi^2 \frac{l^2}{4} c_{\varphi/2}^2 & \sigma_\varphi^2 \frac{l}{2} c_{\varphi/2} \\ \sigma_\varphi^2 \frac{l}{2} s_{\varphi/2} & \sigma_\varphi^2 \frac{l}{2} c_{\varphi/2} & \sigma_\varphi^2 \end{pmatrix} \tag{5.33}$$

Here we replaced $\sin\frac{\varphi}{2}$ and $\cos\frac{\varphi}{2}$ by $s_{\varphi/2}$ and $c_{\varphi/2}$ respectively. Now we need to bring it back to 6 DoF by replacing the untouched terms by zeros:

$$\mathcal{C}_\zeta^* = \begin{pmatrix} \mathcal{C}_\zeta(1,1) & \mathcal{C}_\zeta(1,2) & 0 & 0 & 0 & \mathcal{C}_\zeta(1,3) \\ \mathcal{C}_\zeta(2,1) & \mathcal{C}_\zeta(2,2) & 0 & 0 & 0 & \mathcal{C}_\zeta(2,3) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \mathcal{C}_\zeta(3,1) & \mathcal{C}_\zeta(3,2) & 0 & 0 & 0 & \mathcal{C}_\zeta(3,3) \end{pmatrix} \tag{5.34}$$

We also add a small constant diagonal covariance matrix $\tilde{\mathcal{C}}$ to the final result to introduce some uncertainty in the other 3 directions. If the mobile platform has a suspension, then rotations about x and y as well as translation along z may have greater uncertainties.

$$\mathcal{C}_\text{o} = \mathcal{C}_\zeta^* + \tilde{\mathcal{C}} \tag{5.35}$$

2.3 Middle Point Triangulation

Let $\boldsymbol{t}$ be the translation between the two camera positions; let $\boldsymbol{a}$ and $\boldsymbol{b}$ be the direction vectors from the first and the second viewpoints respectively to a 3D point, which has to be

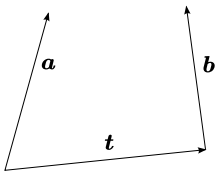


Figure 5.4: $\boldsymbol{a}$ and $\boldsymbol{b}$ are reconstructed directions towards a feature; $\boldsymbol{t}$ is the translation vector between the two positions.

triangulated (see Fig. 5.4). In the ideal case the two rays defined by the directions intersect and there exist λ_a and λ_b such that:

$$\lambda_a \boldsymbol{a} - \lambda_b \boldsymbol{b} - \boldsymbol{t} = 0 \tag{5.36}$$

But, due to errors in the feature detection and in the transformation estimation between the two camera poses, (5.36) is almost never true. To tackle this problem the least squares solution is usually applied:

$$\begin{cases} \boldsymbol{a} \cdot (\lambda_a \boldsymbol{a} - \lambda_b \boldsymbol{b} - \boldsymbol{t}) = 0 \\ \boldsymbol{b} \cdot (\lambda_a \boldsymbol{a} - \lambda_b \boldsymbol{b} - \boldsymbol{t}) = 0 \end{cases} \tag{5.37}$$

The meaning of (5.37) is that the residual $\boldsymbol{c}$ is perpendicular to both $\boldsymbol{a}$ and $\boldsymbol{b}$ (Fig. 5.5).

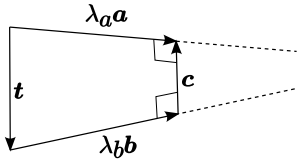


Figure 5.5: $\boldsymbol{c}$ is the residual after solving the least squares. It is perpendicular to both $\boldsymbol{a}$ and $\boldsymbol{b}$, so its length is minimized over λ_a and λ_b.

$$\begin{cases} \lambda_a \boldsymbol{a}^2 - \lambda_b \boldsymbol{a}\cdot\boldsymbol{b} = \boldsymbol{a}\cdot\boldsymbol{t} \\ \lambda_a \boldsymbol{a}\cdot\boldsymbol{b} - \lambda_b \boldsymbol{b}^2 = \boldsymbol{b}\cdot\boldsymbol{t} \end{cases} \tag{5.38}$$

Let us apply Cramer's rule to find the closed-form solution:

$$\begin{aligned} \Delta &= -\boldsymbol{a}^2\boldsymbol{b}^2 + (\boldsymbol{a}\cdot\boldsymbol{b})^2 \\ \Delta_a &= -\boldsymbol{a}\cdot\boldsymbol{t}\boldsymbol{b}^2 + \boldsymbol{b}\cdot\boldsymbol{t}\boldsymbol{a}\cdot\boldsymbol{b} \\ \Delta_b &= \boldsymbol{a}^2\boldsymbol{b}\cdot\boldsymbol{t} - \boldsymbol{a}\cdot\boldsymbol{b}\boldsymbol{a}\cdot\boldsymbol{t} \\ \lambda_a &= \frac{\Delta_a}{\Delta} \\ \lambda_b &= \frac{\Delta_b}{\Delta} \end{aligned} \tag{5.39}$$

Reconstruction Behind The Camera If $\boldsymbol{a}\cdot\boldsymbol{b} > 0$, which is true when the point is far away and the two vectors are close, the following is true:

$$\begin{cases} \boldsymbol{a}\cdot\boldsymbol{t} < 0 \\ \boldsymbol{b}\cdot\boldsymbol{t} > 0 \end{cases} \Rightarrow \begin{cases} \Delta_a > 0 \\ \Delta_b > 0 \end{cases} \Rightarrow \begin{cases} \lambda_a < 0 \\ \lambda_b < 0 \end{cases} \tag{5.40}$$

since $\Delta \leq 0$ for any $\boldsymbol{a}$, $\boldsymbol{b}$. The situation is illustrated on Fig. 5.6. So, just because of the detection noise we may have the point reconstructed behind the camera. What we would like to have, though, is a point far away in front.

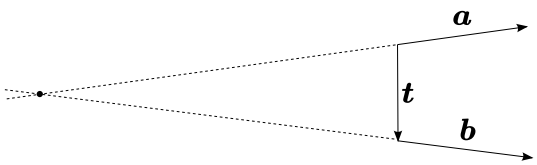


Figure 5.6: In case of diverging $\boldsymbol{a}$ and $\boldsymbol{b}$, the point is reconstructed behind the cameras.

Attempt to Regularize We can try a naive approach to prevent points from being reconstructed behind. The following approach is explained for $\boldsymbol{a}$:

1. Compute Δ and Δ_a
2. if $-\Delta > -\varepsilon\Delta_a, \quad \lambda_a = \dfrac{\Delta_a}{\Delta}$
3. else if $\Delta_a = 0, \quad \lambda_a = \dfrac{2}{\varepsilon}$
4. else $\lambda_a = \dfrac{2}{\varepsilon} - \dfrac{\Delta}{\varepsilon^2\Delta_a}$

$\forall \boldsymbol{a}, \boldsymbol{b}$ it gives a reconstruction in front of the camera.

Jacobian Singularity The regularization does not remove the Jacobian singularity in case of $\boldsymbol{a} = \boldsymbol{b}$. To verify this we can try to compute the following limit:

$$\lim_{\boldsymbol{a}\to\boldsymbol{b}} \frac{\Delta}{\Delta_a} \tag{5.41}$$

Let us say that $\boldsymbol{a}$ is approaching $\boldsymbol{b}$ and the angle between them is δ. For the sake of simplicity let us say that hereafter $||\boldsymbol{a}|| = ||\boldsymbol{b}|| = ||\boldsymbol{t}|| = 1$; it does not change the properties of the triangulation because $\boldsymbol{a}$ and $\boldsymbol{b}$ are just direction vectors and they can be renormalized according to $\boldsymbol{t}$. Then the following is true:

$$\begin{aligned} &\boldsymbol{a}\cdot\boldsymbol{b} = \cos\delta \\ &\boldsymbol{a}\cdot\boldsymbol{t} = \cos\varphi \\ &\boldsymbol{b}\cdot\boldsymbol{t} = \cos(\varphi + a\delta) \end{aligned} \tag{5.42}$$

where $a \in [-1, 1]$ is a function of δ, and it quickly converges to its limit value as $\delta \to 0$; it is -1 if δ and φ are antialigned; if they are aligned, $a = 1$; if they are orthogonal, $a = 0$. First, let's find Cramer's determinants:

$$\Delta = -\boldsymbol{a}^2\boldsymbol{b}^2 + (\boldsymbol{a}\cdot\boldsymbol{b})^2 = \cos^2\delta - 1 = -\sin^2\delta \tag{5.43}$$

$$\begin{aligned} &\Delta_a = -\boldsymbol{a}\cdot\boldsymbol{t}\boldsymbol{b}^2 + \boldsymbol{b}\cdot\boldsymbol{t}\boldsymbol{a}\cdot\boldsymbol{b} = \\ &-\cos\varphi + \cos(\varphi + a\delta)\cos\delta = \\ &-\cos\varphi + (\cos\varphi\cos a\delta - \sin\varphi\sin a\delta)\cos\delta = \\ &\cos\varphi(\cos a\delta\cos\delta - 1) - \sin\varphi\sin a\delta\cos\delta \end{aligned} \tag{5.44}$$

By plugging the values of Δ and Δ_a we can now compute the limit value:

$$\begin{aligned}
&\lim_{\delta\to 0}\frac{\Delta}{\Delta_a} = \lim_{\delta\to 0}\frac{-\sin^2\delta}{(\cos a\delta\cos\delta - 1)\cos\varphi - \sin a\delta\cos\delta\sin\varphi} = \\
&\lim_{\delta\to 0}\frac{-\delta^2}{(1-\frac{a^2+1}{2}\delta^2+\frac{a^2}{4}\delta^4-1)\cos\varphi - a\delta(1-\frac{\delta^2}{2})\sin\varphi} = \\
&\lim_{\delta\to 0}\frac{\delta}{a\sin\varphi}
\end{aligned} \tag{5.45}$$

We don't consider the case $\sin\varphi = 0$ which corresponds to the triangulation of the epipole, since it is impossible. So, depending on a, the value will approach its limit in different ways. It will approach 0 from the left, if $a < 0$, and from the right, if $a > 0$. Finally, if $a = 0$, the limit changes its value:

$$\lim_{\delta\to 0}\frac{\Delta}{\Delta_a} = \lim_{\delta\to 0}\frac{-\sin^2\delta}{(\cos\delta - 1)\cos\varphi} = \lim_{\delta\to 0}\frac{-\delta^2}{(1-\frac{1}{2}\delta^2-1)\cos\varphi} = \frac{2}{\cos\varphi} \tag{5.46}$$

This phenomenon has an intuitive explanation. $a = 0$ corresponds to rotation δ about an axis, perpendicular to $\boldsymbol{a}\times\boldsymbol{t}$. When $\delta = 0$, $\boldsymbol{a} = \boldsymbol{b}$ and the value is not defined (even though we assume that the point is at $+\infty$). But if we rotate $\boldsymbol{b}$ infinitesimally about $\boldsymbol{t}$, λ_a and λ_b immediately become defined and finite, because the common perpendicular to $\boldsymbol{a}$ and $\boldsymbol{b}$ passes near the camera origins (in fact, $\lambda_a = 0.5\cos\varphi$ and $\lambda_b = -0.5\cos\varphi$). All these facts mean that the singularity cannot be eliminated and the Jacobian matrix of the reconstructed point is not well defined in the neighborhood of the singularity.

2.4 Regularized Triangulation

We want the regularized triangulation to have the following properties:

- It is defined even for diverging direction vectors, so it does not prevent the optimization process from converging.
- Its Jacobian matrix is well-defined in the neighborhood of $\boldsymbol{a} = \boldsymbol{b}$.
- For close points and correct motion estimation it gives accurate point reconstruction.

We propose the following triangulation formulation:

$$\begin{cases} (\boldsymbol{a}+\boldsymbol{b})\cdot(\lambda_a\boldsymbol{a} - \lambda_b\boldsymbol{b} - \boldsymbol{t}) = 0 \\ \boldsymbol{t}\cdot(\lambda_a\boldsymbol{a} - \lambda_b\boldsymbol{b} - \boldsymbol{t}) = 0 \end{cases} \tag{5.47}$$

The residual is perpendicular to $\boldsymbol{t}$ and to $\boldsymbol{a}+\boldsymbol{b}$. Let us denote $\boldsymbol{a}+\boldsymbol{b}$ by $\boldsymbol{r}$. The solution given by the Cramer rule is:

$$\begin{aligned}
\Delta &= \boldsymbol{a}\cdot\boldsymbol{t}\boldsymbol{b}\cdot\boldsymbol{r} - \boldsymbol{b}\cdot\boldsymbol{t}\boldsymbol{a}\cdot\boldsymbol{r} \\
\Delta_a &= \boldsymbol{t}^2\boldsymbol{b}\cdot\boldsymbol{r} - \boldsymbol{r}\cdot\boldsymbol{t}\boldsymbol{b}\cdot\boldsymbol{t} \\
\Delta_b &= \boldsymbol{t}^2\boldsymbol{a}\cdot\boldsymbol{r} - \boldsymbol{r}\cdot\boldsymbol{t}\boldsymbol{a}\cdot\boldsymbol{t}
\end{aligned} \tag{5.48}$$

In this case:

$$\begin{aligned}
\Delta_a &\geq 0 \\
\Delta_b &\geq 0
\end{aligned} \tag{5.49}$$

With the equality in the case when $\boldsymbol{b} = \boldsymbol{t}$ for Δ_a and $\boldsymbol{a} = \boldsymbol{t}$ for Δ_b. On the other hand $\Delta = 0 \Rightarrow \boldsymbol{a} = \boldsymbol{b}$.

It means that $\lambda_a^{-1} = \frac{\Delta}{\Delta_a}$ is defined almost everywhere. So, the regularized reconstruction goes as follows (for λ_a):

1. Compute $\lambda_a^{-1} = \frac{\Delta}{\Delta_a}$
2. if $\lambda_a^{-1} > \varepsilon, \quad \lambda_a = \dfrac{1}{\lambda_a^{-1}}$
3. else $\lambda_a = \dfrac{2}{\varepsilon} - \dfrac{\lambda_a^{-1}}{\varepsilon^2}$

where ε is a small constant. Let us call this function ψ_ε:

$$\psi_\varepsilon(x, y) = \begin{cases} \dfrac{x}{y} & \text{if } \varepsilon x < y \\ \dfrac{2}{\varepsilon} - \dfrac{y}{\varepsilon^2 x} & \text{otherwise} \end{cases} \tag{5.50}$$

Its schematic plot is depicted on Fig. 5.7. It is differentiable:

$$\frac{\partial \psi_\varepsilon}{\partial x} = \begin{cases} \dfrac{1}{y} & \text{if } \varepsilon x < y \\ \dfrac{y}{\varepsilon^2 x^2} & \text{otherwise} \end{cases} \qquad \frac{\partial \psi_\varepsilon}{\partial y} = \begin{cases} -\dfrac{x}{y^2} & \text{if } \varepsilon x < y \\ -\dfrac{1}{\varepsilon^2 x} & \text{otherwise} \end{cases} \tag{5.51}$$

In the case when $\varepsilon x = y$ the derivatives from both sides are the same.

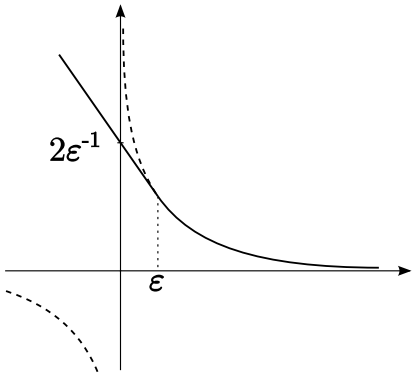


Figure 5.7: Solid curve is the regularized inverse $y = \psi_\varepsilon(1, x)$; dashed curve is just $y = x^{-1}$; ε is the stitching point. Instead of going to $+\infty$ and coming back from $-\infty$ when the argument crosses 0 from right to left, the function continues to get higher and higher.

Using ψ the triangulation is defined as follows:

$$\begin{aligned} \lambda_a &= \psi(\Delta_a, \Delta) \\ \lambda_b &= \psi(\Delta_b, \Delta) \end{aligned} \tag{5.52}$$

2.5 Reprojection Mapping and Jacobian Matrix

Now it is possible to define $\mathfrak{h}$ from (5.25):

$$\begin{aligned} \lambda_a &= \psi_\varepsilon(\Delta_a, \Delta) \\ \boldsymbol{X} &= \zeta_{\mathrm{v}}^{-1}(\lambda_a \boldsymbol{a}) \\ \hat{\boldsymbol{s}} &= \mathfrak{f}(\boldsymbol{X}) \end{aligned} \tag{5.53}$$

$\mathfrak{f}$ is the projection function of the second camera; $\hat{\boldsymbol{s}}$ is the projected point, the output of $\mathfrak{h}$. To avoid unnecessary superscripts, we omit the projection frame of $\boldsymbol{X}$ (which is actually ${}^{c,2}\boldsymbol{X}$), as it is the same across the section. The Jacobian matrix of $\mathfrak{h}$ can be expressed using the chain rule:

$$\frac{\partial \hat{\boldsymbol{s}}}{\partial \zeta} = \frac{\partial \hat{\boldsymbol{s}}}{\partial \boldsymbol{X}} \left(\frac{\partial \boldsymbol{X}}{\partial \zeta} + \frac{\partial \boldsymbol{X}}{\partial \lambda_a} \frac{\partial \lambda_a}{\partial \zeta} \right) \tag{5.54}$$

Let's break it down:

1. $\dfrac{\partial \hat{\boldsymbol{s}}}{\partial \boldsymbol{X}}$ is the projection Jacobian matrix.
2. $\dfrac{\partial \boldsymbol{X}}{\partial \zeta}$ is the part responsible for the 3D point motion directly related to the transformation change.
3. $\dfrac{\partial \boldsymbol{X}}{\partial \lambda_a}$ is the point motion caused by change in triangulation.
4. $\dfrac{\partial \lambda_a}{\partial \zeta}$ is the triangulation dependence on the transformation

While 1 is given by the camera model and 3 is trivial, 2 and 4 are more involved. We show below how to express them.

3D Point Jacobian Matrix The Jacobian matrix we are looking for is equivalent to the kinematic Jacobian matrix. We remind that the we use 1 and 2 instead of $i-1$ and i for frame indices, since we have only two of them. To compute it, let us assume that $O_{\mathrm{b},2}$ is moving with respect to $O_{\mathrm{b},1}$ because $\dot{\zeta} \neq 0$. Then, the following is true:

$$\dot{\boldsymbol{X}} = \frac{\partial \boldsymbol{X}}{\partial \zeta} \dot{\zeta} \tag{5.55}$$

We remind that $\boldsymbol{X}$ is ${}^{c,2}\boldsymbol{X}$. To figure out what are the expressions for $\dfrac{\partial \boldsymbol{X}}{\partial \zeta}$, we need to express $\dot{\boldsymbol{X}}$ through $\dot{\zeta}$. In this case the 3D point is static and the camera is moving. Hence, the relative motion of the point with respect to the camera can be written as follows:

$$\dot{\boldsymbol{X}} = \begin{pmatrix} -I & [\boldsymbol{X}]_\times \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_\mathrm{c} \\ \boldsymbol{\omega}_\mathrm{c} \end{pmatrix} \tag{5.56}$$

Here $(\boldsymbol{v}_\mathrm{c}, \boldsymbol{\omega}_\mathrm{c})$ define the camera's motion in $O_{\mathrm{c},2}$ robot's position, and it is related to the motion of the base frame:

$$\begin{pmatrix} \boldsymbol{v}_\mathrm{c} \\ \boldsymbol{\omega}_\mathrm{c} \end{pmatrix} = \begin{pmatrix} {}^\mathrm{c}R_\mathrm{b} & -{}^\mathrm{c}R_\mathrm{b}[\boldsymbol{t}_\mathrm{c}]_\times \\ 0 & {}^\mathrm{c}R_\mathrm{b} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_\mathrm{b} \\ \boldsymbol{\omega}_\mathrm{b} \end{pmatrix} \tag{5.57}$$

${}^\mathrm{c}R_\mathrm{b}$ is the rotation matrix from ξ_c^{-1}; $\boldsymbol{t}_\mathrm{c}$ is its translational part. $(\boldsymbol{v}_\mathrm{b}, \boldsymbol{\omega}_\mathrm{b})$ is the kinematic screw of $O_{\mathrm{b},2}$. Let us first compute it in the root frame of ζ, that is, in $O_{\mathrm{b},1}$:

$$\begin{pmatrix} {}^1\boldsymbol{v}_\mathrm{b} \\ {}^1\boldsymbol{\omega}_\mathrm{b} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & M_{\boldsymbol{r}} \end{pmatrix} \dot{\zeta} \tag{5.58}$$

Here $M_{\boldsymbol{r}}$ is the interaction matrix which converts $\dot{\boldsymbol{r}}$ into $\boldsymbol{\omega}$. For $\boldsymbol{r} = \boldsymbol{u}\vartheta$ it has the following form:

$$M_{\boldsymbol{r}} = I + \frac{\vartheta}{2} \mathrm{sinc}^2 \left(\frac{\vartheta}{2} \right) [\boldsymbol{u}]_\times + (1 - \mathrm{sinc}\,\vartheta)[\boldsymbol{u}]_\times^2 \tag{5.59}$$

Projecting the vectors into $O_{\mathrm{b},2}$ gives:

$$\begin{pmatrix} \boldsymbol{\nu}_{\mathrm{b}} \\ \boldsymbol{\omega}_{\mathrm{b}} \end{pmatrix} = \begin{pmatrix} {}^{2}R_{1} & 0 \\ 0 & {}^{2}R_{1}M_{\boldsymbol{r}} \end{pmatrix} \dot{\zeta} \tag{5.60}$$

${}^{2}R_{1}$ is the rotation matrix associated with ζ^{-1}. Putting (5.56), (5.57), and (5.60) together yields:

$$\frac{\partial \boldsymbol{X}}{\partial \zeta} = \begin{pmatrix} -I & [\boldsymbol{X}]_{\times} \end{pmatrix} \begin{pmatrix} {}^{\mathrm{c}}R_{\mathrm{b}}\,{}^{2}R_{1} & -{}^{\mathrm{c}}R_{\mathrm{b}}[\boldsymbol{t}_{\mathrm{c}}]_{\times}\,{}^{2}R_{1}M_{\boldsymbol{r}} \\ 0 & {}^{\mathrm{c}}R_{\mathrm{b}}\,{}^{2}R_{1}M_{\boldsymbol{r}} \end{pmatrix} \tag{5.61}$$

The advantage of this form of the Jacobian matrix is that the second matrix does not depend on $\boldsymbol{X}$, so it can be computed only once for each ζ and reused for multiple points.

Triangulation Jacobian Matrix The definition of λ_a is given in (5.53), while Δ_a and Δ are defined in (5.48). The chain rule gives the following expression:

$$\frac{\partial \lambda_a}{\partial \zeta} = \frac{\partial \lambda_a}{\partial \Delta}\frac{\partial \Delta}{\partial \zeta} + \frac{\partial \lambda_a}{\partial \Delta_a}\frac{\partial \Delta_a}{\partial \zeta} \tag{5.62}$$

To find $\frac{\partial \Delta}{\partial \zeta}$ and $\frac{\partial \Delta_a}{\partial \zeta}$, let us use the same method and assume that $O_{\mathrm{c},2}$ moves with respect to $O_{\mathrm{c},1}$ with $\boldsymbol{\nu}$ and $\boldsymbol{\omega}$ due to nonzero $\dot{\zeta}$ (in this paragraph, if a vector's projection frame is not specified, it is $O_{\mathrm{c},1}$); $\boldsymbol{t}$ is the translation part of ζ_{v} and its time derivative $\dot{\boldsymbol{t}} = \boldsymbol{\nu}$; $\boldsymbol{b}$ is bound to the second camera and it is changing due to the rotation:

$$\dot{\boldsymbol{b}} = \boldsymbol{\omega} \times \boldsymbol{b} = -[\boldsymbol{b}]_{\times}\boldsymbol{\omega} \tag{5.63}$$

Keeping in mind that $\boldsymbol{r} = \boldsymbol{a} + \boldsymbol{b}$ and $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{b}^2 = 0$:

$$\dot{\Delta} = (\boldsymbol{b}\cdot\boldsymbol{r}\boldsymbol{a} - \boldsymbol{a}\cdot\boldsymbol{r}\boldsymbol{b}) \cdot \dot{\boldsymbol{t}} + (\boldsymbol{a}\cdot\boldsymbol{t}\boldsymbol{a} - \boldsymbol{b}\cdot\boldsymbol{t}\boldsymbol{a} - \boldsymbol{a}\cdot\boldsymbol{r}\boldsymbol{t}) \cdot \dot{\boldsymbol{b}} \tag{5.64}$$

Hence:

$$\frac{\partial \Delta}{\partial \zeta} = \Big((\boldsymbol{b}\cdot\boldsymbol{r}\boldsymbol{a} - \boldsymbol{a}\cdot\boldsymbol{r}\boldsymbol{b})^{T} \quad -((\boldsymbol{a}\cdot\boldsymbol{t} - \boldsymbol{b}\cdot\boldsymbol{t})\boldsymbol{a} - \boldsymbol{a}\cdot\boldsymbol{r}\boldsymbol{t})^{T}\,[\boldsymbol{b}]_{\times} \Big)\, L \tag{5.65}$$

Here L is the interaction matrix. It is defined via the following general relation between the transformation time derivative and the kinematic screw:

$$\begin{pmatrix} {}^{1}\boldsymbol{\nu}_{2} \\ {}^{1}\boldsymbol{\omega}_{2} \end{pmatrix} = L\,{}^{1}\dot{\xi}_{2} \tag{5.66}$$

L should be expressed in O_1, the base frame of ${}^{1}\xi_{2}$, and the kinematic screw $\left({}^{1}\boldsymbol{\nu}_{2} \atop {}^{1}\boldsymbol{\omega}_{2}\right)$ corresponds to the motion of O_2 with respect to O_1 projected into O_1.

Similarly, we can compute $\frac{\partial \Delta_a}{\partial \zeta}$:

$$\dot{\Delta}_a = (2\boldsymbol{b}\cdot\boldsymbol{r}\boldsymbol{t} - \boldsymbol{b}\cdot\boldsymbol{t}\boldsymbol{r} - \boldsymbol{r}\cdot\boldsymbol{t}\boldsymbol{b}) \cdot \dot{\boldsymbol{t}} + (\boldsymbol{t}^2\boldsymbol{a} - \boldsymbol{b}\cdot\boldsymbol{t}\boldsymbol{t} - \boldsymbol{r}\cdot\boldsymbol{t}\boldsymbol{t}) \cdot \dot{\boldsymbol{b}} \tag{5.67}$$

$$\frac{\partial \Delta_a}{\partial \zeta} = \Big((2\boldsymbol{b}\cdot\boldsymbol{r}\boldsymbol{t} - \boldsymbol{b}\cdot\boldsymbol{t}\boldsymbol{r} - \boldsymbol{r}\cdot\boldsymbol{t}\boldsymbol{b})^{T} \quad -\big(\boldsymbol{t}^2\boldsymbol{a} - (\boldsymbol{b}\cdot\boldsymbol{t} + \boldsymbol{r}\cdot\boldsymbol{t})\boldsymbol{t}\big)^{T}\,[\boldsymbol{b}]_{\times} \Big)\, L \tag{5.68}$$

If we recall that $\boldsymbol{\nu}$ is ${}^{\mathrm{c},1}\boldsymbol{\nu}_{\mathrm{c},2}$ and $\boldsymbol{\omega}$ is ${}^{\mathrm{c},1}\boldsymbol{\omega}_{\mathrm{c},2}$, then L is defined similarly to what we have seen in the previous paragraph. Reusing (5.57) and (5.58) we obtain:

$$\begin{pmatrix} {}^{\mathrm{c},2}\boldsymbol{\nu}_{\mathrm{c},2} \\ {}^{\mathrm{c},2}\boldsymbol{\omega}_{\mathrm{c},2} \end{pmatrix} = \begin{pmatrix} {}^{\mathrm{c}}R_{\mathrm{b}}\,{}^{2}R_{1} & -{}^{\mathrm{c}}R_{\mathrm{b}}[\boldsymbol{t}_{\mathrm{c}}]_{\times}\,{}^{2}R_{1}M_{\boldsymbol{r}} \\ 0 & {}^{\mathrm{c}}R_{\mathrm{b}}\,{}^{2}R_{1}M_{\boldsymbol{r}} \end{pmatrix} \dot{\zeta} \tag{5.69}$$

By simply projecting the vectors into $O_{c,i}$ we get:

$$L = \begin{pmatrix} {}^{c,1}R_{c,2} & 0 \\ 0 & {}^{c,1}R_{c,2} \end{pmatrix} \begin{pmatrix} {}^{c}R_{b}\,{}^{2}R_{1} & -{}^{c}R_{b}[\boldsymbol{t}_{c}]_{\times}\,{}^{2}R_{1}M_{\boldsymbol{r}} \\ 0 & {}^{c}R_{b}\,{}^{2}R_{1}M_{\boldsymbol{r}} \end{pmatrix} = \begin{pmatrix} {}^{c}R_{b} & -{}^{c,1}R_{b,2}[\boldsymbol{t}_{c}]_{\times}\,{}^{2}R_{1}M_{\boldsymbol{r}} \\ 0 & {}^{c}R_{b}M_{\boldsymbol{r}} \end{pmatrix} \quad (5.70)$$

It can be simplified even further:

$$[\boldsymbol{t}_{c}]_{\times}\,{}^{2}R_{1} = {}^{2}R_{1}[{}^{1}R_{2}\boldsymbol{t}_{c}]_{\times}\,{}^{1}R_{2}\,{}^{2}R_{1} = {}^{2}R_{1}[{}^{1}R_{2}\boldsymbol{t}_{c}]_{\times} \quad (5.71)$$

Which produces:

$$L = \begin{pmatrix} {}^{c}R_{b} & -{}^{c}R_{b}[{}^{1}R_{2}\boldsymbol{t}_{c}]_{\times}M_{\boldsymbol{r}} \\ 0 & {}^{c}R_{b}M_{\boldsymbol{r}} \end{pmatrix} \quad (5.72)$$

2.6 Results

The overall architecture of the tested visual odometry is depicted in Fig. C.12. To test this odometry, a synthetic sequence has been used [1]. The trajectories have been generated as follows:

1. A smooth trajectory is defined via a sequence of straight lines and arcs.
2. The wheel odometry bias is simulated by adding a small rotation to every trajectory increments.
3. The Ground Truth, which is used as the camera position for image rendering, is obtained by adding some noise to the smooth trajectory. It is done to model the suspension effect.

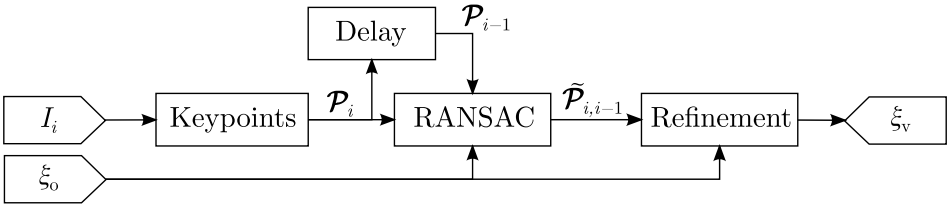


Figure 5.8: Architecture of the visual odometry. I_i and $\xi_{o,i}$ are the inputs: current image and odometry measurements; $\mathcal{P}_i$ is the detected feature set; $\tilde{\mathcal{P}}_{i,i-1}$ is the inlier set, computed by the RANSAC.

Two-point RANSAC with the wheel odometry prior is used. It demonstrates high efficiency. The resulting trajectory is presented in Fig. C.14. The wheel odometry diverges quickly because of a strong rotation bias, while the visual odometry follows the real trajectory much closer.

Possible Improvements

- Use epipolar constraint with the transformation provided by the wheel odometry to improve the matching accuracy.
- Keep the depth estimation for the next step to make the matching even more robust. It would even be possible to track the features.
- In the context of mobile platforms with suspension, like cars, an IMU installed onto the same rigid body as the camera can improve the motion estimation precision and make the system even more resistant to outliers, since IMUs give an accurate rotation estimation, which not only complements the z rotation given by the wheel odometry, but also provides the information about the suspension state.

[1] Technical details of image rendering are given in Appendix B

Figure 5.9: An image from the synthetic sequence.

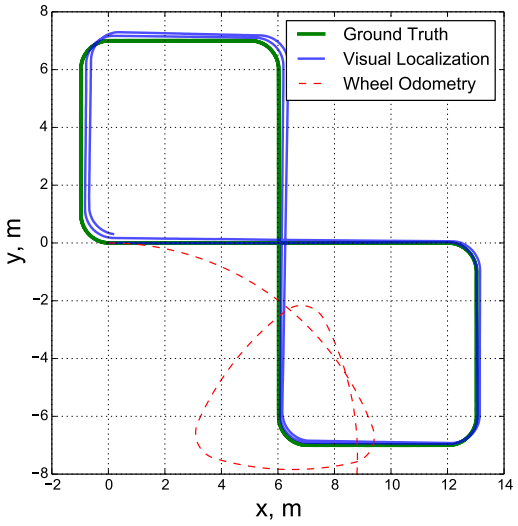


Figure 5.10: Visual odometry test with simulated data. Wheel odometry has a strong orientation bias, nevertheless, the divergence of visual odometry trajectory is less than a half a meter for two laps with total length of 114 m.

3 Direct Visual Odometry

We can try to imagine a system where the step of feature detection is completely omitted. The advantage of the feature-based methods is that the problem becomes purely geometric, once the features are detected. On the other hand, if the environment does not contain enough features of the needed type, then the system gets lost, even though images may contain a lot of different textures. A simple example to show the advantage of direct methods: two black circles on white background have zero point-like features, and yet it is sufficient to localize the camera.

A possible way of egomotion estimation is via dense optical flow, as it has been done for fisheye cameras in Radgui *et al.* (2011). Another approach consists in applying virtual visual servoing for image registration. Photometric visual servoing has been applied to vision-based robot navigation (For example Caron *et al.* (2013)). For virtual visual servoing, a 3D model of the environment is required. At first, the direct methods have been applied together with RGB-D sensors (for example Izadi *et al.* (2011); Kerl *et al.* (2013)). In the case of RGB-D, 3D reconstruction is directly available, which makes the overall problem easier. In Meilland *et al.* (2015) stereo vision is used to perceive the depth independently of the motion estimation. But in the case of a single camera, depth and motion cannot be decoupled and have to be computed simultaneously. Such system has been proposed first as odometry Engel *et al.* (2013), and then as a SLAM system (Engel *et al.* (2014)).

In this section we describe a possible way of applying the Enhanced Unified Model and the direct stereo correspondence algorithm to the direct visual localization. Also we try Mutual Information as a similarity measure between two images instead of photometric error.

3.1 Photometric Registration

In this context registration means, given two images of the same scene taken with a small time interval and a direct 3D reconstruction for one of them, to find the transformation between

the two viewpoints which minimizes the photometric reprojection error. By photometric error we mean the difference in pixel brightnesses.

Brightness Constancy Hereafter we imply the following hypothesis: *an observed surface point produces the same pixel brightness, whatever the distance, the observation angle and the time moment.*

Obviously, this statement is not true for the following reasons:

- Each pixel does not observe a single point, but rather a small patch of the surface and its resulting intensity is the average intensity of that patch. Brightness of pixels which represent a border of an object is an even more complicated matter.
- This assumption also implies so-called Lambertian reflectance, which is a model of a perfect matte surface. It means that the light is dissipated by it uniformly in all directions. This is an idealized model because generally the brightness depends on the observation angle. Moreover, if a surface is glossy then its brightness may vary a lot: from black, when the reflections are not observed, to white when the surface is reflecting a light source.
- The lighting changes over time. Outdoors lighting can change rapidly due to irregular clouds, and it changes slowly and steadily during the day because of the earth rotation. Indoor lighting is more stable, but it can also change when additional light sources are turned on or off. Fluorescent light produces flickering at 50 or 60 Hz depending on the network frequency. Incandescent light reacts to the network tension. Even opening and closing doors and just passing by affects the lighting (a red object, for example, brought closely to a white surface, will make it reddish).

But taking all these effects into account is, so far, impossible. Rapid-dynamics effects must be handled by robust properties of the localization, while slow-dynamics effects (such as day-night changes) don't affect the system as it is supposed to work at much higher frequencies. For long-term localization, other techniques, not sensitive to lighting changes, can be applied. For example, mutual-information-based localization, which we describe later on.

Mathematical Formulation The frame and transformation notations are the same as in the previous section (see Fig. 5.2). Let us define I_1 and I_2 as the images taken from $O_{c,1}$ and $O_{c,2}$ respectively. The error function is defined as follows:

$$E(\zeta) = \frac{1}{2} \sum_{\boldsymbol{p} \in \Omega} \left[I_2 \left(w(\boldsymbol{p}, \zeta_v) \right) - I_1(\boldsymbol{p}) \right]^2 \tag{5.73}$$

Here, $\zeta_v = \xi_c^{-1} \circ \zeta \circ \xi_c$, as in the previous section; $w : \Omega \times \mathrm{SE}(3) \to \mathbb{R}^2$ is a wrap function, which depends on camera's intrinsic parameters and on depth mapping:

$$\mathcal{D} : \Omega \subset \mathbb{R}^2 \to \mathbb{R} \tag{5.74}$$

This mapping is usually represented by a depth map, that is an array of values which represent the depth of pixels in the original image. Generally this map is obtained using a stereo reconstruction algorithm. For now let us just assume that we are given the depth map.

The wrap function is computed as follows:

1. Reconstruct $\boldsymbol{p}$ into a 3D direction $\boldsymbol{e} = \mathfrak{f}^{-1}(\boldsymbol{p})$ using (C.9).
2. Using $\mathcal{D}$, reconstruct 3D point ${}^1\boldsymbol{X} = \mathcal{D}(\boldsymbol{p})\boldsymbol{e}$.

3. Compute ${}^2\boldsymbol{X} = \zeta_{\mathrm{v}}^{-1}({}^1\boldsymbol{X})$
4. Project the point : $w(\boldsymbol{p}, \xi) = \mathfrak{f}({}^2\boldsymbol{X})$

Also, while $\boldsymbol{p}$ has integer coordinates because it is related to a particular pixel, $w(\boldsymbol{p}, \xi)$ is not generally speaking integer, hence $I_2\left(w(\boldsymbol{p}, \xi)\right)$ does not have exactly the same meaning as $I_1(\boldsymbol{p})$. While the latter represents just the value stored in the image matrix, the former requires a certain interpolation technique. In this work, the bicubic interpolation is used. The localization problem is defined as follows:

$$\zeta^* = \underset{\zeta}{\operatorname{argmin}} E(\zeta) \tag{5.75}$$

Taking into account extrinsic transformation ξ_{c}, we can optimize the wrapping process by precomputing the point cloud in $O_{\mathrm{b},1}$ for all $\boldsymbol{p}_i \in \Omega$:

1. $\boldsymbol{e}_i \leftarrow \mathfrak{f}^{-1}(\boldsymbol{p}_i)$ — reconstruct the direction
2. ${}^{\mathrm{c},1}\boldsymbol{X}_i \leftarrow \mathcal{D}(\boldsymbol{p}_i)\boldsymbol{e}_i$ — compute the spatial point in the camera frame
3. ${}^{\mathrm{b},1}\boldsymbol{X}_i \leftarrow \xi_{\mathrm{c}}^{-1}({}^{\mathrm{c},1}\boldsymbol{X}_i)$ — transform it into the base frame

This point cloud remains the same during the whole optimization process. Let us agree that if there is no projection frame specified for spatial points $\boldsymbol{X}$, they are projected into $O_{\mathrm{c},2}$. So the cost function becomes:

$$E(\zeta) = \frac{1}{2}\sum_i \left[I_2\left(\mathfrak{f}\left(\boldsymbol{X}_i\right)\right) - I_1(\boldsymbol{p}_i)\right]^2 \tag{5.76}$$

Where $\boldsymbol{X}_i = (\zeta \circ \xi_{\mathrm{c}})^{-1}({}^{\mathrm{b},1}\boldsymbol{X}_i)$. Since monocular odometry deals with only one camera, the projection mapping $\mathfrak{f}$ is the same for both camera positions.

Solution To solve this problem, we can use non-linear least squares approach, more precisely, Levenberg-Marquardt optimization technique (Lourakis and Argyros (2004)). This method becomes efficient if we can compute the analytic Jacobian matrix expressions, which can be computed using the chain rule. Let $g : \mathrm{SE}(3) \to \mathbb{R}^N$ be the model prediction function. In this case it is defined as:

$$g_i = I_2 \circ \mathfrak{f}\left(\boldsymbol{X}_i\right) \tag{5.77}$$

The objective is to minimize the norm of the error vector:

$$\zeta^* = \underset{\zeta}{\operatorname{argmin}} \|g_i - I_1(\boldsymbol{p}_i)\| \tag{5.78}$$

The Jacobian matrix we are looking for is the following one:

$$\frac{\partial g_i}{\partial \zeta} = \frac{\partial I_2}{\partial \mathfrak{f}}\frac{\partial \mathfrak{f}}{\partial \boldsymbol{X}_i}\frac{\partial \boldsymbol{X}_i}{\partial \zeta} \tag{5.79}$$

Let us brake it down:

1. $\dfrac{\partial I_2}{\partial \mathfrak{f}}$ is the image gradient.
2. $\dfrac{\partial \mathfrak{f}}{\partial \boldsymbol{X}_i}$ is the projection Jacobian matrix, known for the camera model.
3. $\dfrac{\partial \boldsymbol{X}_i}{\partial \zeta}$ defines how the point moves with respect to the second camera when the transformation changes. It has been derived in the previous section, its expression is given in (5.61)

3.2 Mutual-Information-Based Registration

To make the system more robust with respect to changes in object position and light conditions, we suggest to use the mutual-information-based image registration, rather than photometric error minimization. To the best of our knowledge, it has been applied to registration of images of planar objects in Dame and Marchand (2012) and to image-based mobile robot navigation in Raj Bista *et al.* (2016b).

Here we propose to combine it with dense vision-based reconstruction for SLAM purposes. Image registration is done via MI maximization:

$$\xi^* = \underset{\xi}{\operatorname{argmin}} -\mathrm{MI}(I_1, I_2, \mathcal{D}, \xi) \tag{5.80}$$

MI-based image registration has the following advantages:

- MI is almost insensitive to outliers
- MI is quite robust with respect to lighting changes

By definition, MI between two random variables A and B is computed as follows:

$$\mathrm{MI}(A, B) = \sum_{a \in A} \sum_{b \in B} P(a, b) \log \left(\frac{P(a, b)}{P(a)P(b)} \right) \tag{5.81}$$

That is, the sum is done over all possible values of A and B; $P(a)$ is the probability that the random variable A takes the value a; $P(a, b)$ is the joint probability. MI measures how much information one variable conveys about the other. In our case, random variables are pixel brightness values. And we want the two pixel sets, the target image and the wrapped reference image, to be consistent. That is, if two pixels of the reference image have close brightness values, then it is likely that the corresponding pixels on the target image also have close brightness values. It does not imply that the brightness does not change from one image to another, but rather that it changes consistently.

The main challenge is to be able to compute MI itself and the gradient with respect to ζ. Generally, MI for images is computed via histograms

3.2.1 Histogram and MI Computation

In our case, A is the brightness of the target image, while B is the brightness of reprojected pixels from the base image. The first step is to compute the values of the pixels under consideration. It is done in the same manner as in the photometric approach: wrap function w is computed and the brightness values for the reprojected points are computed via the bicubic interpolation.

Now suppose that we have two sets $A = \{a_j\}$ and $B = \{b_j\}$ of brightness samples from the first and the second image respectively; $j \in 1..M$. In order to compute MI we have to construct a 2D histogram.

Let us first consider a 1D histogram and some of its aspects. Generally, a histogram H has N bins delimited by a sequence of thresholds $\{t_i\} \quad i \in 0..N$ and for a given sample set, it counts how many of them lie inside each bin. The number of samples lying between t_{i-1} and t_i is denoted by h_i.

There is an important constraint on the histogram computation, imposed by this application: MI must be a continuous and differentiable function of ζ, hence the histogram values must have the same properties. The brightness values of pixels from the first image are constant. Only the brightness values sampled from the second image are changing. And they are continuous functions of transformation ζ since they are interpolated.

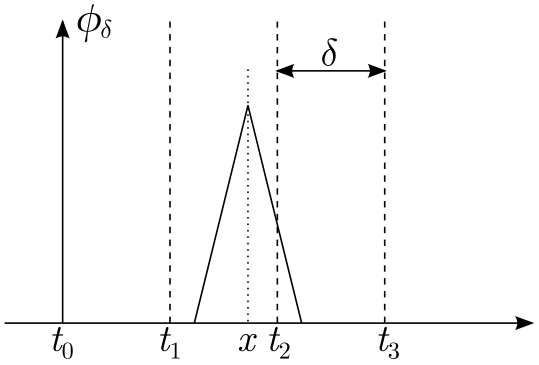


Figure 5.11: Histogram computation via sample response function ϕ. For each sample, a histogram bin increment is calculated as an integral of ϕ between the corresponding bin's limits. ϕ_δ is the function defined in (5.86).

To make the histogram a continuous function, the following simple technique, illustrated in Fig. 5.11, can be applied. Let $\phi : \mathbb{R} \to \mathbb{R}$ be a *sample response function* if it has following properties:

1. $\phi(x) \geq 0 \quad \forall x$
2. $\int_{-\infty}^{\infty} \phi(x)\mathrm{d}x = 1$

For example it can be defined as follows for a certain parameter δ:

$$\phi_\delta(x) = \begin{cases} \frac{1}{\delta} & \text{if} \quad -\frac{\delta}{2} \leq x \leq \frac{\delta}{2} \\ 0 & \text{otherwise} \end{cases} \tag{5.82}$$

Then, the histogram is computed as follows:

$$h_i = \sum_{a \in A} \int_{t_{i-1}}^{t_i} \phi(x - a)\mathrm{d}x \tag{5.83}$$

To keep the total count constant, we must compute the edge bins in a slightly different manner:

$$h_1 = \sum_{a \in A} \int_{-\infty}^{t_1} \phi(x - a)\mathrm{d}x \qquad h_N = \sum_{a \in A} \int_{t_{N-1}}^{\infty} \phi(x - a)\mathrm{d}x \tag{5.84}$$

This way of computing the histogram makes it a continuous and differentiable function of A. We do the differentiation later on in this section. Also, for some applications it may be interesting to compute a non-uniform histogram. In our case, a uniform histogram is used:

$$t_i = i\delta \tag{5.85}$$

Where δ is the bin size. The sample response is defined as follows (Fig. 5.11):

$$\phi_\delta = \begin{cases} \frac{2 + 4x}{\delta} & \text{if} \quad -\frac{\delta}{2} \leq x < 0 \\ \frac{2 - 4x}{\delta} & \text{if} \quad 0 \leq x < \frac{\delta}{2} \\ 0 & \text{otherwise} \end{cases} \tag{5.86}$$

This form has the advantage that its derivative exists, even though it is not continuous. Existence of its derivative makes the second order optimization of MI more consistent. We tried functions of a higher order differentiability for this purpose, but it did not provide any significant improvement in either the convergence properties or the localization precision.

Fig. 5.11 illustrates the histogram computation process. The support of ϕ has length δ, so in the case of 1D histogram generally every sample contributes to two histogram bins, unless it is perfectly aligned with one of their centers.

2D histogram computation is done as follows:

$$h_{ij} = \sum_{k=1}^{M} \int_{t_{i-1}}^{t_i} \phi(x - a_k)\mathrm{d}x \int_{t_{j-1}}^{t_j} \phi(y - b_k)\mathrm{d}y \tag{5.87}$$

The edges must be handled in a similar manner as (5.84).

To compute MI, we need three histograms: H_A, H_B, and H_{AB}. Since we have three histograms here, let us denote by h_i^A and h_j^B the elements of the 1D histograms; elements h_{ij} of H_{AB} can be distinguished thanks to their double subscript indices. H_A is constant; H_{AB} must be computed at every step. H_B is obtained by marginalizing H_{AB}:

$$h_j^B = \sum_{i=1}^{N} h_{ij} \tag{5.88}$$

Once normalized, the histograms provide the necessary probabilities:

$$\begin{aligned} \mathcal{P}\big(a \in [t_{i-1}, t_i[\big) &= \frac{h_i^A}{M} \\ \mathcal{P}\big(b \in [t_{j-1}, t_j[\big) &= \frac{h_j^B}{M} \\ \mathcal{P}\big(a \in [t_{i-1}, t_i[, b \in [t_{j-1}, t_j[\big) &= \frac{h_{ij}}{M} \end{aligned} \tag{5.89}$$

We remind that M is the total number of samples. MI is computed as follows:

$$\mathrm{MI}(A, B) = \sum_{i=1}^{N} \sum_{j=1}^{N} h_{ij} \log \left(\frac{M h_{ij}}{h_i^A h_j^B} \right) \tag{5.90}$$

The gradient can be written as follows, according to the chain rule:

$$\frac{\partial \mathrm{MI}}{\partial \xi} = \frac{\partial \mathrm{MI}}{\partial B} \frac{\partial B}{\partial \xi} \tag{5.91}$$

Here $\dfrac{\partial B}{\partial \xi}$ is the same thing as (5.79). Now we need to figure out how MI changes if we change one of values in B:

$$\frac{\partial \mathrm{MI}}{\partial b_k} = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial \mathrm{MI}}{\partial h_{ij}} \frac{\partial h_{ij}}{\partial b_k} \tag{5.92}$$

We may notice that most of terms that contain $\dfrac{\partial h_{ij}}{\partial b_k}$ are zero, because generally b_k contributes to only four bins. So we assume that the programmed version correctly selects necessary h_{ij}

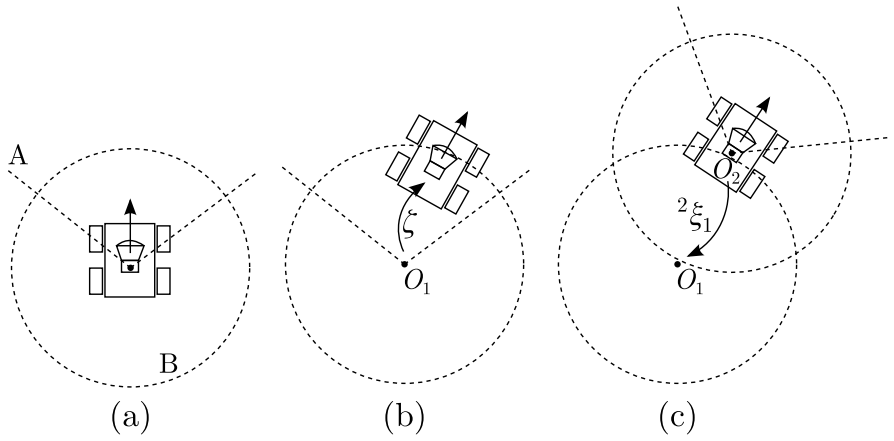


Figure 5.12: Direct visual odometry with keyframes. (a) The first keyframe with origin O_1 is instantiated, it has a certain orientation and distance limits (A and B). There is a depth map $\mathcal{D}_1$ and an image I_1 associated with this frame. (b) While the robot stays within the keyframe limits, its localization ζ with respect to O_1 is computed and used to improve $\mathcal{D}_1$. (c) Once the robot leaves the keyframe limits, a new keyframe is instantiated with origin O_2 and an image I_2, $\mathcal{D}_1$ is projected onto it and is merged with a new depth $\mathcal{D}(I_1, I_2, {}^2\xi_1)$.

to avoid unnecessary computations. Here, on the other hand, we will keep it under this form since it is formally correct and easy to use for demonstration purposes.

$$\frac{\partial \mathrm{MI}}{\partial h_{ij}} = \log\left(\frac{M h_{ij}}{h_i^A h_j^B}\right) + 1 - \frac{h_{ij}}{h_j^B} \tag{5.93}$$

Finally, by differentiating (5.87) we get:

$$\frac{\partial h_{ij}}{\partial b_k} = \left[\phi\left(t_{j-1} - b_k\right) - \phi\left(t_j - b_k\right)\right] \int_{t_{i-1}}^{t_i} \phi(x - a_k)\mathrm{d}x \tag{5.94}$$

3.3 The Complete Localization Loop

A localization loop outline for both registration techniques is the following:

1. Sparse visual odometry is used to initialize the first transformation and a depth map.
2. Using the computed depth map, the next image is registered. The wheel odometry is used as the initial transformation approximation and as scale factor reference.
3. Compute a new depth map using the computed transformation and merge it with the previous one. Go to step 2.

There are multiple ways of implementing such a system. A simplistic experimental system setup has been used in this work. Its purpose is just to demonstrate the practical value of all the theoretical results. The approach is illustrated in Fig. C.15

At any moment, there is an active keyframe which has an associated depth map. All the newly arrived images are registered with respect to it. Then, the depth map is refined using the tracking stereo algorithm. When the robot gets too far away from the active keyframe, a new keyframe is instantiated and the depth map is projected into it. Two thresholds are used: one is for distance, the other is for direction difference, since the camera does not provide 360° view. Then the semi-global-matching algorithm is used to compute a new depth map which is merged with the reprojected one afterwards.

There is an issue of the scale factor drift. In contrast to sparse visual odometry, which does not have any internal state, dense visual odometry keeps the estimation of the depth map. It means that if the depth map is biased, for example all the measurements are underestimated by 10%, then a simple tight coupling with the wheel odometry will not be sufficient, since the translation length is defined by both wheel odometry measurement and the depth map scale. We applied here another solution, based on the assumption that the length estimation given by the wheel odometry is not biased and does not drift over time. It is generally not the case, but it is a fair approximation, since wheel odometry is the only source of scale factor information.

Then the computed motion increment length (not the orientation) is normalized according to the wheel odometry measurement (hard scale-factor normalization).

Another, more involved solution is to include the scale factor into the optimization as an independent variable along with the odometry prior, and correct the depth map at every iteration. This is a particular case of a more general global optimization where the depth map and the motion parameters are estimated simultaneously. In the current implementation, depth estimation and localization are separated.

The results of direct visual localization testing on synthetic data are presented in Fig. C.16. The same data, as for the sparse visual odometry have been used, but the trajectory is twice as long (four laps instead of two), to make the drift visible. Yet the vertical drift, which does not appear in the plot, is about 1 m, which is a significant value. So far, we don't have an ultimate explanation why the vertical drift is much larger. We strongly believe that it is related to the questions of direct SLAM stability, and its relation to the environmental structure and the trajectory. The results of MI-based localization are given in Fig. 5.14.

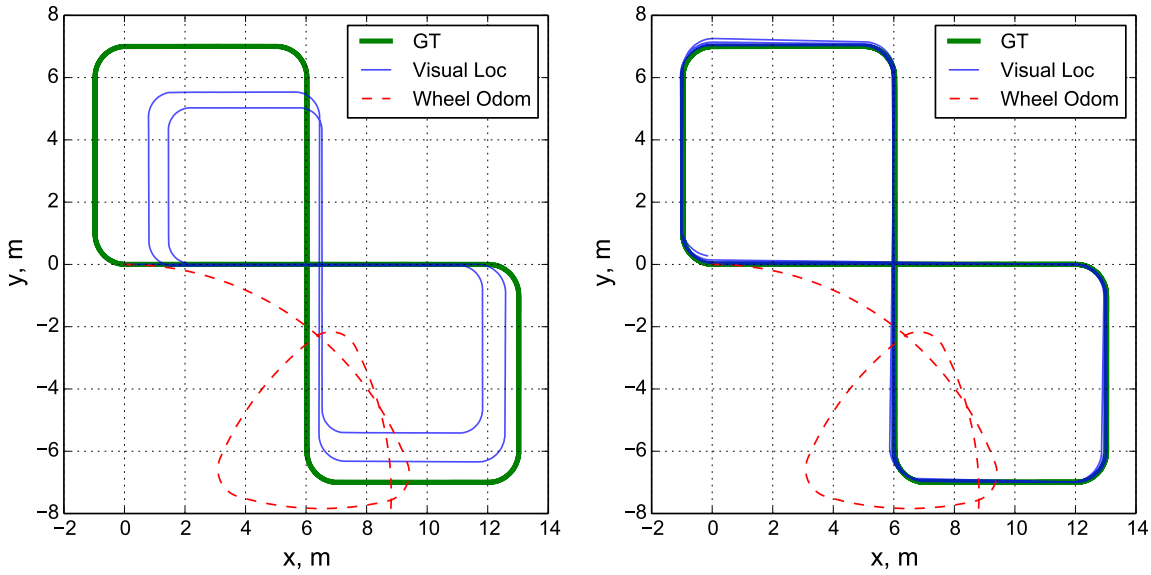


Figure 5.13: Direct visual odometry test with photometric registration. Left — No scale normalization. Right — Hard scale normalization, four 8-shaped laps.

3.4 Stability of Direct Visual Localization

In the original papers (for example Engel *et al.* (2014)), the direct localization system has been proposed and experimentally validated. During our study, we discovered that, depending on numeric parameters and thresholds, the system either converges or diverges. It is important to prove theoretically at least local stability of direct localization methods.

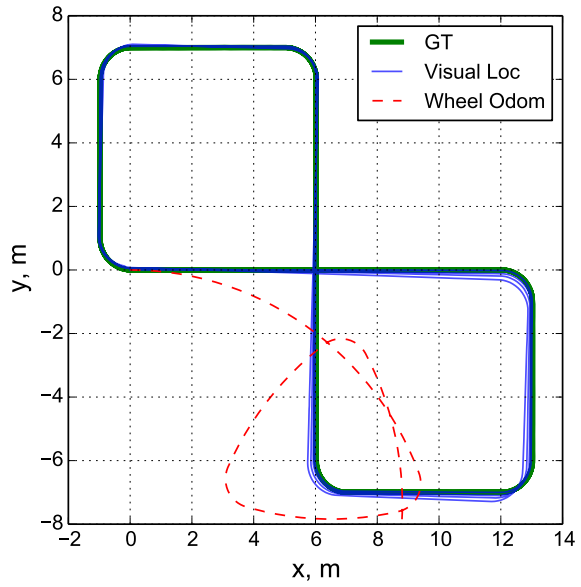


Figure 5.14: Direct visual odometry test with MI-based registration — four 8-shaped laps.

An important difference between feature-based localization and direct localization is that in the first case two images are enough to estimate both feature position and transformation, while in the second case we need at least three images. If we count the number of unknowns and the number of equations we will see the following figures:

- In the case of point-like features, every point gives 2 equations: the reprojection error on the second image along u and v, while it requires only one parameter: the depth in the first image. Hence we need only 5 points to reconstruct the motion up to a translation scale factor.

- The direct method gives a single equation per point for every image beyond the first one: its photometric reprojection error, while the number of unknowns is still one per point. So, if we have only two images the number of unknowns is $N+6$ for N equations, where N is the number of salient points.

We can artificially make the system overdetermined by adding regularization terms, as we did for the stereo reconstruction. Regularization works well in the case when the stereo system is calibrated and the transformation between the cameras is known. In this case, while it introduces a certain bias in the depth estimation, the overall result is so much better with the regularization than without, that we accept the bias as a necessary evil. But the important question is how would this bias interact with the localization part?

Another important question is how the transformation estimation error is propagated. In our system as well as in Engel *et al.* (2014) a feature-based odometry is used to initialize the first transformation. The estimation inevitably contains a certain error. If we use this estimation to compute the depth map, and then use the depth map to compute the next transformation, will this error be amplified or attenuated? In other words, will the error remain bounded?

3.5 Mapping and Localization

The proposed direct visual odometry can be extended to a mapping system. All we need to do is store the keyframes along a trajectory. If the robot is going along a trajectory which is

close to the one used to build the map, it can use the key images to localize itself with respect to the corresponding camera positions. Even though errors in keyframe positions lead to an error in the localization, it is locally consistent with respect to the previous trajectory.

Such a localization can be applied for repeating learned trajectories. For example, an autonomous vehicle should follow a predefined path. This path can be defined by a sensory topological map. If the localization error within each keyframe remains bounded and the keyframes are traversed in the right order, then the robot will arrive to the desired location in the real world even though its own global localization with respect to the starting point will be wrong.

The SLAM system operates in the following way. After the depth map initialization, which takes a certain distance, the system is looking for keyframes. If there are no keyframes in the map so far, or the existing keyframes are too far away, it enters the SLAM mode, which is in fact the direct photometric odometry which stores the keyframes and their absolute coordinates. If the system detects that there is an existing keyframe which is close enough (if there are more than one, it takes the closest one), the system switches to the localization mode. In this mode, the system localizes itself with respect to the keyframe using the MI registration and checks whether it is necessary to change the keyframe or switch to the SLAM mode.

The results obtained for the simulated data are given in Fig. 5.15. The map is built during the first lap, and along the four other laps the system localizes itself with respect to it. The loop closure is not in the scope of this work. For the simulated data, the divergence in the odometry mode is so small, that the last keyframe of a lap and the first one are close enough that the system be able to localize itself with respect to the first keyframe after getting too far away from the last one (Fig. 5.15b).

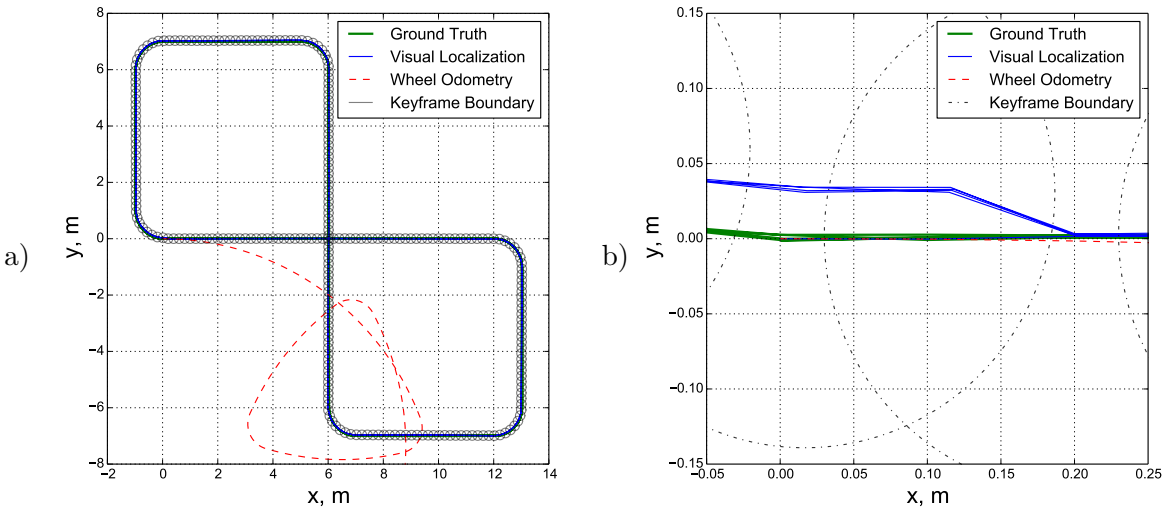


Figure 5.15: Direct SLAM system test on simulated data — five laps. (a) Global trajectory view: the system stably localizes itself within a sensory map that it has built during the first lap. (b) Zoomed view at the origin: at this scale it is visible that both the GT and VO trajectories are not exactly the same, and are noisy. Also, there is a visible transition between the last and the first keyframe which appears because of small odometry divergence.

3.6 Localization Tests with Real Data

To test the precision and robustness of the developed localization technique, the localization scheme has been tested on real data in the following manner:

1. A visual sensory map is constructed using ground truth localization.
2. The localization algorithm is used to localize the vehicle in this map using different recordings along the same trajectory.
3. The localization output is compared to the ground truth.

Data and Environmental Conditions The acquisition was done in a residential area on November 20, 2017 (the map and the first localization datasets) and on November 30, 2017 (the second localization dataset). The mutual-information measure allows us to use a map constructed during a different day with slightly different lighting conditions and other minor changes in the environment, like dead leafs on the ground, differently parked cars, and so on.

The traffic was not dense but there were multiple cars passing by. The speed of the vehicle during the acquisition is between 30 and 50 km/h. The camera frame rate is 20 Hz at 640×480 resolution. Examples of the dataset images are presented in Fig. 5.16. The trajectories are represented in Fig. C.17.

Map Construction The map, as in the previous case, is a sequence of images with associated robot poses in the global frame. The ground truth measurements are obtained using fused odometry, IMU, and RTK GPS at 1 Hz. The map is constructed using three datasets. The construction algorithm is straightforward. For every new image, the system checks whether there is already a keyframe close enough. If not, a new keyframe is instantiated. Thus, the overlapping areas of different datasets are mapped only once. The process is illustrated in Fig. C.18. The resulting map structure is sketched in Fig. 5.18.

Localization The localization pipeline is illustrated in Fig. C.19. The system is initialized with the GT position. The first transformation between camera positions is computed using classical keypoint-based visual odometry. Then this transformation is used to compute the first depth map.

Dense visual odometry (illustrated in Fig. C.15) is constantly used to compute the local motion and to refine the depth map. The local pose with respect to the active keyframe is computed using photometric virtual visual servoing. Then the depth map is used in MI-based virtual visual servoing to compute the relative pose of the active keyframe with respect to the nearest map frame. Two approaches have been used: with and without trajectory smoothing using odometry prior. In the first case, the current pose ξ_{prior} is included into the MI-based localization process as a part of the cost function:

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \; -\mathrm{MI}(\xi) + \lambda \left\| \xi_{\text{prior}}^{-1} \circ \xi \right\| \tag{5.95}$$

In the second case ξ_{prior} is used only for initialization. We have to use λ to equalize the two terms in the cost function because their units are not the same and there is no obvious formally correct way to deal with it. So, λ has been adjusted manually.

The error plots and computed trajectories for different datasets are given in Fig. 5.22–C.21. Fig. 5.22 represents the localization results with the data acquired just after the map data acquisition. Fig. 5.23 contains the results for a different dataset acquired on a different day. Fig. 5.24 and C.21 represent the same thing but with the odometry prior applied for trajectory smoothing. The average errors are summarized in Fig. C.20. The figures demonstrate that precision goes down when we use a sequence from a different day for the localization.

In the case of non-smoothed trajectory, once the system switches the keyframe, the trajectory may undergo a jump because the prior position is not included as an optimization criterion. The absence of smoothing shows the system intrinsic precision and convergence

Figure 5.16: Sample images from the map dataset from 20/11 (a, d), the test datasets from 20/11 (b, e), and the test datasets from 30/11 (c, f). All the three datasets contain moving objects.

Figure 5.17: Trajectories used for the experiment. The approximate lengths are: blue — 650 m, green — 520 m, red — 740 m.

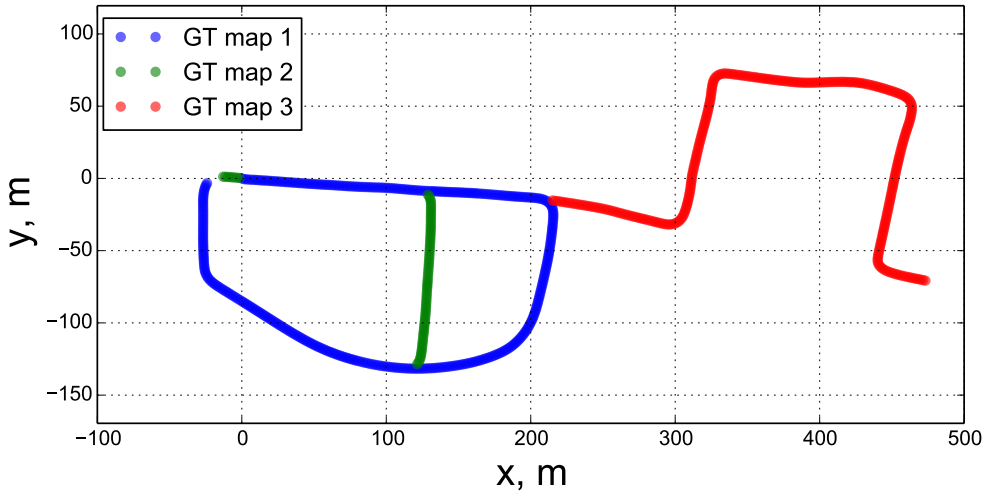


Figure 5.18: The built map. 3 different datasets were used. Colors indicate which dataset the keyframes have been taken from.

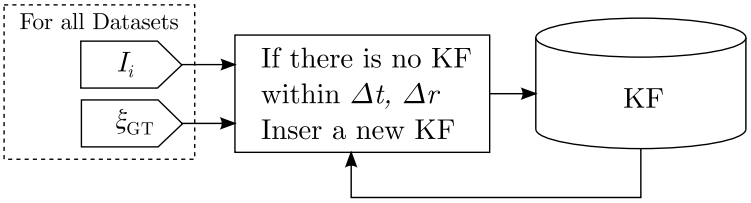


Figure 5.19: Map construction process. I_i and $\xi_{\mathrm{GT},i}$ are measurements: the current image and the Ground Truth localization measurement. Δt and Δr are scalar rotation and translation thresholds for map construction. In our application they are 1.5 m and 1 rad respectively.

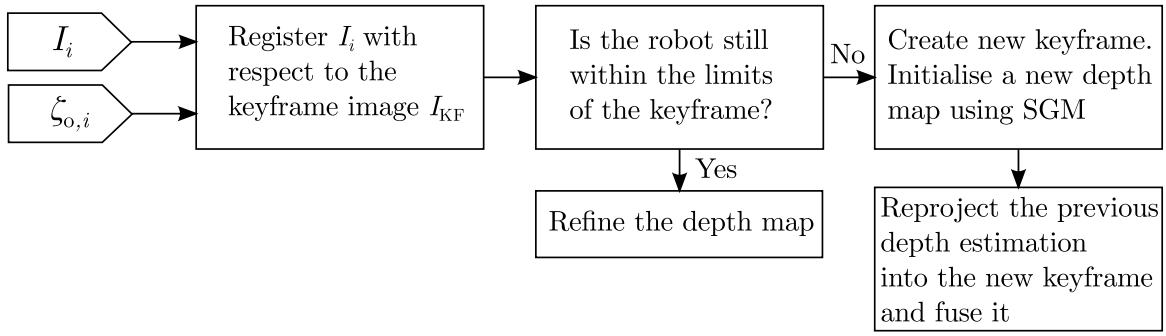


Figure 5.20: Localization pipeline. I_i and $\zeta_{GT,i}$ are the current image and the odometry increment since the last image.

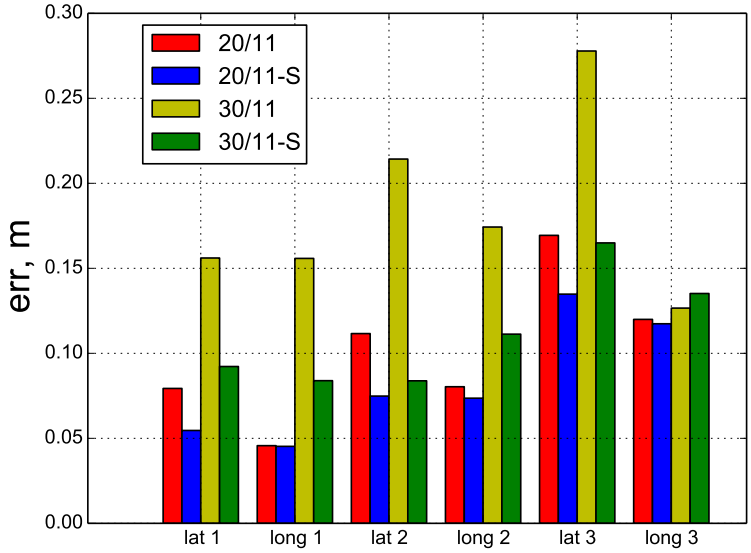


Figure 5.21: Average error comparison for different datasets. 20/11 and 30/11 are the acquisition dates; "-S" stands for odometry prior smoothing. "lat" — lateral error, "long" — longitudinal error.

limits. Even though in some cases the error goes as high as 1.5 m, the system does not get lost and manages to recover and return the actual trajectory.

Trajectory smoothing, on the other hand, obviously improves the localization results in almost all the cases. It removes error spikes and makes the trajectory less noisy. The error standard deviation remains about 10-15 cm, which is a good precision for an absolute visual localization system.

We insist once again on the fact that there is neither explicit feature detection and matching, nor explicit moving object detection. The latter can improve the robustness and localization quality even further.

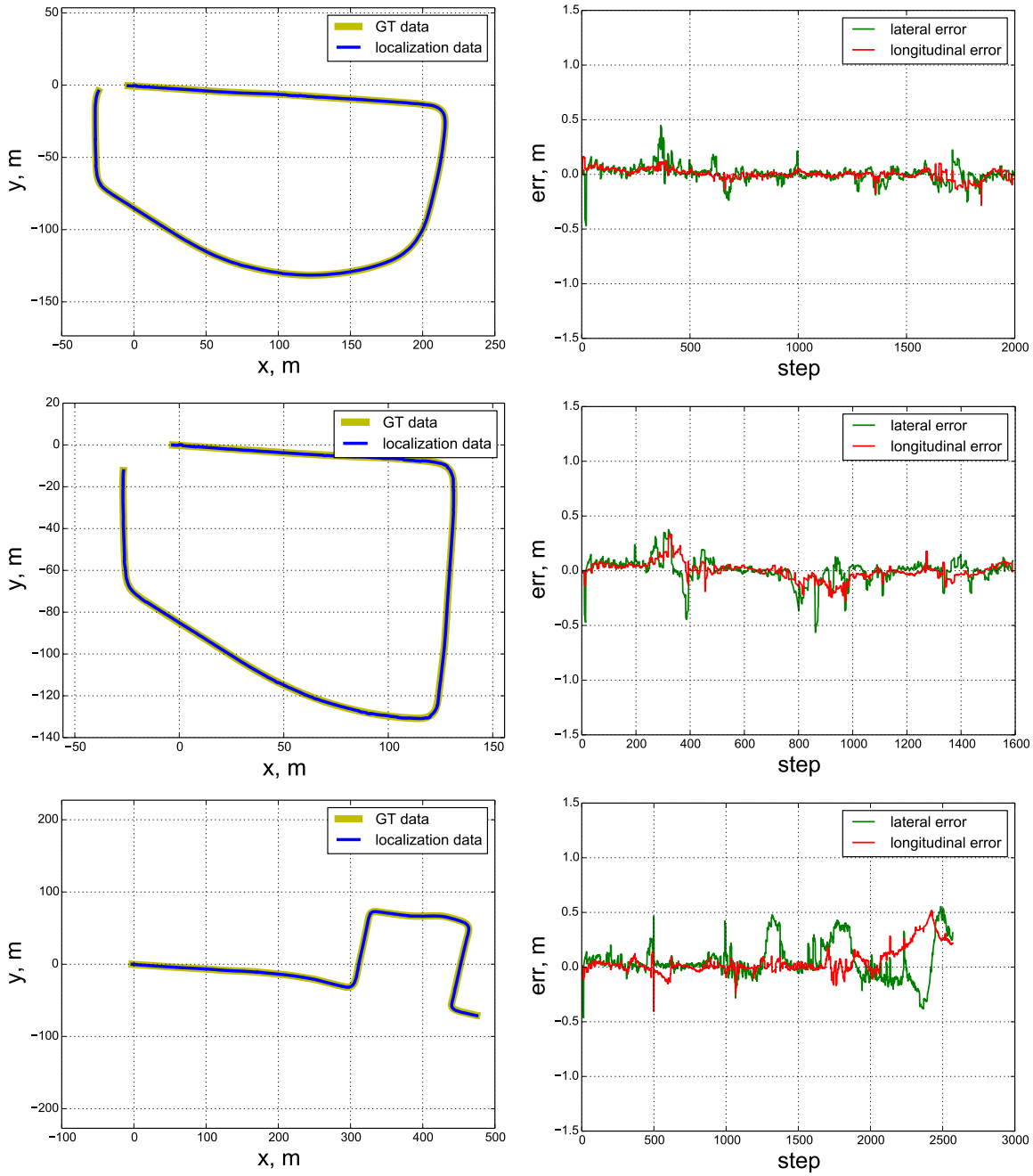


Figure 5.22: Localization results — Reconstructed trajectories and the corresponding localization errors in the robot's frame. Localization data has been acquired within a half hour after the map data acquisition. Poor quality at the end of the third dataset is due to a drop in GT precision (GPS in "float" mode).

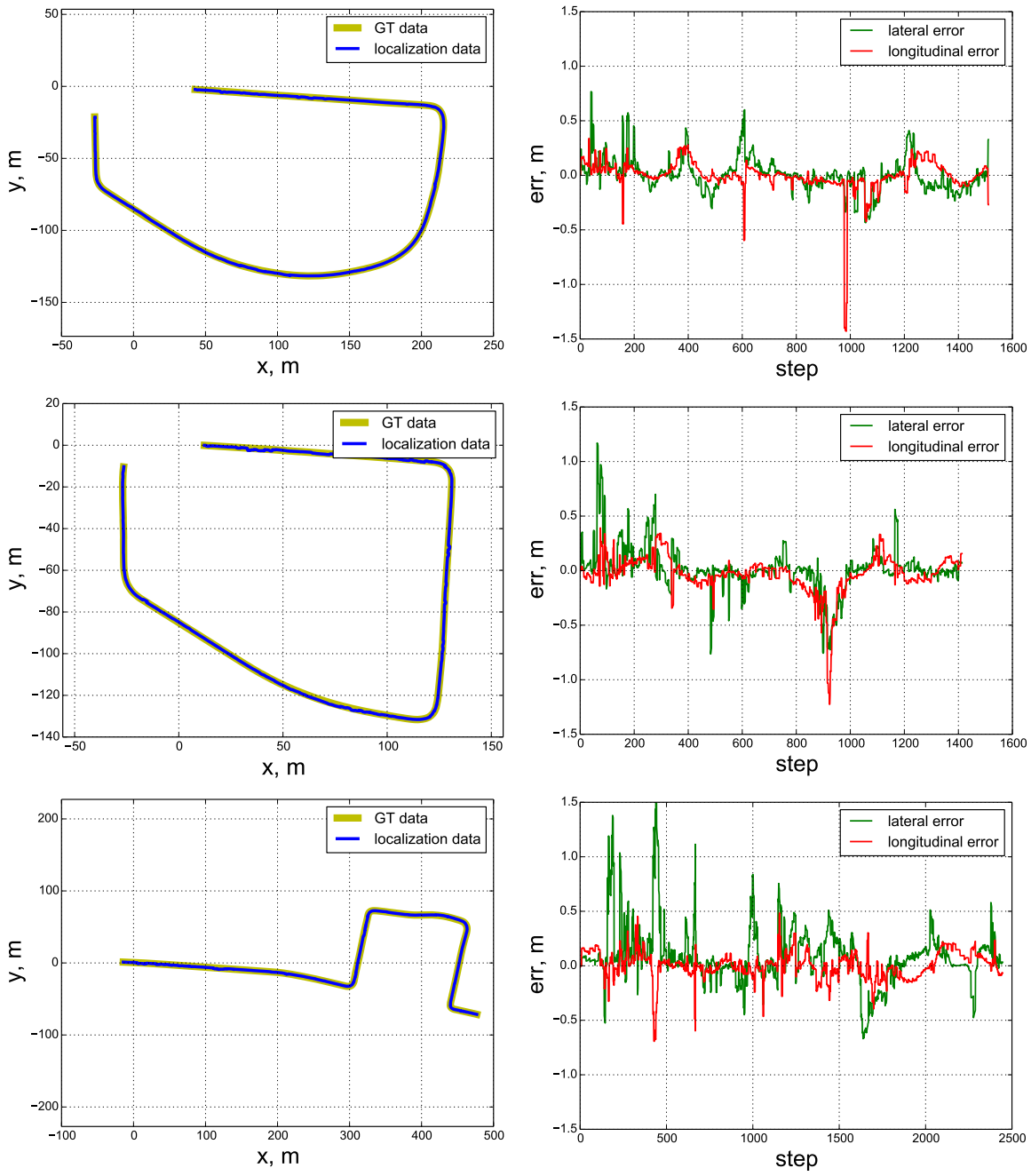


Figure 5.23: Localization results — reconstructed trajectories and the corresponding localization errors in the robot's frame. Localization data has been acquired 10 days after the map data acquisition.

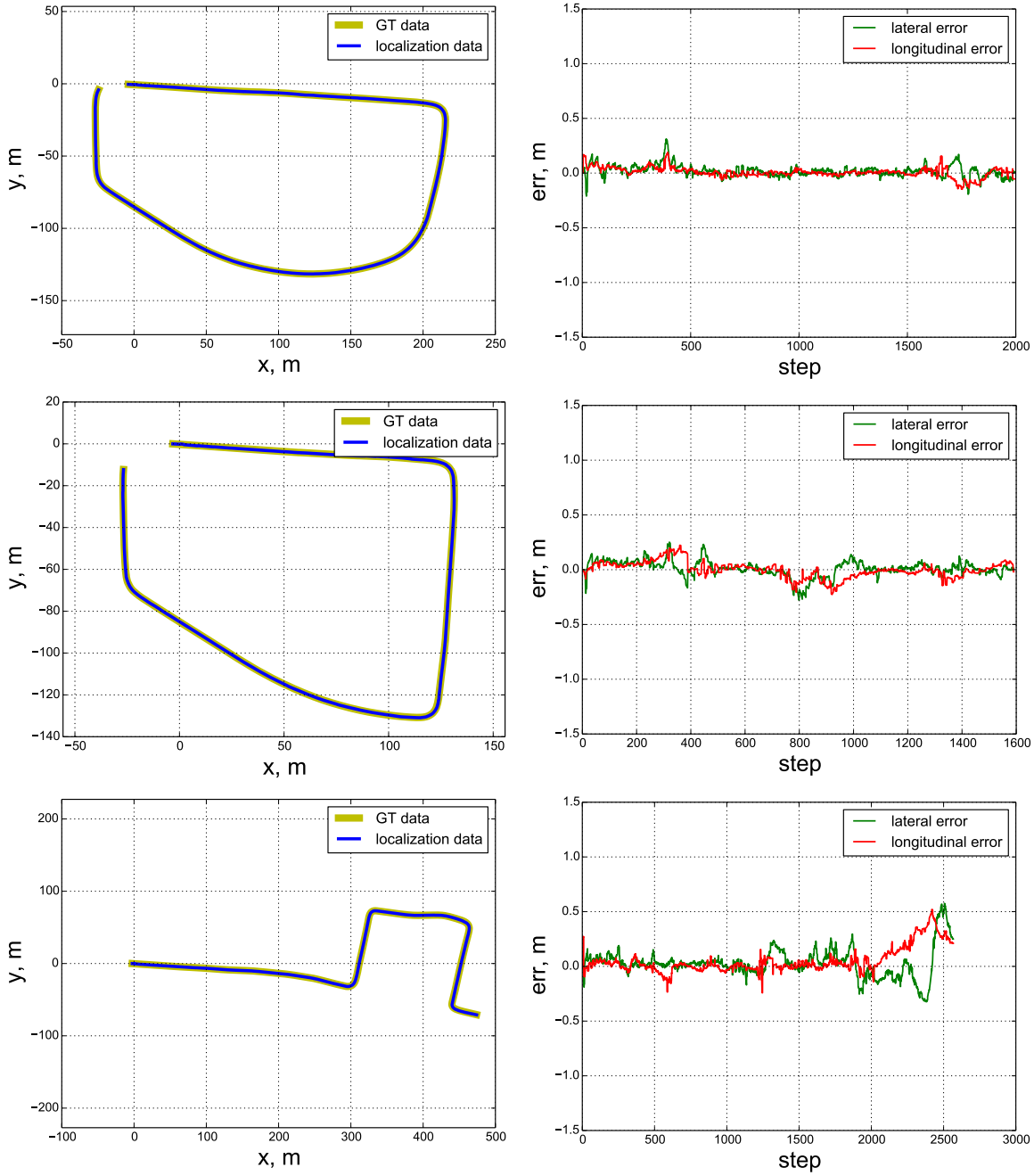


Figure 5.24: Localization results — reconstructed trajectories and the corresponding localization errors in the robot's frame using trajectory smoothing. Localization data has been acquired within a half hour after the map data acquisition. Poor quality at the end of the third dataset is due to a drop in GT precision (GPS in "float" mode).

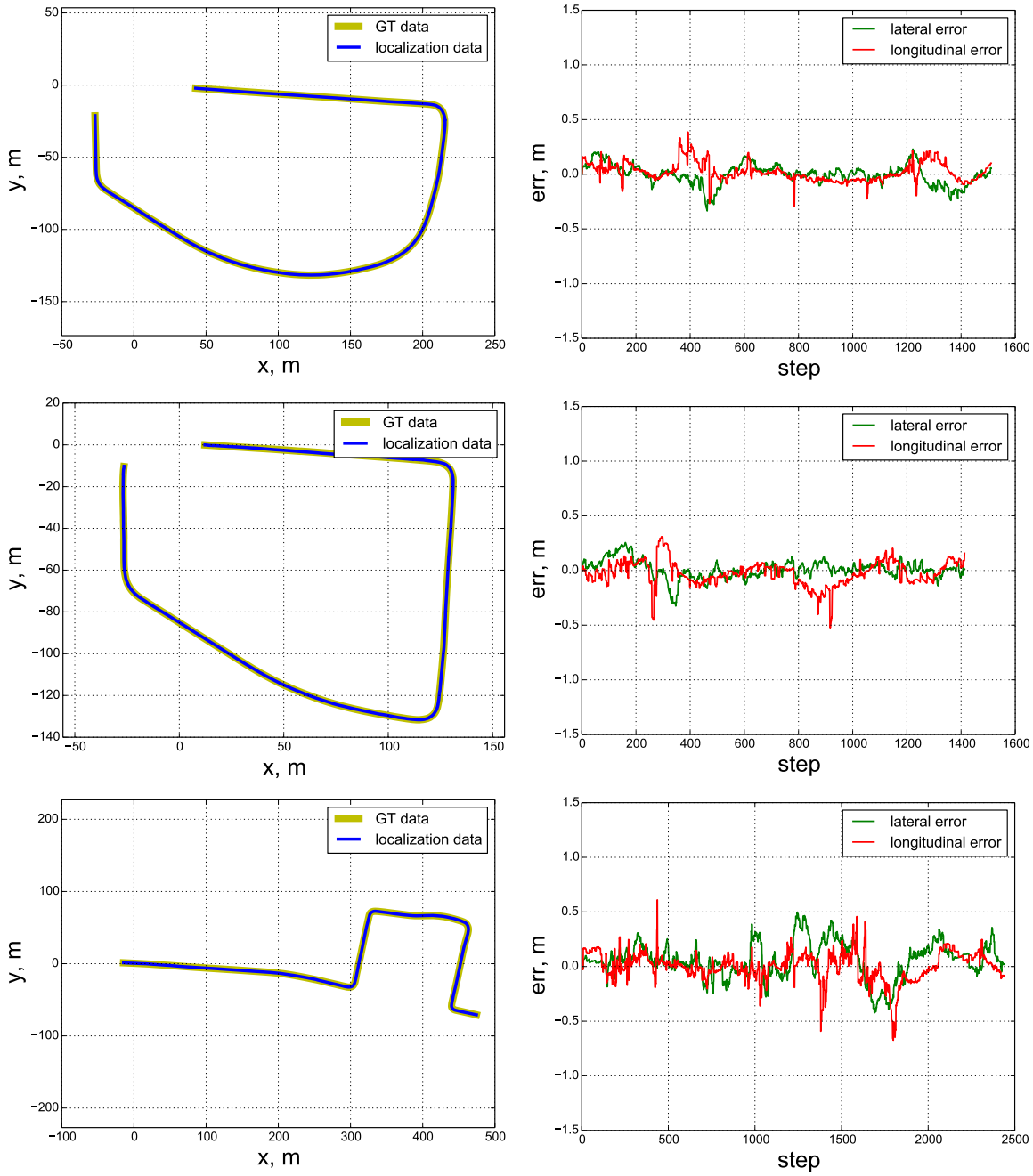


Figure 5.25: Localization results — reconstructed trajectories and the corresponding localization errors in the robot's frame using trajectory smoothing. Localization data has been acquired 10 days after the map data acquisition.

4 Conclusions

Several different localization systems based on fisheye cameras and wheel odometry have been implemented and tested using synthetic data. The main accent is made on the use of direct image registration with photometric and Mutual-Information-based criteria.

A visual localization system, which uses virtually all the concepts introduced in the thesis, has been implemented and tested on real data. It showed robustness with respect to environmental changes, since the localization has been done using data acquired 10 days after the map data acquisition. Moreover, it is robust with respect to moving objects, such as cars and pedestrians.

Photometric registration appears to be more computationally efficient, but it can be applied only when the images to be registered are taken within a short period of time in very similar lighting conditions. On the other hand the MI criterion allows us to register images taken on different days.

Below, we discuss the main contributions in more details and suggest some ideas about possible system improvements.

4.1 Contributions

Regular Triangulation The proposed approach avoids the reconstruction singularities in the case when the point is far away and the direction vectors are close to parallel. It leads to a significant reduction of the number of unknowns from $6 + N$, where N is the number of keypoints, to six, or just a single transformation.

Tight Wheel and Visual Odometry Coupling We included the wheel odometry into the optimization loop during RANSAC. This improves the convergence since we start close to the true minimum. In addition, we reduce the number of keypoints needed to fit the model, since the wheel odometry already constrains certain degrees of freedom. It works in a similar way to the 1-point RANSAC algorithm (Civera *et al.* (2009)), but instead of using an Extended Kalman Filter, the odometry measurement is used as a prediction. Moreover, the system becomes more resistant to moving objects. Even though the visual localization might stick to a moving object, like a bus or a tram, coupled with wheel odometry it has to choose features static with respect to the ground. Finally, combining the two sources of information in the same optimization loop gives us an optimal state estimation using all the information available.

Direct Visual Localization using Fisheye Cameras It is demonstrated that the Enhanced Unified Camera Model can be successfully used to implement a fisheye-based direct localization system. Its advantages are its simple analytical form and simple projection Jacobian matrix, which make it computationally efficient for both registration and stereo matching.

MI for Vision-Based Localization Mutual information has been successfully used for visual servoing and mobile robot navigation before. But there is a major difference between those applications and the one proposed here:

- In image-based visual servoing only the Jacobian matrix depends on the 3D model, but not the error. Moreover, pixel depth values only scale the Jacobian's part that relates the translational motion with the reprojection change.
- In the case of visual localization and image registration the final error depends on 3D reconstruction. If the depth estimation is wrong, then zero photometric residual or MI maximum does not imply zero error in the estimated transformation.

We have demonstrated that it is possible to use MI for dense visual odometry. In the real-data tests a hybrid system has been used. The photometric registration has been used for image registration with respect to the local keyframe for the visual odometry purposes, while MI-based registration has been used for localization with respect to a previously built map.

4.2 Future Development

Topological Mapping The mapping system presented in this work is just a sketch of a real topological mapping system. To make it fully operational, a few things must be added:

- The keyframes must be stored as a graph with neighbor nodes localized with respect to one another, not just as a collection of robot poses with associated images. Like that, when we are using one keyframe for the localization, we know that we should look for the next keyframe among its neighbors, not among all keyframes.
- The keyframe radius must be increased. For that, both the reconstruction and registration algorithms must be improved and better adjusted.
- The system must be able to automatically connect close keyframes which have not been marked as topological neighbors at their construction.

Loop Closure To make a mapping system complete, the loop closure is necessary. The classical FAB-MAP Cummins and Newman (2008) would be sufficient. It remains to be integrated with the rest of the system. In contrast to metric localization systems, here loop closure is necessary for topological consistency, rather than for metric precision improvement, since loop closure generally does not change relative transformations between frames much.

Stability Analysis During simulation tests, it has been observed that the depth estimation and the motion increment either remained bounded and consistent, with a certain increment error, or diverge up to complete depth map deterioration and practically random motion estimation. The stability depends, in particular, on the following factors:

- distance threshold for new keyframe instantiation
- uncertainty increment for the depth map when it is projected into another frame
- localization cost function robust norms

Stability analysis is the most fundamental question about the subject of direct visual localization and mapping.

Moving Object Detection and Tracking For localization and mapping systems, it is highly important to be able to deal with motion in the environment and do it explicitly (like in Wang *et al.* (2007)). State-of-the-art direct localization systems, to our best knowledge, still do not handle the environmental dynamics explicitly and assume the static-world hypothesis. Though outlier rejection paradigm is robust to environmental dynamics to a certain extent, an explicit motion tracking is necessary for robotic applications such as self-driving cars.

Chapter 6

Conclusions

This thesis presents a novel projection model for fisheye cameras, which is mathematically simple and yet shows high precision when applied to real cameras. Geometric properties of the model have been analyzed using the concept of projection surface, introduced in this work. In particular, a closed-form inverse mapping and an implicit equation for straight line projection have been found. This fact has been used to develop a method of direct stereo correspondence on raw fisheye images via implicit curve rasterization. This correspondence algorithm allow us to apply the Semi-Global Matching algorithm to get accurate and regular 3D reconstruction using fisheye stereo systems. All these elements have been shown to be applicable to a direct visual localization system with two different methods of image registration: direct photometric error minimization and mutual information maximization. Intrinsic and extrinsic calibration of a mobile robot with fisheye cameras has been considered and a toolbox for such a calibration has been developed.

In this conclusion, the main contributions of this work are summarized and a few research tracks are proposed. Then, a few more words are said about a more global vision of a visual localization system.

1 Main Contributions

In this work, the motivation was to find a way to model fisheye cameras so that it would be possible to treat raw fisheye images without rectification. We would like a camera model to have the following properties:

1. Simple analytical form to make it computationally efficient.
2. Precision in modeling real wide-angle cameras.
3. Closed-form inverse mapping.
4. Straight line projection equation.

The proposed model combines all these properties, which makes it possible to apply it in different contexts. In particular, the epipolar constraint can be expressed directly in the image space for fisheye cameras.

The next step was to develop an algorithm of direct fisheye stereo correspondence using the novel model. It has been done via rasterization of epipolar curves directly in the image space. This approach shows precise metric results for both simulated and real data, which validates simultaneously the theoretical foundation of the algorithm and calibration quality attained with the Enhanced Unified Model. This algorithm is adapted for both dense and semi-dense stereo correspondence.

Visual Localization Regularized reconstruction for efficient robust visual odometry has been developed. In turn, it allows us to develop a feature-based odometry with implicit reconstruction. Instead of explicitly optimizing the 3D feature point position, only the transformation between the two camera positions is computed. It reduces the number of variables to 6. Another important concept is to combine the wheel odometry and visual odometry to speed up and robustify RANSAC. The concept of 1-point RANSAC has been used and the number of data points used in RANSAC has been reduced to 2.

Two types of direct registration (that is, without feature detection) based on semi-dense 3D reconstruction have been implemented. Photometric registration is used to register consecutive frames of a video sequence. Mutual-information-based registration is used to register two images taken on different days. These techniques have been used to develop a visual SLAM system, which combines all the theoretical results developed before and shows a good performance on synthetic data.

The localization technique has been tested on real data acquired in a residential area. The system is able to compute the localization using a map constructed on a different day. It demonstrates robustness with respect to moving objects (cars, pedestrians) and environmental changes (slightly different lighting, displaced trash cans, opened/closed windows, differently parked cars and so on).

Calibration Toolbox All along this thesis, a calibration toolbox has been developed. So far it includes the following possibilities:

1. Monocular camera calibration.
2. Multiple-cameras system (usually stereo) extrinsic calibration.
3. Odometry-cameras extrinsic calibration based on odometry measurements and observation of a static calibration board only.

The toolbox is implemented in such a way that any combination of these calibration problems can be solved, once they are properly described in a calibration configuration file. The performance of this calibration allows its use in real time. For example, a problem with 789 images, 4736 parameters and 88160 residuals is solved in 0.91 s. It is possible thanks to Ceres Solver by Agarwal *et al.* (2010). The most time-consuming part is the image board detection. It takes 34.8 s to detect board images for the same problem (or about 20 images per second).

2 Limitations and Future Developments

Direct Fisheye Stereo

1. There are multiple different depth map regularization techniques based on image information and some other models like minimum object size. Applying them can provide us with a better depth estimation.
2. Image driven weights in the dynamic programming part can improve depth map consistency. But this question is yet to be studied to quantify the effect. An experimental version is used in the actual implementation, but the current model is based on pure intuition, while it would be better to base it on a statistical analysis of ground truth images and the correlation between image gradient and depth transitions.
3. Every step of the algorithm is completely parallelizable, so CUDA implementation would make it more practical as it would be possible to use it in real time.

Mapping

1. Data structures to manage topological maps are missing, so far all the frames are stored in absolute coordinates, and it is not what we want.

2. Keyframe radius, which defines the width of the map along the path, must be increased. It would facilitate localization and relocalization, as well as topological map construction. On the other hand it requires accurate analysis of localization convergence properties for different distances from the current keyframe origin.

3. A smarter criterion for switching to a new keyframe, instead of a simple distance threshold, should be applied. For example, the singular values of the Hessian matrix of the cost function at the optimum give us some information about localization precision.

4. A loop closure system must be added to enforce topological consistency of constructed maps.

5. The Stability of dense and semi-dense localization systems has to be studied and stability conditions (at least local) must be found in order to be able to apply this type of algorithms in the real world.

Calibration Toolbox The following improvements can be done:

1. Automatically select calibration images or weight data samples to equalize the data point distribution across the image space.

2. Provide metric analysis for the final error by projecting it back onto the calibration board.

3. Add IMU and GPS calibration to the toolbox.

4. Calibrate wheel odometry intrinsic parameters, such as wheels' radii and track gauge.

5. Calibrate measurement noises for all the sensors.

3 Ideas About a Complete System

Let's try to imagine what a complete vision-based localization system could look like. This system should be able to perform the following tasks simultaneously:

1. Wide-angle vision for a better surrounding perception

2. 3D reconstruction of the environment

3. Metric localization with respect to local maps

4. Detection and Recognition of navigable space, obstacles, road signs

5. Detection and tracking of moving objects

6. Topological map construction, topological localization and navigation

7. Loop closure based on distinctive mid- and high-level features

It can be done by integration of geometric computer vision with deep learning. The main advantage of geometric vision is that projection models are relatively simple in comparison to models used in machine learning. They have much fewer parameters (about ten, in contrast to approximately sixty million parameters of the famous AlexNet by Krizhevsky *et al.* (2012)), which allows us to identify them with high precision and be sure about the performance across the whole space. Such techniques as bundle adjustment and dense stereo reconstruction require simple and precise analytical projection models. The stereo reconstruction and localization algorithms presented in this work intensively use geometric modeling. Bundle adjustment is, so far, the most precise way of environment reconstruction. A recent paper by Engel *et al.* (2018) presents a localization algorithm, where bundle adjustment is done on a sparse set of points using directly photometric information.

The weakness of geometric vision is that it requires geometric information, while images provide us with photometric information. And the conversion from photometric to geometric is the source of most errors. Even though direct methods address this problem, there will always be ambiguous data where accurate solution using general methods is impossible given no prior assumptions.

Machine learning (and in particular, deep learning), on the other hand, provide us with the possibility to generalize data, acquired in the real world, to extract some "experience", prior knowledge. Some problems are relatively easy in the presence of such prior knowledge, while being quite ambiguous and complex from the general point of view. If we take 3D reconstruction as an example, general regularizers favor depth maps with as few depth transitions as possible. And yet, for some cases (for example a metal grid fence), this regularization assumption is not the best one. But if only we could recognize the grid and apply a different regularizer, adapted for this particular case, the result would be much better.

Another example is moving object detection, which can be done based only on geometric constraints. But there are objects which are likely to be moving (cars, pedestrians), and there are ones which are expected to be static (trees, light posts, road signs). Also, there are some singular configurations when the motion of the object, its size and position become dependent, and it is not possible anymore to estimate all three quantities simultaneously. Again, prior knowledge about some of these values can improve the reconstruction quality.

Unfortunately, there is a tendency in the community to stick to one or the other approach. Either to use simple analytical models, or to rely on model-free deep learning architectures. Reconciling both seems to promise a lot of benefits. A recent example of such reconciliation is presented by Tateno *et al.* (2017), who has combined a dense direct SLAM algorithm with a machine-learning-based 3D reconstruction prior. Another example has been mentioned in the introduction, Bhatti *et al.* (2016) combined deep reinforcement learning with classical SLAM technologies and neural-network-based object recognition.

We believe that these two approaches are not in competition but rather complementary, and each of them should take its place in solving hard computer vision problems.

Appendix A

Spatial Transformations and Motion

Mathematical description of rigid body motion and position is one of the theoretical foundations of robotics. The principal notations and definitions for this subsection are illustrated in Fig. A.1.

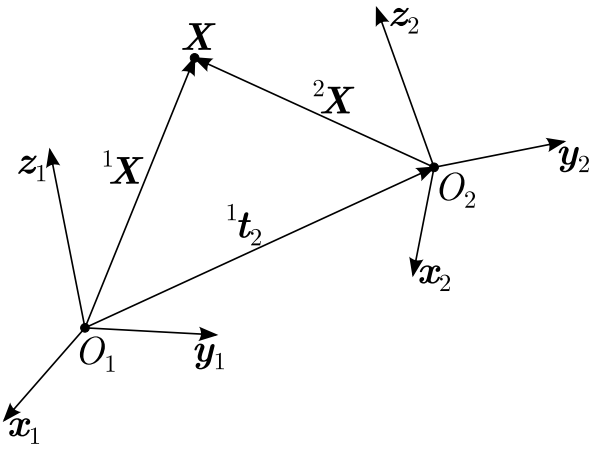


Figure A.1: Frames and relations between them.

1 Frame Definition

A coordinate frame is defined by its origin position O_i and three orthonormal vectors $\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{z}_i$. Let us refer to different frames by its origin's name. Let $\boldsymbol{X}$ be a spatial 3D point. Then ${}^i\boldsymbol{X} \in \mathbb{R}^3$ is its coordinates or its projection into O_i. We say that O_i is the projection frame of ${}^i\boldsymbol{X}$. The latter is defined as follows:

$$ {}^i\boldsymbol{X} = \begin{pmatrix} (\boldsymbol{X} - O_i) \cdot \boldsymbol{x}_i \\ (\boldsymbol{X} - O_i) \cdot \boldsymbol{y}_i \\ (\boldsymbol{X} - O_i) \cdot \boldsymbol{z}_i \end{pmatrix} \tag{A.1} $$

If the projection frame is not precised, then the result is invariant with respect to the projection frame. Sometimes across the text $\boldsymbol{X}$ may mean coordinate expression in a particular frame, which is omitted since it is clear which one it is. It allows us to keep the equations liter.

2 Homogeneous Transformation Matrix

Frames are defined with respect to other frames via spatial transformations, which allow us to change projection frames of spatial points. These transformations have 6 degrees of

freedom: three translational and three rotational. One possible way of parameterizing spatial transformations is by a rotation matrix and a translation vector. Let O_1 and O_2 be two coordinate frames in $\mathbb{R}^3$ and let $\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{z}_1$ and $\boldsymbol{x}_2, \boldsymbol{y}_2, \boldsymbol{z}_2$ be their basis vectors.

Translation Let ${}^1\boldsymbol{t}_2 \in \mathbb{R}^3$ be the position of O_2 with respect to O_1, it is defined as in (A.1):

$$ {}^1\boldsymbol{t}_2 = \begin{pmatrix} (O_2 - O_1) \cdot \boldsymbol{x}_1 \\ (O_2 - O_1) \cdot \boldsymbol{y}_1 \\ (O_2 - O_1) \cdot \boldsymbol{z}_1 \end{pmatrix} \tag{A.2} $$

Rotation Matrix Let R be an element of the special orthogonal group SO(3). It is a 3×3 matrix with the following property:

$$ RR^T = R^T R = I \tag{A.3} $$

R defines a rotation of a rigid body about the origin of 3D Euclidean space. Rotation matrix ${}^1R_2$, which defines the rotation between two frames O_1 and O_2 is defined as follows:

$$ {}^1R_2 = \begin{pmatrix} {}^1\boldsymbol{x}_2 & {}^1\boldsymbol{y}_2 & {}^1\boldsymbol{z}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1 \cdot \boldsymbol{x}_2 & \boldsymbol{x}_1 \cdot \boldsymbol{y}_2 & \boldsymbol{x}_1 \cdot \boldsymbol{z}_2 \\ \boldsymbol{y}_1 \cdot \boldsymbol{x}_2 & \boldsymbol{y}_1 \cdot \boldsymbol{y}_2 & \boldsymbol{y}_1 \cdot \boldsymbol{z}_2 \\ \boldsymbol{z}_1 \cdot \boldsymbol{x}_2 & \boldsymbol{z}_1 \cdot \boldsymbol{y}_2 & \boldsymbol{z}_1 \cdot \boldsymbol{z}_2 \end{pmatrix} \tag{A.4} $$

In other words, columns of ${}^1R_2$ are basis vectors of O_2 expressed in O_1.

Homogeneous Transformation Matrix Let T be an element of the special Euclidean group SE(3). It defines a transformation between two frames with different origins and different axes' orientation. For two frames O_1 and O_2 it is defined as follows, using (A.2) and (A.4):

$$ {}^1T_2 = \begin{pmatrix} {}^1R_2 & {}^1\boldsymbol{t}_2 \\ 0\,0\,0 & 1 \end{pmatrix} \tag{A.5} $$

Transformation matrices can be combined as follows:

$$ {}^1T_2 {}^2T_3 = {}^1T_3 \tag{A.6} $$

T is used to change the projection frame of spatial points:

$$ {}^1T_2 {}^2\boldsymbol{X} = {}^1\boldsymbol{X} \tag{A.7} $$

In order to make (A.7) possible we must use homogeneous coordinates of ${}^2\boldsymbol{X}$ which is the three normal coordinates and a 1 concatenated to them which looks like $(x, y, z, 1)^T$. It makes (A.7) equivalent to the following form:

$$ {}^1\boldsymbol{X} = {}^1R_2 {}^2\boldsymbol{X} + {}^1\boldsymbol{t}_2 \tag{A.8} $$

Usually, a special notation is used to distinguish homogeneous form and normal 3-components form of vectors. But in this text the homogeneous form is not used, so we do not have a specific symbol for it. Instead we use ${}^1\xi_2$ to denote a transformation, referring to a non-redundant transformation parametrization with 6 numbers:

$$ {}^1\xi_2 = \begin{pmatrix} {}^1\boldsymbol{t}_2 \\ {}^1\boldsymbol{r}_2 \end{pmatrix} \tag{A.9} $$

Here $\boldsymbol{t}$ is the translation and $\boldsymbol{r}$ refers to a rotation parametrized with 3 numbers. And we use ${}^1\xi_2({}^2\boldsymbol{X})$ to denote the projection frame change. The simplest way to compute transformed points is via transformation matrix ${}^1T_2$ computed from ${}^1\xi_2$.

3 Angle-Axis Parametrization

Since the main numeric tool in this work is nonlinear optimization, having a non-redundant rotation parametrization is absolutely necessary. If the parametrization is redundant, there are some additional constraints, violating which leads to a non-valid rotation. Examples of redundant representation are unitary quaternions with one constraint and rotation matrices with six constraints. In this case, if these constraints are not respected during the optimization, the whole process might break down, since its rotation-ness is used in the cost function calculation. Respecting them, on the other hand, requires extra cost functions, which would lead to a worse performance.

Another solution is just using non-redundant parametrization. In this case, the optimizer may change the parameter values as it wants without any risk of divergence.

There are multiple different non-redundant rotation parametrizations. In this work, we use only the so-called angle-axis parametrization $\boldsymbol{r} = \boldsymbol{u}\vartheta$. $\boldsymbol{u}$ is a unitary vector which defines the axis of rotation and ϑ is the rotation angle in radians. We may refer to $\boldsymbol{r}$ as "rotation vector", even though it is not a vector in the exact sense of this word because generally $\boldsymbol{r}_1 + \boldsymbol{r}_2$ is meaningless. The best way of thinking of $\boldsymbol{r}$ is a set of three real numbers which parametrize the rotation.

This parametrization does not have singularities and it is not unique only when $\vartheta = \pi$. It can be converted into rotation matrix via the matrix exponent of its cross-product matrix:

$$R = \exp\left([\boldsymbol{r}]_\times\right) \tag{A.10}$$

In the case of an exponent of a cross-product matrix, using its definition we can find a different, simpler form, which is called the Rodrigues' rotation formula:

$$R = I + \sin\vartheta[\boldsymbol{u}]_\times + (1 - \cos\vartheta)[\boldsymbol{u}]_\times^2 \tag{A.11}$$

4 Kinematic Screw and Transformation Time Derivative

If frame O_2 moves with respect to frame O_1, its motion can be parametrize with two vectors: linear velocity $\boldsymbol{v}_2$ and angular velocity $\boldsymbol{\omega}_2$. Together they form the so-called kinematic screw. These two vectors can be expressed in either frame, and during any mathematical demonstration one should take a particular care to make sure that the projection frames of all vectors and kinematic screws are clear.

If linear velocity $\boldsymbol{v}_2$ is expressed in the frame with respect to which O_2 moves (in this case in O_1), it can be integrated directly to get the frame origin position:

$${}^1\boldsymbol{t}_2(\mathrm{t}) = {}^1\boldsymbol{t}_2(0) + \int_0^{\mathrm{t}} {}^1\boldsymbol{v}_2(\tau)\mathrm{d}\tau \tag{A.12}$$

On the other hand integration of $\boldsymbol{\omega}$ does not give any meaningful value. What can be integrated is rotation vector time derivative $\dot{\boldsymbol{r}}$. There is a linear transformation between $\dot{\boldsymbol{r}}$ and $\boldsymbol{\omega}$:

$${}^1\dot{\boldsymbol{r}}_2 = M_{\boldsymbol{\omega}}({}^1\boldsymbol{r}_2){}^1\boldsymbol{\omega}_2 \tag{A.13}$$

Notice that all $\dot{\boldsymbol{r}}$, $\boldsymbol{r}$, and $\boldsymbol{\omega}$ are projected to the same frame, with respect to which the moving frame moves. The expression for $M_{\boldsymbol{\omega}}$ is the following one:

$$M_{\boldsymbol{\omega}}(\boldsymbol{r}) = I - \frac{\vartheta}{2}[\boldsymbol{u}]_\times + \left(1 - \frac{\operatorname{sinc}\vartheta}{\operatorname{sinc}^2(\vartheta/2)}\right)[\boldsymbol{u}]_\times^2 \tag{A.14}$$

However, usually we are interested in the inverse mapping since we want to know what would be $\boldsymbol{\omega}$ if we start changing $\boldsymbol{r}$:

$$^1\boldsymbol{\omega}_2 = M_{\boldsymbol{r}}(^1\boldsymbol{r}_2)\,{}^1\dot{\boldsymbol{r}}_2 \tag{A.15}$$

$$M_{\boldsymbol{r}}(\boldsymbol{r}) = I + \frac{\vartheta}{2}\mathrm{sinc}^2\left(\frac{\vartheta}{2}\right)[\boldsymbol{u}]_\times + (1 - \mathrm{sinc}\,\vartheta)[\boldsymbol{u}]_\times^2 \tag{A.16}$$

In the text the argument $\boldsymbol{r}$ can be omitted to avoid unnecessary parentheses.

Appendix B

Image Rendering

Synthetic images are used to validate different algorithms across this work. That is why we provide a short description of the image rendering pipeline. It consists of two stages: geometric processing and texture processing.

This system is quite simple, there is no light modeling, no normal processing; the constant brightness constraint is perfectly respected. All the objects are rectangular planes with a certain texture. The main feature is an anisotropic anti-aliasing filter.

1 Geometric Processing

At this step, for each pixel, it is computed which point of which plane is observed. Two buffers are computed: depth buffer and plane index buffer.

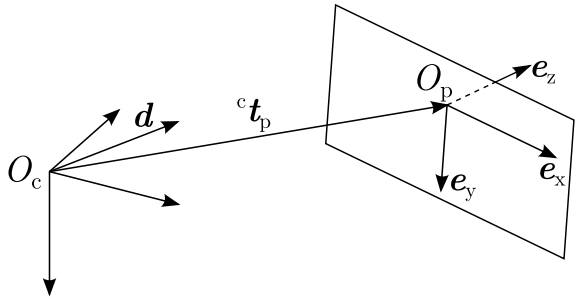


Figure B.1: Computation of the intersection between a given direction ray, represented by vector d, and an arbitrary plane with position ${}^c\boldsymbol{t}_p$ and orientation defined by its frame O_p

For each pixel, the direction ray is computed. Then for every object, it is checked, whether the ray intersects it. It is done by first checking whether the ray is directed towards the plane. Then the intersection coordinates are computed. If they lie within the rectangular textured area, then the depth is checked. If the depth is less than the one stored in the depth buffer, then the new depth value, object index, and the intersection coordinates are stored.

To find the intersection coordinates, first the distance to the plane in a given direction $\boldsymbol{d}$ must be computed:

$$\lambda \boldsymbol{d} \cdot \boldsymbol{e}_z = {}^c\boldsymbol{t}_p \cdot \boldsymbol{e}_z \tag{B.1}$$

where ${}^c\boldsymbol{t}_p$ is the plane's position in the camera's frame; $\boldsymbol{e}_{x,y,z}$ is the plane frame basis (defines the orientation); $\boldsymbol{d}$ is a given direction (see Fig. B.1). We can solve this equation with respect to λ. We should check whether $\lambda > 0$, otherwise the plane is actually in the direction defined by $-\boldsymbol{d}$. To do so we check:

$$\boldsymbol{d} \cdot \boldsymbol{e}_z {}^c\boldsymbol{t}_p \cdot \boldsymbol{e}_z > \varepsilon \tag{B.2}$$

where ε is a small positive value. If it is true, then λ is computed as follows:

$$\lambda = \frac{\boldsymbol{d} \cdot \boldsymbol{e}_z}{{}^{\mathrm{c}}\boldsymbol{t}_{\mathrm{p}} \cdot \boldsymbol{e}_z} \tag{B.3}$$

Then we can compute the texture coordinates of the intersection:

$$\boldsymbol{p} = \begin{pmatrix} f_u(\lambda \boldsymbol{d} - {}^{\mathrm{c}}\boldsymbol{t}_{\mathrm{p}}) \cdot \boldsymbol{e}_x + u_0 \\ f_v(\lambda \boldsymbol{d} - {}^{\mathrm{c}}\boldsymbol{t}_{\mathrm{p}}) \cdot \boldsymbol{e}_y + v_0 \end{pmatrix} \tag{B.4}$$

f_u, f_v, u_0, and v_0 define the transformation between texture image and the plane coordinates.

2 Texture Processing

Once the intersection coordinates are found, the brightness value must be computed. If we just take the nearest pixel's value on the texture, a very strong aliasing will appear. You can compare the original texture and the rendering result in Fig. B.2.

Our AA-filter is based on what is described in Chen *et al.* (2004). We can view the anti-aliasing problem in the following way: a continuous signal (texture) is projected using a diffeomorphism (projection mapping) into the image space; how to filter the texture before sampling in the image space to reduce the aliasing effect? A low-pass filter (for example, a Gaussian filter) should be used. And its kernel size is defined by the image sampling frequency. According to the sampling theorem, the texture should not contain any frequencies higher that $0.5\,\mathrm{px}^{-1}$. A perfect low-pass filter requires an infinite-size kernel, while Gaussian filter, even though is does not have a clear cut-off frequency, approximates a low-pass filter well enough. We used the fact that the Gaussian's frequency response is a Gaussian and the following identity is true:

$$\sigma_x \sigma_f = \frac{1}{2\pi} \tag{B.5}$$

Where σ_f and σ_x are respectively frequency and spatial standard deviations. We choose $\sigma_f = 0.3$, which is, in case of Gaussians, a good trade-off between cutting high frequencies, and preserving the low ones. From the identity we get $\sigma_x \approx 0.5$. We choose the kernel radius as $3\sigma_x = 1.5$. Then, we need to choose the minimal number of samples to take along x and y. Again we have to use the sampling theorem, this time for the texture. Let us assume that the texture does not contain frequencies higher than $0.5\,\mathrm{px}^{-1}$ in its own space. It means that the texture image adequately represents the underlying continuous texture signal. Also we assume that the texture is isotropic, that is we need to use the same sampling frequency in every direction. So, we sample it with a step of one pixel.

The basis vectors $(1, 0)^T$ and $(0, 1)^T$ are mapped to the texture space. to do so we just map the top and the left neighbor pixels to the texture space by the same procedure, as the one applied to the pixel under consideration. Of course, the process can be sped up by precomputing the intersection coordinates for all the pixels to avoid redundant computations.

Kernel is an outer product of two 1D kernels. According to length of the mapped basis vectors we choose an appropriate 1D kernel in either direction. Then we generate a grid in the texture space and sample the texture using the bilinear interpolation:

$$I(x, y) = \sum_{i=1}^{N} \sum_{i=1}^{M} w_N[i] w_M[j] J(\boldsymbol{p}_{i,j}) \tag{B.6}$$

here N and M are the required kernel sample numbers in x and y directions respectively; w_N, w_M are corresponding precomputed kernels. J is the texture, interpolated bilinearly; $\boldsymbol{p}_{i,j}$ represents the grid in the texture space. The latter is just an approximation because even

if the hypothesis about its frequency content is true, bilinear interpolation is not the way to reconstruct the intermediate values. Nevertheless this approximation works and provides a significant speedup.

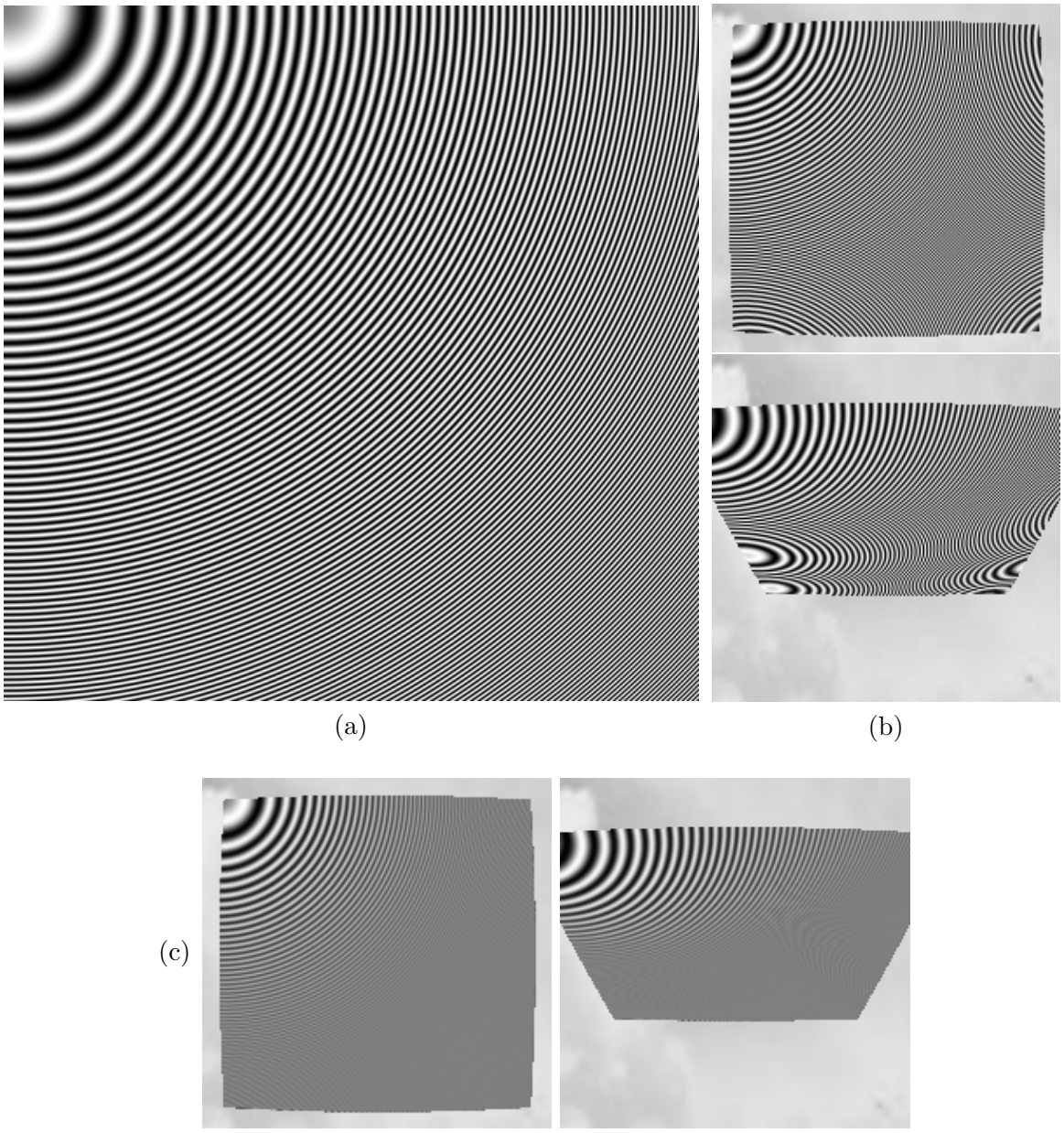


Figure B.2: (a) test texture, (b) images rendered without AA-filter applied, (c) anisotropic AA-filter is applied. A large scale of the image is necessary to avoid aliasing during the document viewing and printing. As you can see the AA-filter does not suppress the aliasing completely, yet increases the rendering quality a lot.

Appendix C

Résumé étendu en français

1 Introduction

Le but de ce travail est d'étudier des couches différentes de la perception visuelle basée cameras fisheye, ainsi que d'établir une base solide pour les algorithmes de perception de haut niveau. Dans cette introduction, nous évoquons des aspects différents de la vision par ordinateur, ses avantages et inconvenances. Puis nous décrivons les choix principales faites au cours de ce travail ainsi que sa structure.

1.1 Pourquoi la vision est-elle si attirante ?

Les cameras ont de nombreux avantages par rapport aux autres modalités de perception. Les principales sont les suivants:

1. Cameras sont moins chères que la plupart d'autres capteurs extéroceptifs, elle sont petite et légères, elles n'ont pas de pièces mobiles, donc elle sont mécaniquement plus résistantes.

2. Il existe une vaste variété de cameras avec un large choix de fréquences, de résolutions, de niveaux de bruit, de champs de vision et de consommation d'énergie.

3. La vision fournit des informations denses et riches sur l'environnement. C'est une modalité de perception extéroceptive, qui peut avoir plusieurs application différentes, comme la localisation, la navigation, la manipulation avec des objets.

4. Sachant que le but ultime est d'intégrer les robot dans la société humaine, ces derniers doivent pouvoir comprendre notre système de signaux et de communication. Or l'infrastructure qui est mise en place utilise une signalisation visuelle pour donner des information sur les direction et la sécurité de navigation.

1.2 Pourquoi la vision est-elle difficile

La formation d'images est un processus complexe, qui inclut de nombreux facteurs. Nous allons voir quelles difficultés proviennent de là. Le premier défi est que les cameras sont des capteurs passif, c'est à dire elles n'ont pas d'émetteurs et elle captent seulement la lumière déjà présente dans l'environnement. Alors les algorithmes de traitement doivent être robustes par rapport au changement d'éclairage, qui en général n'est pas contrôlé. Certains problèmes, comme la gamme dynamique, doivent être résolues au niveau du matériel.

Une autre difficulté est que la vision a besoin d'un environnement texturé. En fait, même si toutes les surfaces étaient peintes en blanc, il y aurait des nuances et des gradients. Mais habituellement, les algorithmes existant requierent des textures de haut contraste,

des lignes, des coins, des cercles, des ellipses etc. Si ce genre de caractéristiques manque à l'environnement, les images de ce dernier sont considérablement plus dur à traiter. Les techniques denses et semi-denses, qui traitent les images pixel par pixel, semblent une bonne solution à ce problème. Mais actuellement, il n'y a pas d'algorithme bien connu qui serait adaptatif et qui traiterait aussi bien les parties d'images de haut contraste que les parties plus lisses et floutées. Cela évoque le problème de la système de perception universelle qui serait capable d'intégrer toutes sortes d'information d'une façon optimale.

Les lidars mesurent directement la distance jusqu'un obstacle dans une direction donnée, tandis que les cameras génèrent des images, c'est à dire des tableaux numériques qui représentent l'intensité de lumière dans des direction différentes. L'information géométrique dans ce cas est plus facile à utiliser et à interpréter. D'autre part l'information photométrique est presque inutile en soi. Les images doivent être traitées d'une façon intensive afin de trouver des correspondances et reconstruire l'environnement 3D. Le traitement de haut niveau est pour l'instant le goulot d'étranglement des systèmes basées vision.

Encore une autre difficulté est causée par les occlusions et la manque de visibilité. Le problème de correspondance en presence des occlusions est NP-dur, donc la solution exacte est introuvable. Des algorithmes approximatives et pourtant efficaces pour dans le cas des application réelles ont été proposée.

1.3 Choix principaux

Au cours de ce travail, les idées expliquées ci-dessous nous ont servi de lignes directrices

Cameras grand angle Images omnidirectionnelles et grand angle nous fournissent de meilleurs propriétés de localisation que celles obtenues avec des cameras classiques. Grâce à un champ de vision plus large, l'environnement est observé plus longtemps pendant les virages, ce qui nous permet de le reconstruire et nous localiser plus précisément. Les images fisheye sont aussi meilleurs en tant que repères clés, car elles incluent plus d'information et imposent des contraintes angulaires moins rigoureuses que celles des images petit angle. Deux images fisheye nous suffisent pour couvrir 360° de vue.

Mais tous ces avantages vont avec la géométrie complexe des optiques fisheye. Pour que l'on puisse utiliser les images fisheye avec la même facilité que les images ordinaire, on a besoin d'un nouveau modèle, analytiquement simple et pourtant assez expressif pour approcher une large gamme d'optiques fisheye.

Vision géométrique Dans le cas des images prises avec une cameras, la modélisation géométrique peut faciliter leur traitement, améliorer la précision de reconstruction Les contraintes épipolaires réduisent par exemple l'espace de recherche de correspondances stéréo, ainsi que permettent de vérifier la consistance géométrique des données. Finalement, la reconstruction de l'environnement requiert un modèle géométrique ainsi qu'un algorithme de correspondance fiable.

Traitement d'image directe Les application de vision par ordinateur classiques avaient de longues chaînes de traitement d'images. Nous défendons le point de vue que tout traitement d'image modifie l'information originale et donc il vaut mieux être aussi proche de la source que possible. C'est à dire de traiter des images sous la forme sous laquelle elles sont acquises et nous fournies par les capteurs.

Par exemple, dans le cas de reconstruction stéréo, la rectification des images est une façon courante d'éviter le problème de modélisation géométrique et de transformer les courbe

épipolaires en droites. Pourtant, cette procédure casse l'uniformité de bruit et d'information dans l'image a cause d'une dilatation de certaines parties et d'une contraction des autres.

Donc notre bute est de développer une approche à la modélisation géométrique, telle qui nous permet de traiter avec facilitée des images d'une géométrie complexe.

Fusion des capteurs L'utilisation de modalités différentes et de mesures décorrélées réduit l'impact du bruit et augmente la robustesse générale du système. Les deux modalités les plus utilisées avec des cameras sont la centrale inertielle et l'odométrie des roues, s'il s'agit d'un robot mobile. L'odométrie a une plus mauvaise précision angulaire et une meilleure précision de mesure de distance que la centrale inertielle. La mesure de distance a un rôle particulier pour la perception visuelle car dans le cas monoculaire, la vision ne donne pas de mesures de longueur et donc il faut avoir une autre source de référence métrique.

1.4 Structure du document

Ce résumé étendu est censé d'éclairer les résultats principales de ce travail. Sa structure est la suivante.

Modélisation géométrique Le nouveau modèle est présente et sep propriétés géométriques sont analysées. Le modèle inverse analytique est décrite. Finalement la projection des droites est analysé et les équations implicites des courbes épipolaires sont montrées.

Étalonnage Principalement, les résultats numériques d'étalonnage de ce modèle sont présentés. Le modèle est compare avec d'autres modèles de l'état de l'art pour des optiques différentes. Aussi, la problématique d'étalonnage extrinsèque des robots mobiles équipés d'une camera est traitée.

Correspondance stéréo directe Les équations des courbes épipolaires sont employées pour calculer la correspondance stéréo entre deux images fisheye sans les rectifier.Des testes quantitatives sur des données synthétiques ainsi que réelles sont données.

Localisation visuelle Des méthodes de recalage d'images directes sont utiliser pour compléter le système de localisation visuelle. En faisant le recalage et la reconstruction alternativement, on obtient un système de odométrie visuelle. Un système de localisation et le résultats de ses testes avec des données réelles sont présontés.

2 Modelisation geometrique de cameras grand angle

La première étape pour monter un système de perception visuelle serait de modéliser les camera. Comme nous avons choisi les cameras fisheye comme matériel, il nous faut un modèle qui capte bien les distorsion géométriques, introduites par l'optique, et qui en même temps soit analytiquement simple. Nous avons proposé un modèle qui est base sur le modèle unifies (aussi dit modèle sphérique). Le nouveau modèle est défini par les équations suivantes:

$$\boldsymbol{m} = \begin{pmatrix} \dfrac{x}{\alpha\rho + (1-\alpha)z} \\ \dfrac{y}{\alpha\rho + (1-\alpha)z} \\ 1 \end{pmatrix} \quad \rho = \sqrt{\beta(x^2+y^2)+z^2} \qquad \text{(C.1)}$$

$$\boldsymbol{p} = K\boldsymbol{m}$$

Les deux paramètres de distorsion sont $\alpha \in [0,1]$ et $\beta > 0$. Il est requis que $\alpha\rho+(1-\alpha)z > 0$. K est la matrice de projection qui contient les paramètres intrinsèques f_u, f_v, u_0, v_0.

2.1 Surface de projection

Pour analyser ce modèle, on introduit la notion de surface de projection. Cette surface est définie par une équation avec les coordonnées 3D.

Cette notion peut être appliquée à une grande variété de modèles de projection avec de types de distorsion différents. Soit $\eta : \mathbb{R}^3 \to \mathbb{R}_+$ une fonction homogène de degré 1:

$$\forall \lambda \in \mathbb{R}_+ \quad \eta(\lambda \boldsymbol{X}) = \lambda \eta(\boldsymbol{X}) \tag{C.2}$$

Le modèle de projection est défini de la façon suivante:

$$\boldsymbol{m} = \begin{pmatrix} \dfrac{x}{\eta(\boldsymbol{X})} \\ \dfrac{y}{\eta(\boldsymbol{X})} \\ 1 \end{pmatrix} \tag{C.3}$$

Alors, la surface de projection est défini comme:

$$\eta(\boldsymbol{X}) = 1 \tag{C.4}$$

N'importe quelle fonction η defini un modèle de projection avec de propriétés différentes.

Pour le modèle proposé, $\eta(\boldsymbol{X}) = 1$ mène à:

$$\alpha\sqrt{\beta(x^2+y^2)+z^2} + (1-\alpha)z = 1 \tag{C.5}$$

En remplaçant $1-\alpha$ par γ et x^2+y^2 par r^2, on peut arriver à la forme suivante:

$$\alpha^2\beta r^2 + (\alpha-\gamma)z^2 + 2\gamma z = 1 \tag{C.6}$$

Cette équation sera utile pour calculer le modèle inverse et des équations de droites projetées.

2.2 Modèle inverse

Soit $\mathfrak{f} : \mathbb{R}^3 \backslash 0 \to \mathbb{R}^2$ le modèle de projection défini par (C.1). Soit $\mathfrak{f}^{-1} : \mathbb{R}^2 \to \mathbb{R}^3$ l'inverse de droite de $\mathfrak{f}$:

$$\mathfrak{f}(\mathfrak{f}^{-1}(\boldsymbol{m})) = \boldsymbol{m} \tag{C.7}$$

ou, autrement dit, $\mathfrak{f} \circ \mathfrak{f}^{-1} = I$. On cherche un difféomorphisme entre les points de l'image et les directions dans le champ visuel de la camera. Or, les points sur la surface de projection se projettent sur l'image orthogonalement. Alors, $\mathfrak{f}^{-1}$ peut être défini comme:

$$\mathfrak{f}^{-1} : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ z(x,y) \end{pmatrix} \tag{C.8}$$

On obtient $z(x,y)$ en résolvant (C.6) et en choisissant la bonne solution parmi deux. Le résulta est (avec $r = \sqrt{x^2+y^2}$):

$$z = \frac{1-\alpha^2\beta r^2}{\alpha\sqrt{1-(\alpha-\gamma)\beta r^2} + \gamma} \tag{C.9}$$

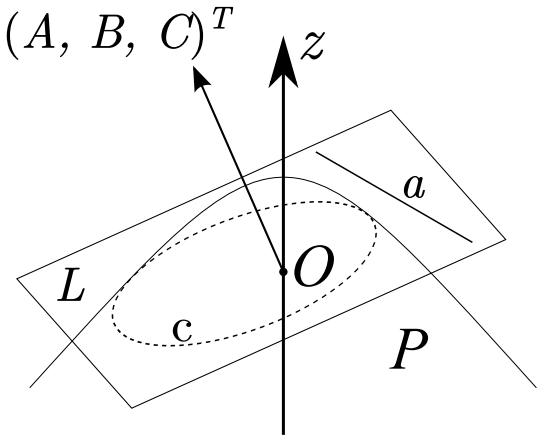


Figure C.1: Droite a avec l'origine de projection O définissent plan L. L'intersection entre L et P définissent courbe c. En projetant c orthogonalement sur le plan normal on obtient la projection de a.

2.3 Straight Line Projection

Avec la notion de surface de projection, on peut montrer que les droites se projettent comme des sections coniques. Étant donné droite a, soit L le plan défini par a et O (Fig. C.1). L'équation de L est de la forme suivante:

$$Ax + By + Cz = 0 \tag{C.10}$$

Pour trouver l'image de la droite, d'abord on doit projeter celle-dernière sur la surface de projection. Pour ça, on cherche l'intersection c entre L et P. En prenant (C.6) et (C.10) on obtient le système d'équations suivant:

$$\begin{cases} Ax + By + Cz = 0 \\ \alpha^2\beta(x^2 + y^2) + (\alpha - \gamma)z^2 + 2\gamma z = 1 \end{cases} \tag{C.11}$$

L'étape suivant est de projeter c orthogonalement sur le plan normal. C'est à dire, on doit exclure la coordonnée z du système d'équations. Si $C = 0$, alors l'image de la droite est une droite qui passe par le centre d'image:

$$Ax + By = 0 \tag{C.12}$$

Si $C \neq 0$, alors on peut exprimer z de la première équation:

$$z = -\frac{Ax + By}{C} \tag{C.13}$$

et le substituer dans la deuxième:

$$\alpha^2\beta(x^2 + y^2) + (\alpha - \gamma)\left(\frac{Ax + By}{C}\right)^2 - 2\gamma\frac{Ax + By}{C} = 1 \tag{C.14}$$

Donc, on a un polynôme en x et y de degré deux, qui défini une section conique.

Epipolar Curve Equation On peut calculer analytiquement les expressions pour les coefficients des équations de courbes épipolaires dans le cas d'un système stéréo calibré. Considérons deux cameras avec les modèles de projection étalonnés $\mathfrak{f}_1$ et $\mathfrak{f}_2$. La transformation entre ces deux cameras est connue et est représentée par une matrice de rotation ${}^1R_2$ et un

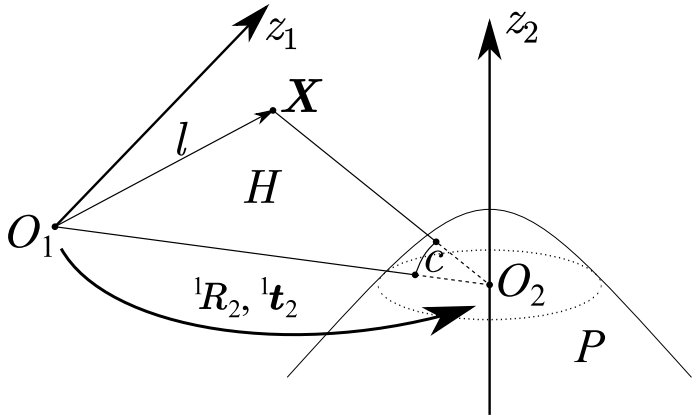


Figure C.2: Un système stéréo étalonné. ${}^1R_2$ et ${}^1\boldsymbol{t}_2$ définissent la transformation entre les repères des cameras (une matrice de rotation et un vecteur de translation); $\boldsymbol{X}$ est un point reconstruit; la droite l est définie par X et le center de projection O_1 de la première camera. Le plan H passe par O_2 et l. La courbe c est l'intersection entre H et la surface de projection P. Pour obtenir la courbe épipolaire, on doit exclure la coordonnée z de l'équation de c.

vecteur de translation ${}^1\boldsymbol{t}_2$ (voir Fig. C.2). Le plan épipolaire H dans le repère O_2 est défini par l'équation suivante:

$$ {}^1\boldsymbol{X}_1^T [{}^1\boldsymbol{t}_2]_\times {}^1R_2 \boldsymbol{X} = 0 \tag{C.15}$$

L'étape finale est de remplacer x et y par leurs expressions en fonction de u et v:

$$ x = \frac{u - u_0}{f_u} \quad y = \frac{v - v_0}{f_v} \tag{C.16}$$

pour produire un polynôme de la forme suivante:

$$ k_{uu}u^2 + k_{uv}uv + k_{vv}v^2 + k_u u + k_v v + k_1 = 0 \tag{C.17}$$

2.4 Conclusions

Les modèles de projection fisheye existants montrent des combinaisons différentes d'avantages et d'inconvenances. Mais aucun d'entre eux n'est pas complètement universel car certains ne sont pas assez précis ou ne vont que pour une famille d'optique limitée, pendant que d'autres ne sont pas analytiquement inversibles et coûteux du point de vue computationnel. Dans ce travail, nous proposons un nouveau modèle de projection ainsi qu'obtenons des résultats important sur les propriétés géométriques de ce modèle. Le contributions principales sont décrites ci-dessous.

Modèle unifié amélioré En augmentant le Modèle Unifié, on obtien un modèle, qui est analytiquement élégant et simple, et pourtant approche plus précisément les vraies optiques fisheye, grâce à un degré de liberté supplémentaire. Les résultat quantitatives, décrits dans la section 3, montrent que le modèle proposé rend une fonction de distorsion additionnelle inutile même pour les optiques fisheye avec une distorsion importante.

Surface de projection Cette notion est un outil efficace d'analyse des propriétés géométriques des modèles de projection. En utilisant ce concept, nous avons trouvé les expressions analytiques du modèle inverse, ainsi que montré que ce modèle de projection est capable d'approcher toute surface de projection de degré deux.

Image des droites Nous avons également montré que les droites se projettent comme des sections coniques. Une méthode de calcule des équations de ces courbes a été trouvé. Pour un système stéréo étalonné, il est possible de trouver des expressions analytiques de courbes épipolaires. Le dernier fait est utiliser dans ce travail pour le calcul stéréo.

3 Étalonnage des systèmes de perception visuelle

Une toolbox d'étalonnage de cameras fisheye a été développée. Elle contient un détecteur de mire d'étalonnage, qui est plus efficace et plus rapide que celui fourni avec OpenCV. Ce détecteur nous permet d'utiliser des centaines d'images pour le processus d'étalonnage.

3.1 Étalonnage du modèle unifié amelioré

Les résultats d'étalonnage sont présentés sur les Fig. C.3–C.4. Six optiques ont été étalonnées, y compris quatre optiques fisheye, une optique grand angle et une optique trou d'épingle. Trois modèles ont été testés : modèle unifié (UCM, 5 parametres), modèle unifié amélioré (EUCM, 6 parametres), modèle unifié avec des distorsion (UCM-D, 10 parametres).

Fig. C.3 montre que la précision de l'EUCM est plus proche de celle de UCM-D plut que de UCM. Donc en rajoutant un seul paramètre de plus, on gagne en précision d'une façon significative. Par ailleurs, le temps d'étalonnage de EUCM est plus petit que celui de UCMD, et tend vers celui de UCM, ce qui nous dit que EUCM est presque aussi simple du point de vu du calcul que UCM.

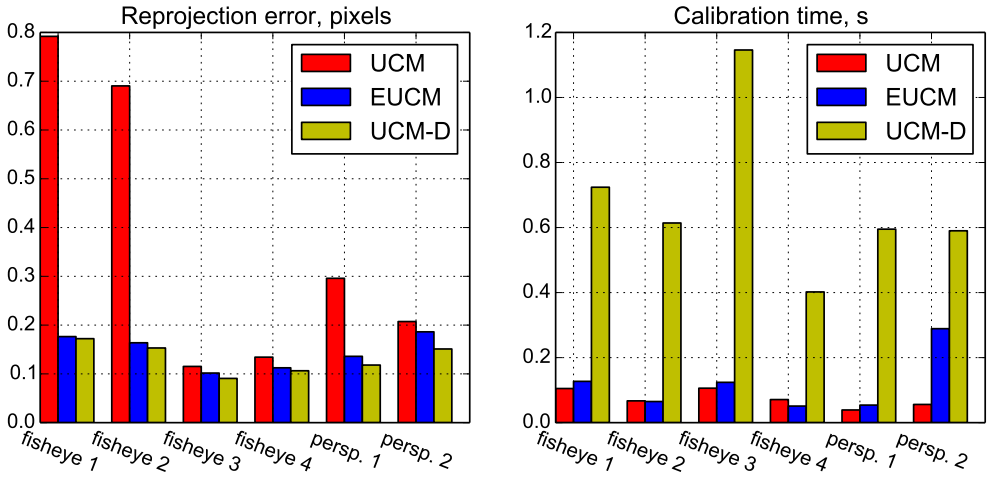


Figure C.3: Résultats d'étalonnage monoculaire.

3.2 Étalonnage extrinsèque de l'odométrie

Nous avons proposé une méthode de calcul des trajectoires optimales pour l'étalonnage des systèmes de perceptions visuelle des robots mobiles. L'idée est de trouver les trajectoires qui maximisent le déterminant de la matrice Hessienne du problème d'étalonnage.

Ce concept a été testé avec des données synthétiques. Cela nous permet de comparer les résultats d'étalonnage avec les vrais valeurs. Pour que les paramètres extrinsèques soient identifiables, la courbure de la trajectoire ne doit pas être constante. Nous avons choisi comme trajectoire deux arcs avec des courbures différentes. Une fois les trajectoires optimales générées, un ensemble d'images correspondantes est synthétisé. Ces images ont été utilisées pour étalonner la cameras ainsi que la transformation entre la base du robot mobile et la

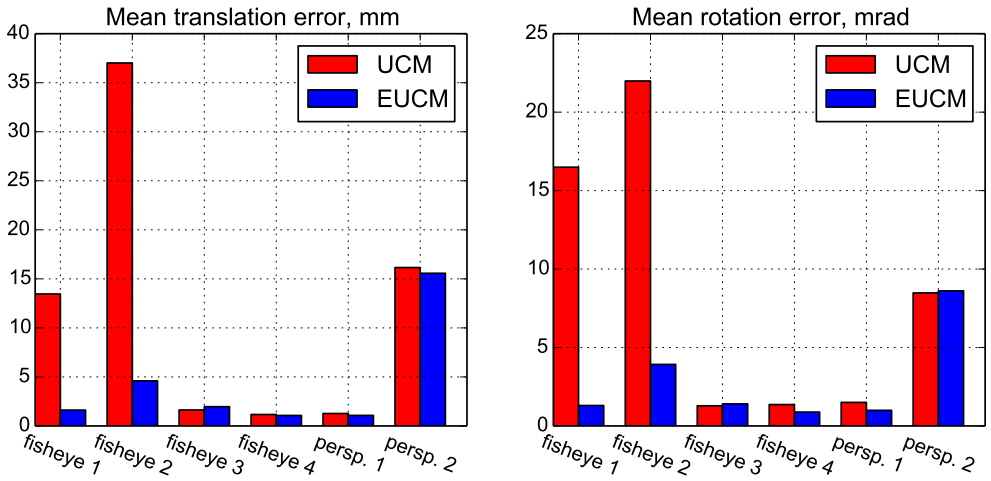


Figure C.4: Résultats d'étalonnage monoculaire — une comparaison de reconstruction de pose.Les poses calculées avec le modèle UCM-D sont prises comme une référence (10 paramètres).

camera. Bien que les images soient synthétiques, le traitement d'images introduit une certaine erreur dans le processus, car la détéction de la mire n'est pas parfaite (quoique relativement précise)

Les résultats d'étalonnage ont été comparés avec ceux d'un étalonnage fait avec des trajectoires suboptimales. Ces trajectoires suboptimales ont été obtenues en arrêtant le processus d'optimisation des trajectoires avant que la convergence soit atteint. Les deux ensembles de trajectoires son donnés sur Fig. C.5.

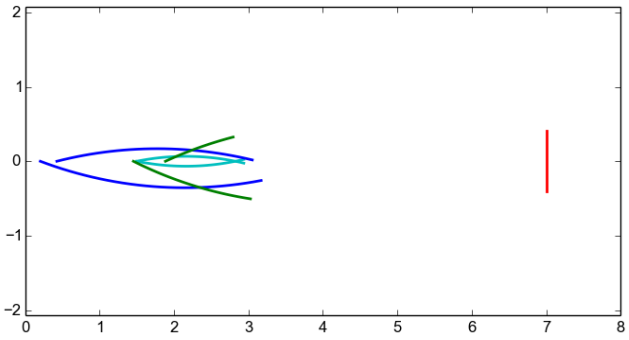


Figure C.5: Trajectoires générées ainsi que leur coût pendant l'étalonnage E: Cyan – trajectoires initiales, $E = 688$, la contrainte sur la courbure n'est pas satisfaite; vert – trajectoires suboptimales, $E = -29.6$; bleu – trajectoires optimales, $E = -37.4$.

Pour évaluer la robustesse de l'étalonnage, un bruit a été ajouté aux mesures d'odométrie. Le résultat d'étalonnage extrinsèque en présence de bruit sont visualisés sur Fig. C.6.

Éssais avec des données réelles Cette méthode d'étalonnage a été utilisé avec succès pour étalonner un véhicule équipé d'une camera. L'étalonnage a été fait deux fois pour deux jeux de données, similaires mais distinct. Le résultat dés deux étalonnages sont très proches, ce qui nous montre la répétabilité de cette expérimentation. Les résultats de cet étalonnage ont ensuit été utiliser pour faire de la localisation basée vision et odométrie des roues.

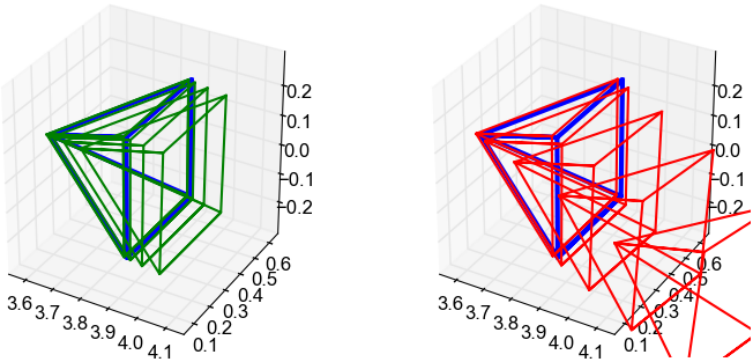


Figure C.6: Visualisation de l'étalonnage sous influence du bruit. Bleu — la vraie position, vert — trajectoires optimales, rouge — trajectoires suboptimales. Le modèle de camera est une pyramise de taille $0.4 \times 0.4 \times 0.4$ m. Plus fort est le bruit, plus importante est l'erreur d'estimation. Pourtant la trajectoire optimale (à gauche) paraît plus résistante au bruit.

3.3 Conclusions

Une toolbox d'étalonnage flexible et efficace pour les systèmes multi-camera a été développé. Cette toolbox contient un nouveau détecteur de la mire d'étalonnage de type échiquier. Le détecteur est plus efficace et plus rapide que celui, fourni avec OpenCV. Il nous permet d'utiliser de grand jeux de données pour faire l'étalonnage en gardant le temps d'étalonnage relativement petit.

Le modèle unifié amélioré a été testé intensivement. La précision de ce modèle, évaluée avec l'erreur de reprojection après l'étalonnage, est proche à celle du Modèle Unifié avec la distorsion. Pourtant le temps d'étalonnage du modèle amélioré correspond plutôt à celui du modèle sphérique. En ayant deux paramètres de distorsion seulement, nous évitons tout risque de surapprentissage (*overfittig* en anglais). Ce fait rend le modèle aussi utilisable pour les cameras faibles distorsions.

Nous avons aussi proposé et testé une méthode de calcul des trajectoires optimales pour l'étalonnage extrinsèque de robots mobiles. La robustesse de cette méthode à l'égard du bruit de bruit de mesure a été démontrée avec des donnée synthétiques. Un véhicule équipé d'une camera a été étalonné avec cette méthode et les résultats de cet étalonnage ont été utilisé pour faire de la localisation basée vision et odométrie des roues.

4 Direct Stereo Correspondence for Fisheye Cameras

La correspondance stéréo est la base du système de perception développé dans le cadre de ce projet. L'idée de l'algorithme de correspondance est d'éviter la rectification des images fisheye et les traiter directement. Le modèle de caméra propose nous permet de calculer les équations des courbes épipolaires et d'utiliser la notion de disparité, c'est à dire, la distance parcouru le long d'une courbe épipolaire mesurée en pixels. Tout cela nous permet d'utiliser l'algorithme de correspondance stéréo qui s'appelle *Semi-Global Matching*.

Une autre option est d'utiliser une image de profondeur calculée pour une base stéréo plus petite comme valeurs initiales pour le calcul avec une base plus large. Comme ça, on réduit significativement l'intervalle de recherche de correspondance ainsi que la probabilité de trouver une correspondance erronée (à cause d'une ambiguïté). Cet algorithme est particulièrement utile dans le cas d'une séquence vidéo. Pour initialiser l'image de profondeur on utilise SGM avec une base stéréo petite.

4.1 Results

L'algorithm a été testé avec des données synthétiques, ce qui nous a permis d'évaluer sa précision. Nous avons testé les deux algorithmes sur les même séquences, qui représentent la même scène avec la transformation stéréo différente. La première image de chaque séquence est la même (Fig. C.7).

Les deux montrent une performance similaire. Fig. C.8 nous montre la dernière image de chaque séquence ainsi que deux images de profondeur pour chaque des deux méthodes: avec la base stéréo $\approx$ 6 cm et $\approx$ 18 cm. Certaines parties de la carte de profondeur sont vides parce que pour les limites de disparité données, pour ce pixels la recherche le long des courbes épipolaires excéderais les limites de l'images. On peut voir que dans toit les cas, l'algorithme parvient à reconstruire la scène, bien que la transformation aie un impacte sur la reconstruction.

Pour SGM, l'effet de quantification est particulièrement remarquable à cause du fait que le spectre de profondeur est discret. Donc les discontinuité en profondeur sont inévitables. Au contraire, la reconstruction faite par l'algorithme prédictif parait plus lisse car cet algorithme accumule des informations provenantes des images différentes, ce qui lui permet d'avoir un spectre plus dense.

À mesure que la base stéréo s'élargit, le spectre de profondeur devient de plus en plus dense et la reconstruction devient plus lisse et précise.

Real Data Tests L'algorithme a été également testé avec des données réelles. Les images ont été prises avec une camera fisheye étalonnée attachée à un bras robotique, ce qui nous a permis de calculer la transformation stéréo pour toutes les paires d'images. Fig. C.10 montre quelques exemples d'images et les images de disparité calculé.

Pour avoir une évaluation quantitative de l'algorithme, la reconstruction 3D d'un objet planaire à été faite et comparée avec les valeurs de référence (see Fig. C.11). La réalité de terrain a été obtenu de la façon suivante:

1. La transformations entre la base du robot et le plan a été mesurée.

2. En sachant la configuration du robot, il es possible de calculer la transformation camera-plan.

3. L'image de profondeur a été générée avec les paramètres intrinsèques de la camera.

L'objet planaire ne couvrait pas l'images entière, donc plusieurs jeux de données ont été acquis pour tester la méthode dans les parties différentes de l'image. Les images de disparité, c'est à dire la sortie directe de l'algorithme de stéréo, ont ensuit été converties en images de profondeur. Puis, l'erreur a été calculé et analysé. Seules les pixels appartenant au plan ont été pris en compte. Tous les pixels avec une erreur de reconstruction supérieure à 100 mm sont considérés comme outliers et rejetés.

Les résultats montrent que plus la base stéréo est large, plus l'erreur est petite. La distribution d'erreur est relativement fine par rapport aux valeurs de distance actuelles. Aussi, le taux de inliers est élevé, ce qui veut dire que globalement cet algorithme est efficace et précis pour la reconstruction d'objets planaires.

4.2 Conclusions

Le propriété du Modèle Unifié Amélioré nous permettent de modéliser les systèmes fisheye stéréo. Les équations de courbes épipolaires calculé avec ce modèle, une fois rasterisées, peuvent être utiliser pour calculer la correspondance stéréo directe.

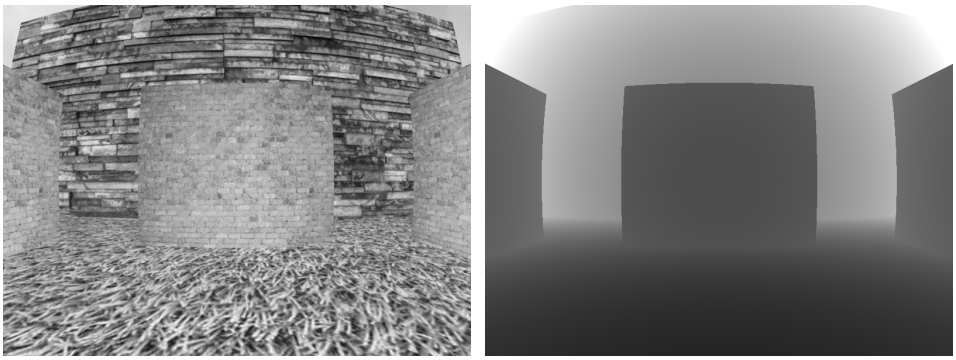


Figure C.7: L'image de base et la profondeur de référence.

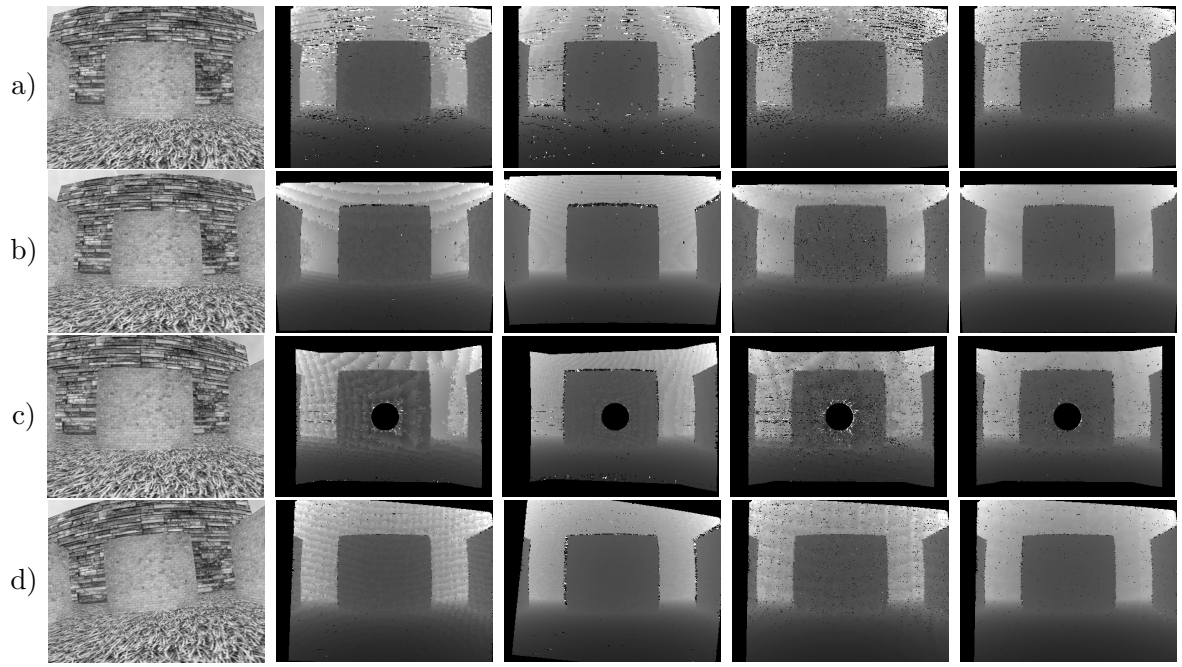


Figure C.8: Exemples d'images de profondeur calculés pour des mouvements de camera différents. La première colonne montre les images finales de chaque séquence. Deuxième et troisième colonnes montrent les résultats de reconstruction par SGM. Quatrième et cinquième colonnes montrent les résultats pour l'algorithme prédictif. Deuxième et quatrième colonnes correspondent à une base stéréo de ≈ 8 cm. Troisième et cinquième colonnes correspondent à une base stéréo de ≈ 20 cm. (a) mouvement à droite, pas de rotations. Le mur de l'arrière-plan donne beaucoup de bruit de reconstruction car certaines de ses caractéristiques sont parallèles aux courbes épipolaires. (b) mouvement vers le bas avec une rotations vers le haut. Le mur de l'arrière-plan est reconstruite d'une façon plus régulière. (c) mouvement en avant avec une rotation à droite, L'aire noire sur le bord de l'image apparaît parce que quand la camera va en avant, tout ce qui est sur le bord de l'image disparaît. Le cercle noir au milieu est autour de l'épipole, ou la reconstruction 3D rencontre une singularité. (d) mouvement à droites et en avant avec une rotation autour de l'axe optique. Cela nous montre l'un des avantages les plus intéressants de cette méthode, la rotation autour de l'axe optique n'abîme pas les résultats de reconstruction.

a)

b)

c)

d)

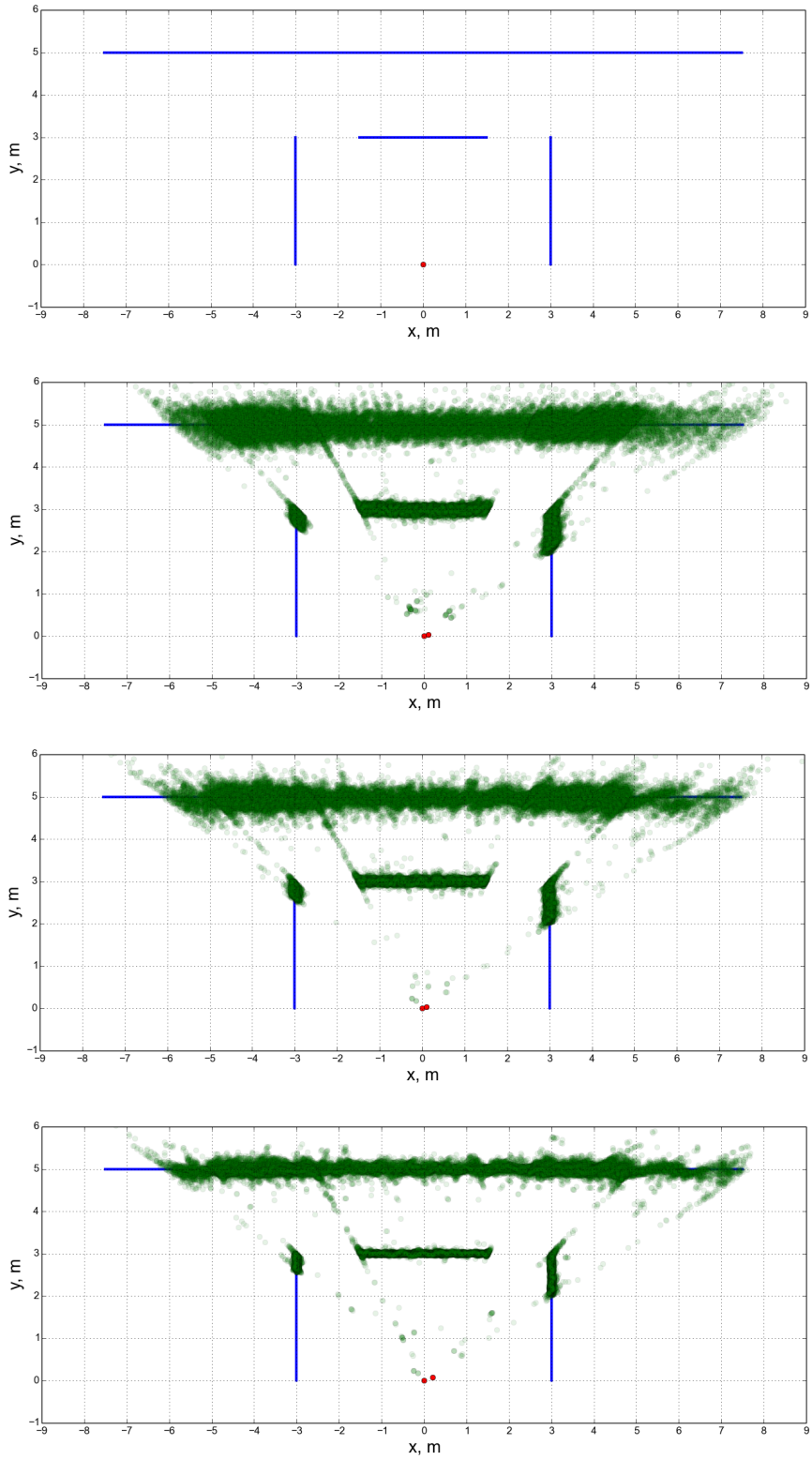


Figure C.9: Exmemples de reconstruction. Seul la partie du nuage de points au-dessus du sol est afficher. La sequence (d) de Fig. C.8 est utilisé. La camera se déplace à gauche et vers l'avant en tournant autour de l'axe optique. Les points rouges représentent les positions de la camera utilisées pour le calcul stéréo. (a) Le schéma de la scène, position de la camera initiale, vue de dessus. (b) Reconstruction avec SGM, la base stéréo est de 10 cm environ. (c) La même configuration que (b), mais avec la méthode prédictive : le résultat est légèrement meilleur parce que l'algorithme fusionne les information de toutes les itération précédentes. (d) Reconstruction basée prédiction pour la base de 20 cm.

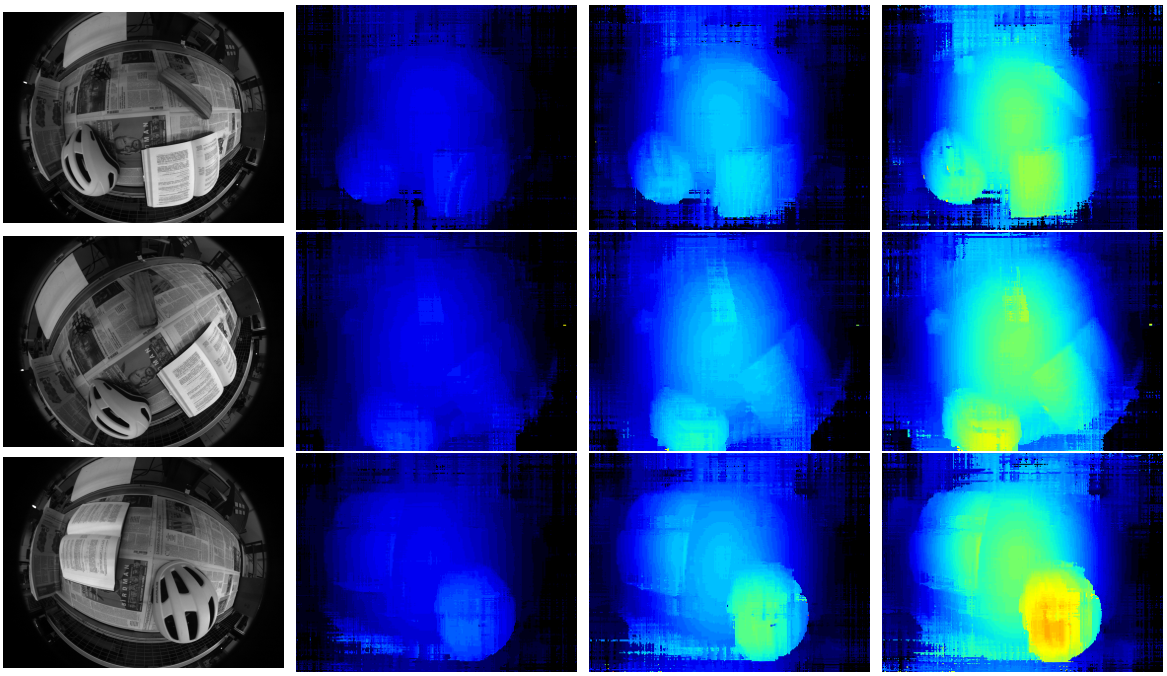


Figure C.10: Images originales (à gauche) et les images de disparité correspondantes (à droite) pur les bases stéréo 10, 20, 30 mm.

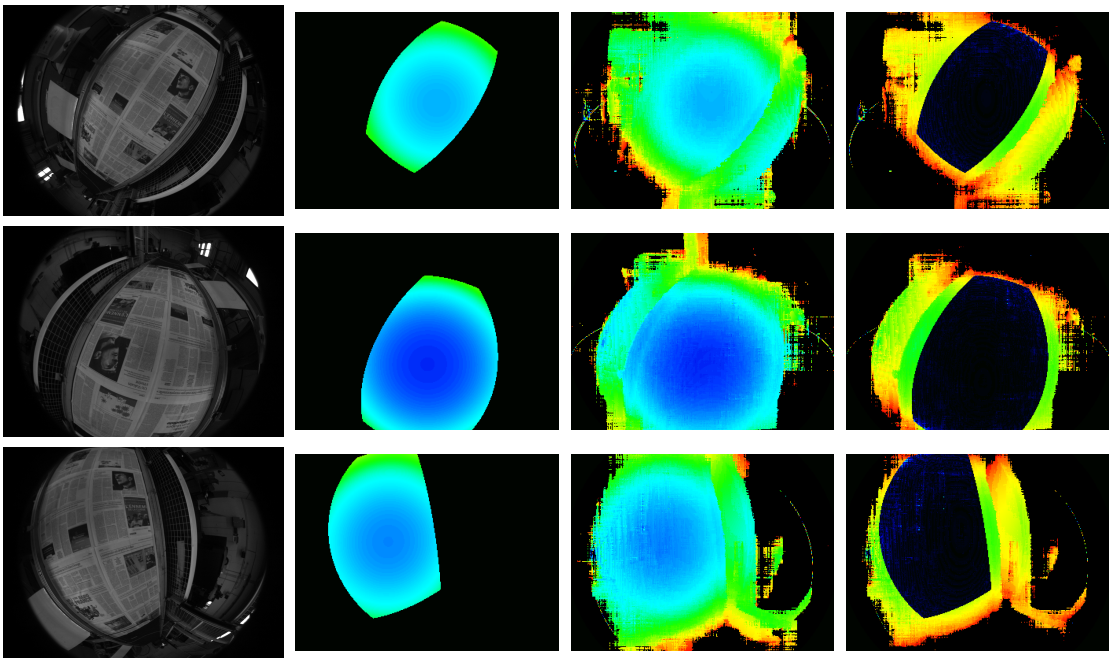


Figure C.11: Illustration de la methodologie d'évaluation. (a) l'image de base. (b) la vraie image de profondeur du plan du journal. (c) les résultats de reconstruction stéréo. (d) la difference absolue entre la réalité de terrain et la reconsrtuction calculée.

La correspondance stéréo basée disparité nous donne une reconstruction 3D précise, ainsi qu'une méthode d'estimation d'erreur de reconstruction.

Comme cela est montré dans ce travail, l'algorithme stéréo fisheye peut être la base d'un système de localisation basée vision. Nous suggérons que cet algorithme peut potentiellement atteindre le niveau de précision nécessaire pour la navigation autonome sécurisée et robuste.

5 Vision-Based Localization and Mapping

Un modèle de camera fisheye étalonné et un algorithme de reconstruction stéréo sont le ingrédient nécessaires pour mettre en place un système de localisation visuelle directe. La seule chose manquante est la méthode de calcul de transformation entre deux positions basée vision. S'il s'agit d'une méthode directe, on utilise le terme "recalage d'images". En faisant alternativement le calcul de profondeur et le recalage d'images on arrive à faire la localisation visuelle.

5.1 Odométrie basée points caractéristiques

Le point fort des points caractéristiques c'est que L'architecture de l'odométrie basée points caractéristiques est représentée sur Fig. C.12. Cet odométrie utilise une méthode de reconstruction 3D régularisée ce qui nous permet d'utiliser l'optimisation non-linéaire et les mesures d'odométrie des roues Une séquence synthétisée a été utilisée pour tester cet algorithme. Un bruit à été ajouté aux mesures d'odométrie des roues.

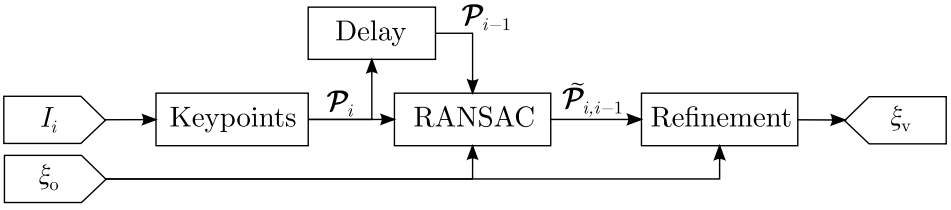


Figure C.12: L'architecture de l'odométrie visuelle. I_i et $\xi_{o,i}$ sont les entrées: l'image courante et la mesure d'odométrie des roues; $\mathcal{P}_i$ est l'ensemble des caractéristiques détectées; $\tilde{\mathcal{P}}_{i,i-1}$ est l'ensemble d'inliers calculé par RANSAC.

Nous utilisons RANSAC à deux points avec l'odométrie des roues en tant que valeur a priori. La trajectoire reconstruite est présenter sur Fig. C.14 L'odométrie des roues dérive très vite a cause d'un biais d'orientation très fort. Et pourtant l'odométrie visuelle suit la vraie trajectoire beaucoup plus précisément.

5.2 Odométrie visuelle directe

L'odométrie visuelle directe est basée sur la minimisation de l'erreur photométrique. Cette odométrie est basée sur le principe de repères clés, illustré sur Fig. C.15.

(a) Le premier repère clé avec l'origine O_1 est créé. Il a certaines limites d'orientation et de distance (A et B). Il y a une image de profondeur $\mathcal{D}_1$ et une image I_1 associée avec ce repère.

(b) Tant que le robot reste dans les limites du repère, il se localise par rapport à O_1. Cette transformation ζ est puis utilisée pour raffiner $\mathcal{D}_1$.

(c) Dès que le robot sort des limites, un nouveau repère clé est créé, avec l'origine O_2 et l'image I_2. $\mathcal{D}_1$ est projeté dans O_2 et fusionné avec une nouvelle image de profondeur $\mathcal{D}(I_1, I_2, {}^2\xi_1)$.

Figure C.13: Une image de la séquence synthétique.

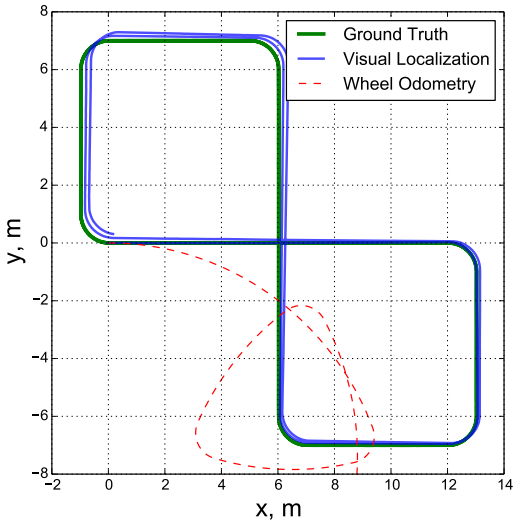


Figure C.14: L'odométrie visuelle testée sur des données synthétiques.

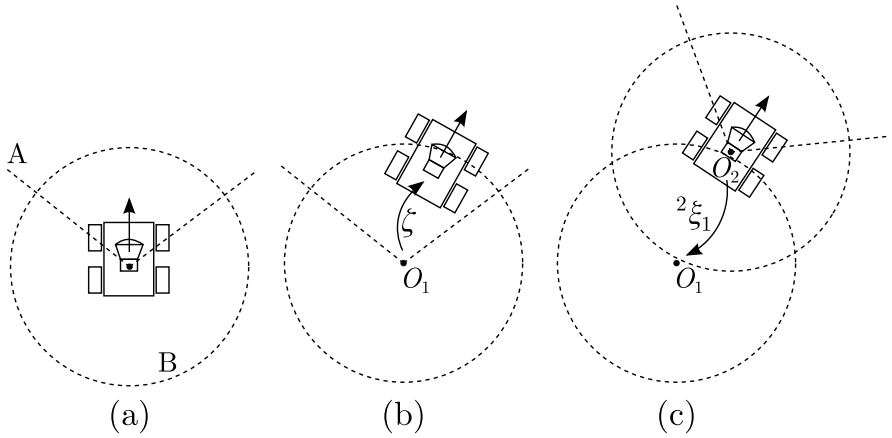


Figure C.15: L'odometrie visuelle directe.

Pour résoudre le problème d'échelle, on normalise la distance de l'incrément de mouvement selon la mesure de l'odométrie des roues. C'est à dire, l'odométrie des roues nous fournit la référence d'échelle.

Le resultat de reconstruction de trajectoire est montré sur Fig. C.16. On peut voir l'impacte de la normalisation d'échelle.

5.3 Localization avec des données réelles

Pour tester la précision et la robustesse de la technique de localisation proposée, nous avons fait le suivant:

1. Une carte sensorielle visuelle a été construite en utilisant la réalité de terrain.

2. L'algorithme de localisation a été utilisé pour reconstruire la trajectoire absolue du véhicule.

3. La sortie a été comparée avec la réalité de terrain.

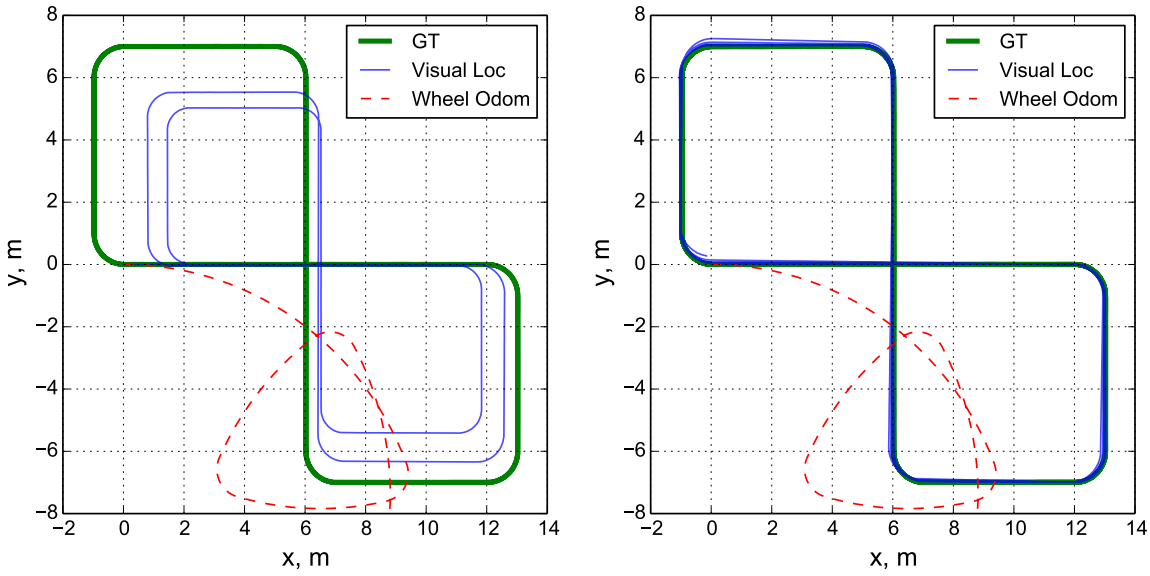


Figure C.16: Odometrie visuelle directe avec le recollage d'images photométrique. quatre tours en forme d'un 8. À gauche : pas de normalisation d'échelle, à droite : la normalisation d'échelle dure.

L'acquisition a été faite dans un quartier résidentiel le 20 novembre 2017 (la carte et le premier jeu de donnée) et le 30 novembre 2017 (le deuxième jeu de donnée). L'information mutuelle est utilisée pour se localiser avec une carte construite dix jours plus tôt. La trajectoire est présentée sur Fig. C.17.

Figure C.17: Les trajectoires utilisées dans l'expérience. Les longueurs approximatives: bleu — 650 m, vert — 520 m, rouge — 740 m.

Map Construction La carte est une séquence d'images avec une position de camera dans le repère global associée à chaque d'entre elles. La réalité de terrain a été obtenue en fusionnant les mesures de l'odométrie des roues, de la centrale inertielle, et de GPS RTK haute précision. Le processus de construction de la carte est illustré sur Fig. C.18.

Localization Le schéma du système de localisation est montré sur Fig. C.19 La position est initialisée avec la réalité de terrain, la première transformation est calculée avec l'odométrie basée points caractéristiques. Cela nous permet de calculer l'image de profondeur et ensuit utiliser la localisation visuelle directe.

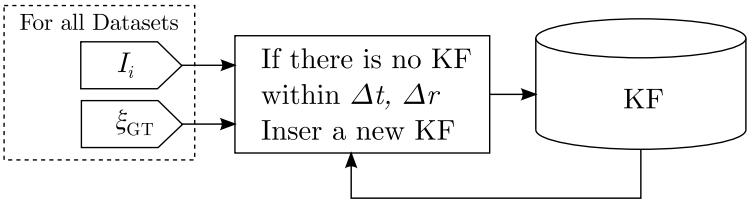


Figure C.18: Le processus de construction de la carte. I_i et $\xi_{\mathrm{GT},i}$ sont des mesures : l'image courante et la réalité de terrain pour de la position du véhicule. Δt et Δr sont les seuils de la translation et de la rotation pour la construction de carte. Dans notre cas ils sont 1.5 m et 1 rad respectivement.

Deux approches ont été testées. Dans la première, sans lissage de trajectoire, quand on change de repère de référence, on n'utilise la position actuelle que pour initialiser le processus d'optimisation. Dans la deuxième approche, on inclut la position courante comme critère d'optimisation.

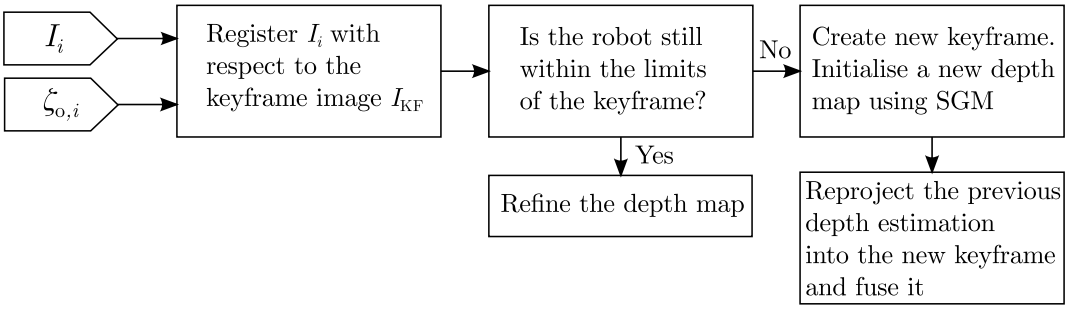


Figure C.19: Le schéma de localisation. I_i et $\zeta_{\mathrm{GT},i}$ sont l'image courante et l'incrément d'odométrie depuis l'image précédente.

Des exemples des graphes d'erreur et les trajectoires calculées sont sur Fig. C.21. Les erreurs moyennes sont résumées sur Fig. C.20. Le lissage de trajectoire augmente la précision presque dans tous les cas. Il enlève les épines d'erreur et rend la trajectoire moins bruitée. L'écart-type de l'erreur reste autour de 10-15 cm, ce qui est une précision suffisante pour la localisation et navigation des véhicules autonomes.

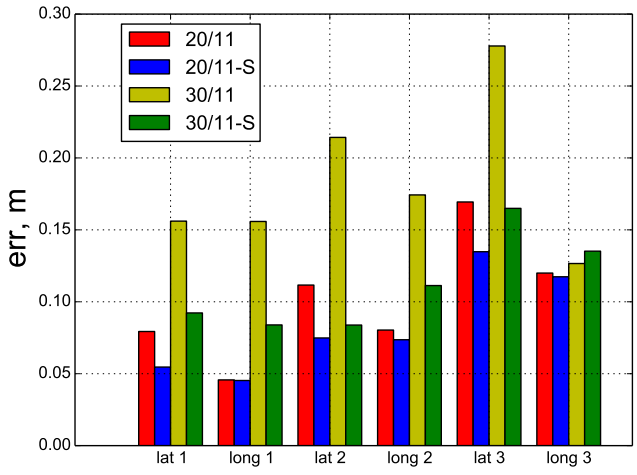


Figure C.20: Comparaison des erreurs pour des jeux de données différents. 20/11 et 30/11 correspondent aux dates d'acquisition. "-S" correspond à "lissage" (*smoothing* en anglais). "lat" — erreur latérale, "long" — erreur longitudinale.

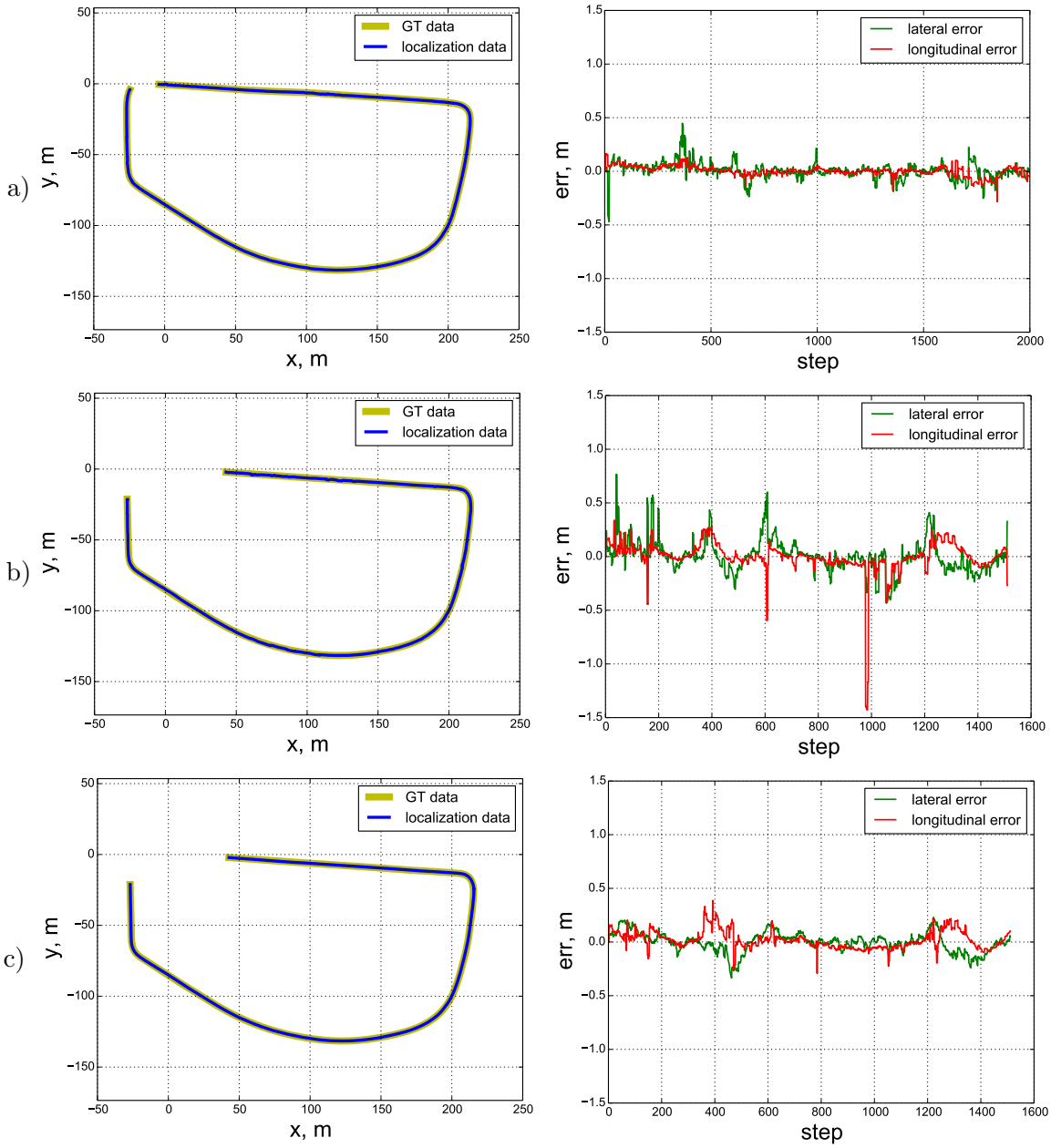


Figure C.21: Resultats de localisation — les trajectoires reconstruites et l'erreur correspondante dans le repère du robot. a) Les données de localisation datent du même jour que celle utilisées pour construire la carte

5.4 Conclusions

Des systèmes différents de localisation visuelle basée camera fisheye et odométrie des roues ont été implémentés et testés avec des données synthétisées. On met l'accent sur l'utilisation de recalage d'images directe photométrique ou basé information mutuelle.

Un système de localisation visuelle, qui utilise les concepts principales développés dans le cadre de ce travail, a été implémenté et testé sur des données réelles. Il s'est montré robuste par rapport aux changements environnementaux, qui étaient apparus au cours des 10 jours, qui séparaient la construction de la carte et la localisation. En plus, le système est robuste par rapport aux objets mobiles, tels que les voitures, les piétonnes.

Le recalage photométrique s'avère être plus efficace du point de vue du calcul, mais il ne peut être employé que dans le cas où les images à recaler sont acquises pendant une durée courte, dans les mêmes conditions d'éclairage. En revanche, l'information mutuelle nous permet de recaler des images acquise sur deux jours différents.

Bibliography

S. Abraham and W. Förstner. Fish-eye-stereo calibration and epipolar rectification. {*ISPRS*} *Journal of Photogrammetry and Remote Sensing*, 59(5):278 – 288, 2005.

S. Agarwal, K. Mierle, and Others. Ceres solver. `http://ceres-solver.org`, 2010.

G. Antonelli, F. Caccavale, F. Grossi, and A. Marino. Simultaneous calibration of odometry and camera for a differential drive mobile robot. In *IEEE International Conference on Robotics and Automation*, pages 5417–5422, May 2010.

Z. Arican and P. Frossard. Dense disparity estimation from omnidirectional images. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 399–404, Sept 2007.

B. Armstrong. On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics. *The International Journal of Robotics Research*, 8(6):28–48, 1989.

S. Baker and S. K. Nayar. Panoramic vision. chapter Single Viewpoint Catadioptric Cameras, pages 39–71. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

C. Beall and F. Dellaert. Appearance-based localization across seasons in a metric map. In *6th Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV)*, Chicago, USA, 2014.

P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. S. Torr. Playing doom with slam-augmented deep reinforcement learning. *CoRR*, abs/1612.00380, 2016.

P. Biber and T. Duckett. Dynamic maps for long-term operation of mobile service robots. In S. Thrun, G. S. Sukhatme, and S. Schaal, editors, *Robotics: Science and Systems*, pages 17–24. The MIT Press, 2005.

F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *J. Intell. Robotics Syst.*, 53(3):263–296, Nov. 2008.

G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 896–901. AAAI Press, 1996.

C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. Simultaneous localization and mapping: Present, future, and the robust-perception age. *CoRR*, abs/1606.05830, 2016.

M. Calonder, V. Lepetit, C. Strecha, and P. Fua. *BRIEF: Binary Robust Independent Elementary Features*, pages 778–792. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

G. Caron, E. Marchand, and E. Mouaddib. Photometric visual servoing for omnidirectional cameras. *Autonoumous Robots*, 35(2):177–193, October 2013.

D. Caruso, J. Engel, and D. Cremers. Large-scale direct slam for omnidirectional cameras. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 141–148, Sept 2015.

A. Censi, A. Franchi, L. Marchionni, and G. Oriolo. Simultaneous calibration of odometry and sensor parameters for mobile robots. *IEEE Transaction on Robotics*, 29:475–492, 2013.

B. Chen, F. Dachille, and A. E. Kaufman. Footprint area sampled texturing. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):230–240, March 2004.

W. Churchill and P. Newman. Experience-based Navigation for Long-term Localisation. *The International Journal of Robotics Research (IJRR)*, 2013.

J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point ransac for ekf-based structure from motion. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3498–3504, Oct 2009.

R. T. Collins. A space-sweep approach to true multi-image matching. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 1996. IEEE Computer Society.

J. Courbon, Y. Mezouar, L. Eckt, and P. Martinet. A generic fisheye camera model for robotic applications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1683–1688, Oct 2007.

J. Courbon, Y. Mezouar, and P. Martinet. Autonomous navigation of vehicles from a visual memory using a generic camera model. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):392–402, Sept 2009. ISSN 1524-9050. doi: 10.1109/TITS.2008.2012375.

J. Courbon, Y. Mezouar, and P. Martinet. Evaluation of the unified model of the sphere for fisheye cameras in robotic applications. *Advanced Robotics*, 26(8-9):947–967, 2012.

M. Cummins and P. Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

A. Dame and E. Marchand. Second-order optimization of mutual information for real-time image registration. *IEEE Transactions on Image Processing*, 21(9):4190–4203, Sept 2012.

K. Daniilidis. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, 18:286–298, 1998.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1322–1328 vol.2, 1999.

S. Dominguez, B. Khomutenko, G. Garcia, and P. Martinet. An optimization technique for positioning multiple maps for self-driving car's autonomous navigation. In *IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2694–2699, Sept 2015.

J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision*, pages 1449–1456, Dec 2013.

J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849, 2014.

J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In *Transactions on Pattern Analysis and Machine Intelligence*, Mar. 2018.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6): 381–395, June 1981.

D. Fox, W. Burgard, S. Thrun, and A. B. Cremers. Position estimation for mobile robots in dynamic environments. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 983–988, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3877, 2007.

M. Gautier and W. Khalil. Exciting trajectories for the identification of base inertial parameters of robots. In *[30th IEEE Conference on Decision and Control*, volume 1, pages 494–499, Dec 1991.

A. Geiger, M. Roser, and R. Urtasun. Efficient large-scale stereo matching. In *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part I*, ACCV'10, pages 25–38, Berlin, Heidelberg, 2011. Springer-Verlag.

C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical applications. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 445–461, London, UK, UK, 2000. Springer-Verlag.

C. Giovannangeli, P. Gaussier, and G. Desilles. Robust mapless outdoor vision-based navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3293–3300, Oct 2006.

S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *CoRR*, abs/1702.03920, 2017.

C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys. Real-time direct dense matching on fisheye images using plane-sweeping stereo. In *Proceedings of the 2Nd International Conference on 3D Vision - Volume 01*, pages 57–64, Washington, DC, USA, 2014. IEEE Computer Society.

C. Harris and M. Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

L. Heng, B. Li, and M. Pollefeys. Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1793–1800, Nov 2013.

H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 807–814 vol. 2, June 2005.

H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, Feb 2008.

J. D. Hobby. Rasterization of nonparametric curves. *ACM Trans. Graph.*, 9(3):262–277, July 1990.

S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM.

S. D. Jones, C. Andresen, and J. L. Crowley. Appearance based process for visual navigation. In *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 2, pages 551–557 vol.2, Sep 1997.

J. Kannala and S. S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28:1335–1340, 2006.

C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *IROS*, pages 2100–2106. IEEE, 2013.

B. Khomutenko, G. Garcia, and P. Martinet. An enhanced unified camera model. *IEEE Robotics and Automation Letters*, 1(1):137–144, Jan 2016a.

B. Khomutenko, G. Garcia, and P. Martinet. Direct fisheye stereo correspondence using enhanced unified camera model and semi-global matching algorithm. In *14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–6, Nov 2016b.

B. Khomutenko, G. Garcia, and P. Martinet. Exciting trajectories for extrinsic calibration of mobile robots with cameras. In *20th The IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1–6, Oct 2017.

V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 508–515 vol.2, 2001. doi: 10.1109/ICCV.2001.937668.

T. Kos, I. Markezic, and J. Pokrajcic. Effects of multipath reception on gps positioning performance. In *Proceedings ELMAR*, pages 399–402, Sept 2010.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, Washington, DC, USA, 2011. IEEE Computer Society.

M. I. A. Lourakis and A. A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, FORTH-ICS, 2004.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, Feb 2016.

C. McManus, B. Upcroft, and P. Newman. Learning place-dependant features for long-term vision-based localisation. *Autonomous Robots*, 39(3):363–387, Oct 2015.

C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *IEEE International Conference on Robotics and Automation*, pages 3945–3950, April 2007.

M. Meilland, A. I. Comport, and P. Rives. Dense omnidirectional RGB-D mapping of large scale outdoor environments for real-time localisation and autonomous navigation. *Journal of Field Robotics*, 32(4):474–503, June 2015.

M. Milford, G. Wyeth, and D. Prasser. Ratslam: a hippocampal model for simultaneous localization and mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 403–408. IEEE, 2004.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, Mar. 1970.

D. Nistér and H. Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168, 2006.

R. Orghidan, J. Salvi, and E. Mouaddib. Modelling and accuracy estimation of a new omnidirectional depth computation sensor. *Pattern Recognition Letters*, 27(7):843–853, 2006.

A. Radgui, C. Demonceaux, E. Mouaddib, M. Rziza, and D. Aboutajdine. Adapted approach for omnidirectional egomotion estimation. *IJCVIP*, 1(1):1–13, 2011.

S. Raj Bista, P. Robuffo Giordano, and F. Chaumette. Appearance-based Indoor Navigation by IBVS using Line Segments. *IEEE Robotics and Automation Letters*, 1(1):423–430, Jan. 2016a. Also presented in IEEE Int. Conf. on Robotics and Automation, Stockolm, Sweden.

S. Raj Bista, P. Robuffo Giordano, and F. Chaumette. Appearance-based Indoor Navigation by IBVS using Mutual Information. In *IEEE Int. Conf. on Control, Automation, Robotics and Vision, ICARCV*, Phuket, Thailand, Nov. 2016b.

M.-A. Raúl, M. J. M. M., and T. J. D. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

P. Renaud, N. Andreff, G. Gogu, and M. Dhome. Optimal pose selection for vision-based kinematic calibration of parallel mechanisms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2223–2228, Oct 2003.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag.

S. Roy, J. Meunier, and I. Cox. *Cylindrical rectification to minimize epipolar distortion*, pages 393–399. 1997.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergent stereo in autonomous navigation: From bees to robots. *International Journal of Computer Vision*, 14(2):159–177, Mar 1995.

D. Scaramuzza and A. Martinelli. A toolbox for easily calibrating omnidirectional cameras. In *In Proc. of the IEEE International Conference on Intelligent Systems, IROS06*, pages 5695–5701, 2006.

D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 195–202, Washington, DC, USA, 2003. IEEE Computer Society.

M. Schönbein, T. Strauß, and A. Geiger. Calibrating and centering quasi-central catadioptric cameras. In *IEEE International Conference on Robotics and Automation*, pages 4443–4450, May 2014.

J. Shi and C. Tomasi. Good features to track. Technical report, Ithaca, NY, USA, 1993.

P. Sturm and S. Maybank. On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, USA*, pages 432–437, Juin 1999.

R. Swaminathan, M. D. Grossberg, and S. K. Nayar. Caustics of catadioptric cameras. In *Proceedings Eighth IEEE International Conference on Computer Vision*, volume 2, pages 2–9 vol.2, 2001.

R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6565–6574, July 2017. doi: 10.1109/CVPR.2017.695.

C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *Int. J. Rob. Res.*, 26(9):889–916, Sept. 2007.

K. Zhu, M. Butenuth, and P. d'Angelo. Comparison of dense stereo using cuda. In *Proceedings of the 11th European Conference on Trends and Topics in Computer Vision - Volume Part II*, ECCV'10, pages 398–410. Springer-Verlag, 2012.

Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364, 2017.