Cumulative methods for image based driver assistance systems : applications to egomotion estimation, motion analysis and object detection

Qiong Nie

UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE : Sciences et Technologie de l'Information, des Télécommunications et des Systèmes

Institut d'Electronique Fondamentale (IEF)

DISCIPLINE : Physique

THÈSE DE DOCTORAT

soutenance prévue le 12/06/2015

par

Qiong NIE

Cumulative methods for image based Driver Assistance Systems: applications to egomotion estimation, motion analysis and object detection

Directeur de thèse :	Alain MERIGOT	Professeur (IEF, Université Paris-sud)
Co-Directeur de thèse :	Samia BOUCHAFA	Professeur (IBISC, Université d'Évry Val d'Essonne)
Composition du jury :		
Rapporteurs :	Didier AUBERT	Directeur de Recherche (IFSTTAR, LEPSIS)
	Séverine DUBUISSON	HDR (Université Pierre et Marie Curie)
Examinateurs :	Vincent FREMONT	HDR (Heudiasyc, Universit de Technologie de Compigne)
	Sylvie LELANDAIS	Professeur (IBISC, Université d'Évry Val d'Essonne)

Acknowledgements

First and foremost I want to thank my advisors, Professors Alain Merigot and Samia Bouchafa, for supporting me during these past five years. Alain has always been supportive and patient, even during tough times in my PhD study. He can always understand my problems and help me to overcome them. I am also very grateful for his scientific advice, insightful discussions and suggestions. Samia is someone you will instantly love and never forget once you meet her. She has given me many insightful suggestions about the research. She has also shared her useful experience as a mother. I am thankful for the excellent example she has provided as a successful professor and also a lovely mother.

I also would like to thank my committe members, Prof. Sylvie Lelandais , Prof. Didier Aubert, Dr. Severine Dubuisson and Dr. Vincent Fremont. I want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions.

I want to thank present and past members of Autonomous Systems Team, thanks to Abdelhafid Elouardi for helping to finish my teaching tasks in the science faculty, Marius Vasiliu for sharing his sequence database, Yasser Almehio and Adrien Bak for all their helps when i started my PhD study. I also want to thank Agnes Priou. I really appreciate to work with you in IUT Cachan.

A very special word of thanks goes for my parents, who have sacrificed their lives for me and provided unconditional love and care. I love them so much, and i would not have made it this far without them.

The best outcome from these past five years is finding my best friend, soul-mate, and husband. Zhengnan has been a true and great supporter and has unconditionally loved me during my good and bad times. These past several years have not been an easy ride, both academically and personally. I truly thank Zhengnan for sticking by my side, even when I was irritable and depressed. I feel that what we both learned about life strengthened our commitment and determination to each other and to live life to the fullest.

The last word goes for Estelle, my baby girl, who has been the light of my life for the last two years and who has given me the extra strength and motivation to get things done. This thesis is dedicated to her.

Table des matières

Table des figures

Liste des tableaux

1 General Introduction

Robotics, as one of mankind's greatest accomplishments, is widely used in many military and civilian applications. In military applications, robots, such as Unmanned Aerial Vehicles (UAVs) or Unmanned Ground Vehicle (UGVs), are usually deployed to achieve missions that are hazardous for human. Robots for civilian applications could be domestic (robot vacuums, robot toys) or industrial (intelligent vehicles, surveillance cameras).

Since vehicles become a necessary of life, encompassing almost every aspect of our daily lives, traffic situation is of course more and more difficult. With such traffic, fatalities due to vehicle accidents are also enormous. There is an estimation of 1.24 million deaths worldwide in the year 2010. That is one person killed every 25 seconds. So, improving safety in order to reduce the number of accidents is the main motive of designing intelligent vehicles.

Simple systems such as safety belts, air bags, etc. were developed to protect people's lives. However, these systems are only able to reduce the severity of injury when accidents happen. They do not provide any help to reduce the number of accidents. As a consequence, researches that focus on intelligent vehicles have shifted from passive safety to active safety.

Active Safety Systems or Advanced Driver Assistance Systems (ADAS) are usually composed by an acquisition system to get some information from different sensors, a processing phase to deliver warnings to the driver, and sometimes a react phase to control the acceleration, braking, etc. Using both proprioceptive sensors (eg. Inertial measurement unit, GPS) and exteroceptive sensors (LIDAR, RADAR) is common in ADAS systems even if most intelligent vehicles (eg. The Google Driverless Car) use LIDAR mainly since it can generate a high resolution 3D map of the environment. As sensors provides heterogeneous measurements, data fusion from different sensors is required to achieve different tasks in ADAS systems. For example, [4] proposed an obstacle detection and

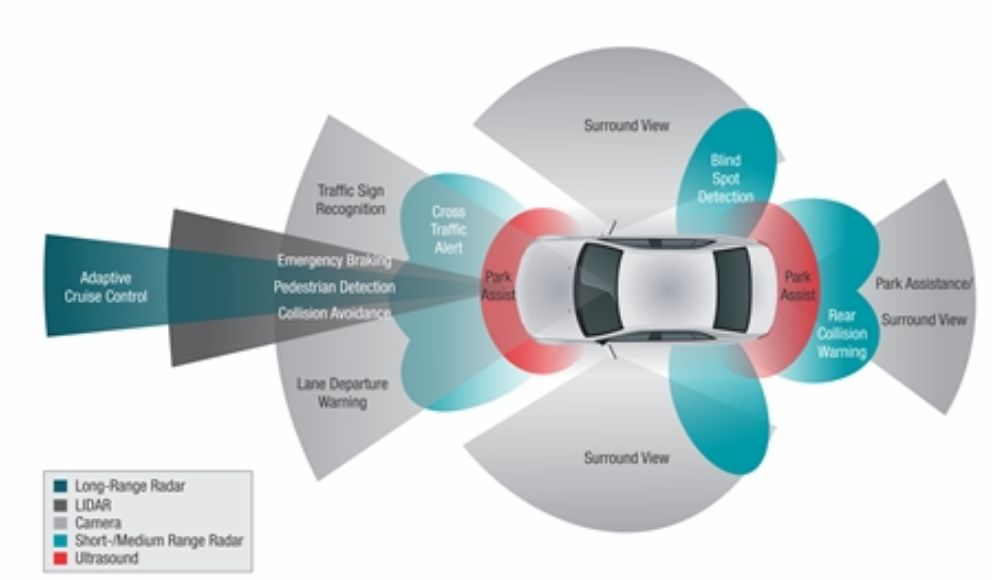


FIGURE 1.1: The usages of different sensors for ADAS systems[1]

tracking algorithm based on fusing data from more than a dozen sensors. [5] performed vehicle localization and obstacle detection using data fusion from both proprioceptive and exteroceptive sensors. Along with the development in image processing, cameras are increasingly supplemented by other sensing technologies to expand ADAS capabilities and also to enhance the robustness of results. In [6], a precise road estimation algorithm is based on measurements provided from camera, radar, wheel speed sensors, and IMU, where camera is used for detecting lane marking. As plenty of information can be exploited from images (visual attributes, motion, disparity), moreover cameras are really inexpensive compared with LIDAR [1], we do believe that vision-based ADAS system has a promising future. In fact, many researches have already proved abilities of purely vision-based systems. For example, stereo vision is used now to describe the environment or detect obstacles by computing occupancy grids [7–9]. The road, as an important source of information could be detected through feature extraction and classification [10–12]. Motion estimation is required for many tasks in the ADAS systems such as egomotion estimation, moving-object detection, and 3-D reconstruction while optical flow shows its potential. [13] studied the influence of vehicle speed and scene texture on optical-flow accuracy using synthetic images. However, optical flow is rarely implemented on intelligent vehicles. It is usually combined with other sensors like radars [14] or stereovision [15], and plays consequently just a supporting role. The main reason is that existing optical flow estimation results are not always convincing, especially when vehicle motion is large and scene textures are rare.

1. *The LIDAR system mounted on the google driverless car costs $70,000. The camera hardware can cost just tens of dollars*

An important issue of intelligent vehicles is to detect and identify from images relevant objects in the driving environment (obstacles on the road for instance) for ensuring navigation in safe conditions. As explained before, stereo-vision based systems are widely used to achieve this objective. However, the resulting disparity map require a pre-processing step of calibration and image rectification. Moreover, we consider that stereovision is not always necessary as it is possible to exploit first monocular vision and motion analysis.

Contributions

This thesis deals with monocular vision for object detection in the context of automatic driver assistance systems. Our proposed method use 2D motion information (optical flow and Focus Of Expansion) to define and to exploit specific voting spaces in order to detect objects. Optical flow estimation and FOE estimation are considered as two preliminary steps. The success of these two steps conditions the quality of detection. Hence, we have to deal with two difficulties that are specific to the application : homogeneous regions and large displacements. These difficulties are both problematic for estimating optical flow with enough precision. In this thesis, we propose a model-based motion estimation method that exploits available a priori knowledge from other sensors to compensate dominant flow in order to facilitate estimation of the remaining part of motion using a classic optical flow method. We focus also on the second preliminary step of our detection approach : FOE estimation as its location is required as an input of our algorithm. We have studied and compared different FOE estimation methods. We propose three different methods that we have adapted to object detection. Finally, we have proposed different decision strategies to detect more precisely obstacles using for instance an velocity histogram splitting approach.

Thesis outline

This thesis is structured as follows :

Chapter 2 begins by an introduction of existing vision-based obstacle detection methods for ADAS. Then, it focuses on plane-model based detection methods. A stereo-vision based method called "v-disparity" will be explained in details as our proposed monocular detection method - "c-velocity" was inspired from it. Then, we will describe the most important part of this chapter and also the core of this thesis - the "c-velocity" approach. The principle, the required information (inputs), the detection process, optimizations,

and finally the comparison with the "v-disparity" approach will be presented. In order to combine disparity information with motion to detect obstacles, we define the "v-velocity" approach as a variant of c-velocity and v-disparity. We will show that "v-velocity" is easy to combine with "v-disparity".

Chapter 3 gives a the current state of the art for 2D motion estimation methods. We will present evaluation benchmarks and more particularly ground truth optical flow methods. Finally, we will propose our model-based optical flow estimation method. The results will be compared with other existing methods.

Chapter 4 will starts with a general introduction on existing FOE estimation methods. Three estimation methods will be proposed and tested both on synthetic toy examples and real images.

Chapter 5 will present an iterative histogram splitting method to adapt the "c-velocity" approach in order to detect several obstacles that belong to different models.

2

C-velocity : Principle and Implementation

Contents

2.1 Background

Advanced Driver Assistance Systems (ADAS) are one of the fastest-growing segments in automotive research. Their objective is to provide safe and intelligent driving tools. Among them, potential obstacles (static or mobile) detection as well as scene reconstruction is essential for navigation safety. These functionalities need an efficient sensor technology or vision/camera systems. Especially for the former, many different types of sensors (exteroceptive or proprioceptive) can be mounted on vehicles to capture environment information. However, sensors always provide imprecise and missing information. So, a natural solution is the multi-sensor cooperation [5, 16] . Vision based systems are also more and more used in intelligent vehicles because of their wealthy information and low cost. Their contributions are exhibited by many publications. Since all these proposed approaches are so different and diverse, a complete synthesis of recent techniques based on vision systems are rarely found or even does not exist. For this reason, in the following subsection, we will try to make a detailed elaboration about the existing obstacle detection approaches using vision systems for intelligent vehicles and also classify them into four categories.

2.1.1 An introduction of vision based obstacle detection in the application of ADAS

Obstacle detection is an essential task for intelligent vehicles since it is very important for obstacle avoidance or inter distance management. Perceptive systems are widely used for this purpose and many different approaches are proposed. However, we can generally categorize them into four classes according to the different type of information that is extracted from perceptive systems. For example, approaches using visual attributes from an image are called "2D approaches (x, y)" since only 2D image information is exploited. When dealing with the image sequences, motion information could be extracted to help obstacle detection. This kind of approaches are called "motion based 3D approaches (x, y, t)". Besides monocular systems, stereo systems aim at exploiting disparity information and turning it into depth information for obstacle detection. These stereo based approaches are then considered as "structure based 3D approaches (x, y, z)". The last type of approaches exploits all the available information (temporal and spatial) for the detection, so these approaches, which combine the above approaches, are called 4D approaches (x, y, z, t). In the next part of this subsection, an introduction will be given for all these approaches.

2.1.1.1 "Model" based - 2D approaches (x, y)

A great deal of information can be exploited from a 2D image for object detection. Especially in the case of obstacle detection in autonomous environment, models of visual attributes can be defined to distinguish obstacles from other "objects". And different information such as the symmetry [17], textures [18] or colors [19] can be exploited for these model-based detection methods. Such type of approach is rather adequate for detecting rigid obstacles like vehicles, but not suitable for pedestrian extraction. Moreover, the detection is easily disturbed by view point variation, illumination difference, occlusion, deformation, background clutter, etc. Thus, learning-based classification approaches are applied for the detection. In these approaches, a set of examples are firstly used for an off-line training. The classification relies on different features that characterize different objects. The performance of these methods depends on many factors : training sets, feature sets and used classifier. Generally a various training set makes the classifier more flexible to differentiate objects. With a view to feature sets, different features and descriptors are proposed, like Histograms of Oriented Gradients (HOG) [20], Joint Ranking of Granules [21], Haar Wavelets [22] and Principal Component Anaysis (PCA) [23] - of which HOG is widely used for pedestrians detection. In fact, the selection of feature can be unique or multiple. When choosing classifiers, we have Support Vector Machine (SVM) [24], neural network [25], Adaboost[26] and cascading classifier [27].

2.1.1.2 "Structure" based - 3D approaches (x, y, z)

These approaches are based on the structure estimation which characterize potential obstacles by exploiting another camera. Stereo-vision can estimate the depth information of perceived obstacles. In these studies, obstacles are assumed as fronto-parallel planes. This assumption facilitates obstacle detections, especially when the stereo-vision system is perfectly calibrated and rectified. In the example of [2], authors defined a new space in which the representation of fronto-parallel planes became into lines. The detection of fronto-parallel planes are then simplified to the line detections that can be easily done by Hough Transform. This method will be presented in detail since the concept of detecting objects from a new space deeply inspired us for our studies. Different from this plane similarity detection, the methods that using map construction, or more precisely occupancy grids mapping also obtained an ideal effect on obstacle detection [28, 29]. The vehicle environment is presented by a two dimensional lattice of rectangular cells. Each cell is associated with a probability value to assess whether this cell is occupied by obstacles. And the computation of such probability value depends on the sensor information. These approaches are thus very suitable for the cooperation of different sensors, even with different modalities (Sonar, LIDAR, RADAR, stereo-vision). In fact,

the collaboration between LIDAR and stereo-vision is a popular way for ADAS. LIDAR information is used to provide detection hypotheses and then stereo-vision system will confirm these hypotheses [30]. Since the objective of obstacles detection is to make sure that vehicles can avoid these obstacles and drive on the free space. Instead of localizing obstacles, we can also identify the free space around. Then potential danger avoidance subject becomes into free space estimation for navigation [31].

2.1.1.3 "Motion" based - 2D+t approaches (x, y, t)

The input of these approaches is a 2D motion field [1] estimation, projection of the 3D motion onto the image. The computation of such motion field can be found in Appendix A.1. Their equations are shown in Equation 2.1.

$$\begin{cases} u = \frac{xy}{f}\Omega_X - (\frac{x^2}{f} + f)\Omega_Y + y\Omega_Z + \frac{xT_Z - fT_X}{Z} \\ v = (\frac{y^2}{f} + f)\Omega_X - \frac{xy}{f}\Omega_Y - x\Omega_Z + \frac{yT_Z - fT_Y}{Z} \end{cases} \tag{2.1}$$

The above expression of the 2D motion field is based on a perspective camera model. Where $\mathbf{T} = (T_X, T_Y, T_Z)$ and $\Omega = (\Omega_X, \Omega_Y, \Omega_Z)$ are respectively translational and rotational velocities of the camera. (x, y) is the image coordinate of the observed static [2] 3D point $\mathbf{P}(X, Y, Z)$, f is the focal length of the camera. Z is the depth. From the above equations, one can conclude that :

- Depth information is needed for 2D motion estimation.
- A 2D Motion vector has two components, one depends only on 3D translation, the other only on 3D rotation.
- Only the translational component depends on depth information.
- All discontinuities of 2D motion are due to depth variation.
- Because of the scaling ambiguity (T and Z can only be derived up to a scale factor), a 2D motion vector can result from different 3D motions.

Considering of the above conclusions and due to the nonlinearity of equations, some early approaches tried to simplify the projection model [32]. In the case of large focal length and objects close to the optical axis, a weak-perspective model can be used as linear approximation. This 3D→2D projection can be presented by an orthographic projection followed by isotropic scaling. When objects are far from the camera or object sizes are smaller than their distance to projection center, a para-perspective projection can also be used to simplify the equations [33]. Without any assumption about the scene, we can also use spherical projection to adapt vector representation.

1. *In fact we can only exploit apparent motion from images*
2. *If* $\mathbf{P}$ *is not static, then* T *and* Ω *represent the relative motion of the camera to* P

Simplification can be not only done on project model, but also on motion model. In other words, it is not necessary to always consider the six motion parameters (T_X, T_Y, T_Z, Ω_X, Ω_Y, Ω_Z). For example, only longitudinal translation could be considered or sometimes one or two rotation angle [3] are added into the assumption[34, 35]. Of course, reducing the degrees of freedom is not the only possibility, one can also separate the translation component (that depends on depth) from the rotation component by exploiting parallax (motion parallax, affine motion parallax, plane+parallax). Since parallax is due to depth discontinuities, the relative motion of two instantaneously coincident points depends only on the translational component. So in "Plane+parallax" methods, the camera rotation is removed in some image regions where the depth variation is not significant. And then the camera translation can be easily computed from the parallax. Besides, we can also obtain the translation information by means of Focus Of Expansion (FOE) position [36–38].

Researches on the estimation of vehicle-mounted camera's 3D motion (egomotion) and scene reconstruction (structure from motion) have increased in recent years. Therefore, numerous classifications according to various criteria exist. In this report, we will classify them into three principal categories : discrete approaches, continue approaches and direct approaches.

— **Discrete** approaches are based on feature matching between two successive images. Corresponding points are related using the fundamental matrix, which includes camera motion parameters and camera intrinsic parameters. Therefore, the problem becomes a linear algebra problem since the fundamental matrix have to be estimated. Several approaches have been proposed to solve this problem that differs from the outliers management [39–41].

— **Continuous** approaches make use of optical flow. According to the relation between optical flow and 3D motion (see Equation 2.1 [4]), the motion parameters as well as depth map are computed using optimization algorithms. The performance of these approaches depends directly on the quality of the flow.

— In **direct** approaches, the 3D motion parameters are determined "directly" from the brightness constancy constraint. Classical optimization approaches are used for the estimation of motion parameters. **Continuous** approaches and **direct** approaches are also called global methods as the whole image is used for the estimation. It is intuitive that these global methods are more robust than **discrete** approaches since much more information is used for the estimation [42, 43].

3. *Usually the yaw is to model the turn*
4. *Here we consider optical flow is an approximation of 2D motion field*

2.1.1.4 "Structure/Motion" cooperation based 4D approaches (x, y, z, t)

The idea of cooperation between motion and structure estimation is not new since such collaboration can exploit available information from both space and time. In fact, the researches about this combination started in the early 2000s just after the computational power allows to tackle these two processes at once. There exist many significant researches since then, such as the approaches proved in [44] and [45]. While [44] focused on the exhibition of an invariant - ratio of the optical flow norm and depth value. Another way to the fusion of depth and motion information is the 6D-Vision method presented in [45]. The idea is to track points with depth known from stereo vision over two and more consecutive frames and to fuse the spatial and temporal information using Kalman Filters. Another approach is to use occupancy grid mapping because it facilities the intergration of different sensors. This formalism can be exploited to construct a representation of observed scene [46] or just enrich the temporal information [47, 48]. The "hottest" approaches to combine spatial and temporal information are those who evaluate scene-flow. To do this, one can track the interest points [49] or integrate the stereo vision into classic Horn&Schunck optical flow estimation method [50, 51]. Then from the matching fields, classic segmentation methods can be used to obtain the scene representation based on the apparent motions of objects.

2.1.2 Obstacle detection on planar road

The "obstacles" in the case of autonomous driving refer to the structures that block the path by sticking out of the road. Hence, most of the obstacle detection methods start with road extraction. This allows to define a free driving corridor and also to refine a Region of Interest (ROI) for obstacle detection. Some assumptions are usually defined in order to facilitate the road surface locating process. For instance, [52] estimated the homography of the road by matching a set of point features (corners), this homography was then used to compensate the motion of the road and to detect obstacles as areas in the image that appear non-stationary after the motion compensation. [53] made use of apparent orientations of surface in the image in order to detect the presence of obstacles. Both applications were based on the assumption that road is planar surface. Some techniques also assumed that there was a constant plane [54]. A more complex road model like estimating the inclination of road plane was presented in [55]. Recently a B-spline Modeling of road surface proved also to be efficient for road detection [56] . However, [2] proposed a robust road detection method in stereovision by construction and investigation of a "v-disparity" image. From the presentation of [2], we know that the pixels from a planar road surface have the same disparity if they are located in the same row of the image. And the disparities in the different image row can be computed by

$$d(v) = a \times v + b \tag{2.2}$$

Where v is the image row, a and b are parameters depending on camera height and tilt angle. In a new representation space called the "v-disparity" image [5], the representation of planar road becomes a straight line that is easily extractable. Moreover, the obstacles, which are considered as fronto-vertical plane, project on the "v-disparity" image as quasi-vertical straight lines that intersect with the road line [6]. This method inspired us on two points. 1) the first one concerns the plane modeling simplification. In order to facilitate the detection, objects can be modeled as different planes. For instance, the road is usually assumed as horizontal plane. The vehicles above the road is assumed as fronto-vertical plane. 2) The other concerns the transformation onto another space that facilitates detection. In fact, obstacles' shapes are very variant. Instead of detecting objects directly from images, maybe we can first transform them into another space where objects become into easily detectable shapes (for example, lines, parabolas, etc.). In [2], the construction of transformed space is based on disparities. When considering motion instead of stereo, disparity could be compared with velocity in some motion conditions and camera configuration. Then a "c-velocity" method is proposed in [57]. The whole thesis research is about this "c-velocity" method, so we will present first the basic idea of this approach and then discuss in more details each factors that will influence the performances of this method.

The structure of this chapter is the following : first, a brief introduction about the "v-disparity" approach will be given to fetch out the "c-velocity" concept. Then we will present the "c-velocity" in detail, including the mathematic demonstrations.

2.2 V-disparity : a road geometry representation in stereovision

The core of the "v-disparity" approach is to transform 3D plane detection on the scene into a simple line detection in a new representation space. Such transform is based on the assumption that the road is a planar surface or at least a piecewise planar surface and obstacles [7] are vertical or quasi-vertical planes. Moreover, this method does not need any extraction of lane-markings but exploits all the relevant information in the image

5. *Disparity value is on the x-axis and image row is on the y-axis*
6. *Since obstacles (vehicles) are located above the road*
7. *Moving vehicles on the road are considered as obstacles*

(texture of the road, shadows, road edges, etc.). From the new space that is called "v-disparity", we can not only detect the road surface but also estimate the vehicle pitch and the relative height of the stereo sensor at real time.

In this section, a theoretic introduction about the "v-disparity" approach including the using assumptions will be given. Then, the most important steps of the construction of "v-disparity" image will be shown using a toy example. Finally, an explanation of the transition from "v-disparity" approach to "c-velocity" will be given.

2.2.1 Assumptions and models

Assume a stereo coordinate system (see Figure 2.1) where the two image planes are supposed to belong to the same plane and are at the same height from the road. In this case, the epipolar lines of two images are parallel and horizontal. Let us define some parameters :

— θ : angle between the optical axis of the cameras and the horizontal
— h : camera height from the ground
— b : distance between the two cameras

Here we should distinguish three coordinate frames : R_a (abosolute), R_{cl} (left camera) and R_{cr} (right camera).

On the basis of Figure 2.1, the transformations from the absolute frame to the camera frame is a combination of translations T_i ($i = cl$ or cr, ε_i = -1 in R_{cl} or 1 in R_{cr}) and rotations $\mathbf{R}$ (see Equation 2.3).

$$\underbrace{\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix}}_{\mathbf{P_i}} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}}_{R} * \Bigg(\underbrace{\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}}_{P} + \underbrace{\begin{pmatrix} \varepsilon_i \frac{b}{2} \\ h \\ 0 \end{pmatrix}}_{T_i} \Bigg) \tag{2.3}$$

Using the pin-hole camera model, the 2D projection on the image plane of a given point $\mathbf{P}(X_i,\ Y_i,\ Z_i)$ in the left camera frame R_{cl} or in the right camera frame R_{cr} can be expressed by Equation 2.4 :

$$\begin{cases} u_i = \alpha \frac{X_i}{Z_i} + u_0 \\ v_i = \alpha \frac{Y_i}{Z_i} + v_0 \end{cases} \tag{2.4}$$

Where $(u_0,\ v_0)$ is the projection of the optical center on the image, $\alpha = f/t$ (f is the focal length and $t = t_u = t_v$ the pixel size). Using Equation 2.3 and Equation 2.4, the

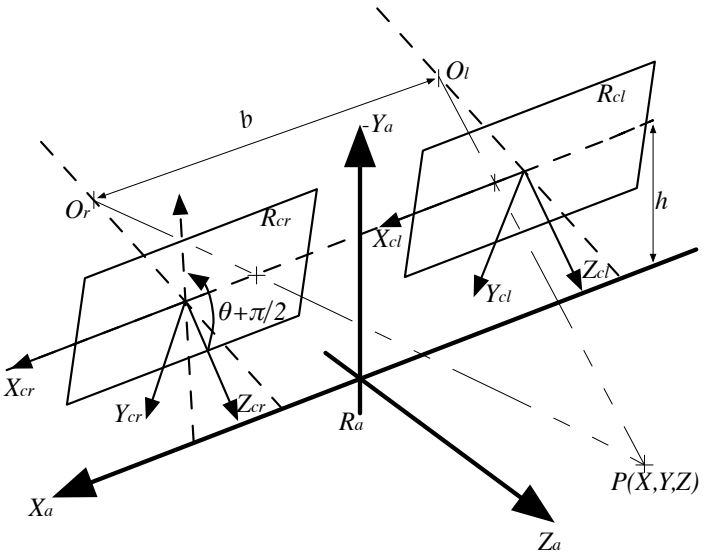


FIGURE 2.1: The calibrated stereo coordinate system [2]

coordinates of the projection of a point $\mathbf{P}(X, Y, Z)$ in R_a on the left and right images are the same $v_l = v_r = v$:

$$v = \frac{[v_0 \sin\theta + \alpha\cos\theta](Y+h) + [v_0\cos\theta - \alpha\sin\theta]Z}{(Y+h)\sin\theta + Z\cos\theta} \tag{2.5}$$

Moreover, we can also compute the disparity Δ of the point $\mathbf{P}$ by

$$\Delta = u_r - u_l = \frac{\alpha b}{(Y+h)\sin\theta + Z\cos\theta} \tag{2.6}$$

Equation 2.6 is a general disparity equation adaptable to any 3D point in R_a. Now let us consider a plane with equation $Z = aY + d$, the disparity is then expressed as a function of v :

$$\Delta_M = \frac{b}{ah-d}(a\cos\theta + \sin\theta)(v - v_0) + \frac{b}{ah-b}\alpha(a\sin\theta - \cos\theta) \tag{2.7}$$

From the above equation, a linear relationship can be easily found between the disparity Δ_M and the image row v. More particularly, when $a = 0$, the plane equation becomes $Z = d$ (obstacle plane) and the equation is :

$$\Delta_M = -\frac{b}{d}\sin\theta(v - v_0) + \alpha\cos\theta \tag{2.8}$$

When $a \to \infty$, the projection equation of a horizontal plane $Y = 0$ (road plane) becomes :

$$\Delta_M = \frac{b}{h}\cos\theta(v - v_0) + \frac{b}{h}\alpha\sin\theta \tag{2.9}$$

Moreover, the pitch and relative height of a stereo sensor can also be estimated from the "v-disparity" image. If c_r is the line slope and v_{or} is the image row value for $\Delta = 0$, then the expressions of θ and h can be obtained by

$$\theta = \arctan(\frac{v_0 - v_o r}{\alpha}) \tag{2.10}$$

$$h = b\frac{\cos\theta}{c_r} \tag{2.11}$$

2.2.2 Construction of the "v-disparity" image

Since object detection is performed in the "v-disparity" image, the construction of such "v-disparity" image from the original camera image is an important point. To tackle this problem, the image raw value and the disparity value should be known for each pixel at first. The image raw is known directly from the pixel coordinates. The disparity map is constructed using a dense image matching method. In the following subsections, we will present the semi-global matching method [58] that is used in [2] and then explain how to construct the "v-disparity" image.

2.2.2.1 Dense image matching

From the requirement of the "v-disparity" image construction, a dense disparity map should be estimated at first. A classic disparity computation process of the stereo vision system is divided into three steps : calibration, rectification and stereo correspondence. While the calibration is an off-line procedure that aims at finding the intrinsic parameters of the cameras like the focal length f, image center (u_0, v_0), lens distortion parameters, and the extrinsic parameters which allow to align the cameras. All these parameters are obtained by processing several stereo pairs of chessboard patterns. Efficient algorithms can be found both in Opencv [59] and Matlab [60]. Then the original images captured from stereo cameras are rectified into a standard form and at the same time removing lens

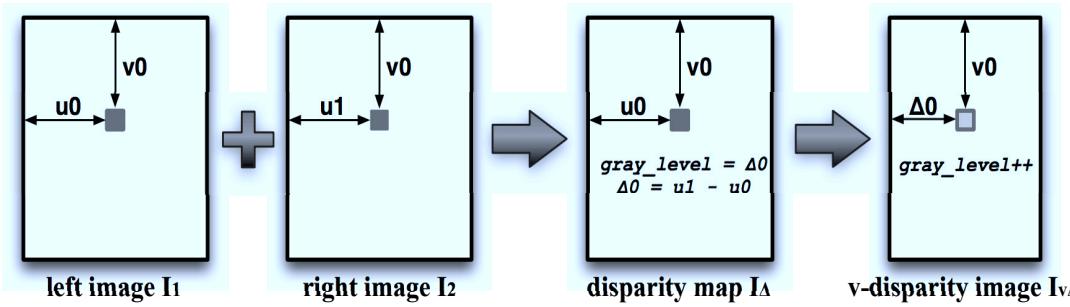


FIGURE 2.2: Construction of the "v-disparity" image

distortions. The pixel correspondences are carried out on the rectified images. [2] used a semi-global matching algorithm [58] to compute the disparity map. This matching method is to find the best disparity assignment by minimizing a cost function over the image pairs. While matching cost calculation is based on Mutual Information (MI), which is insensible to recording and illumination changes. This method can also produces accurate and robust disparity results for object boundaries.

2.2.2.2 "V-disparity" image construction

The construction of the "v-disparity" image is based on a voting process. Once we compute the disparity value for each pixel by the semi-global matching algorithm, a disparity map is drawn to construct the "v-disparity" image (see Figure. 2.2). For each pixel, its projection in the "v-disparity" image is determined by the y-coordinate ("v" value) and its corresponding gray level (disparity value) in the disparity map. In other words, the pixel $(u0, v0)$ with disparity $\Delta 0$ in the left image I_1 is projected to the pixel $(\Delta 0, v0)$ in the "v-disparity" image. As different pixels may be projected to the same pixel[8], an accumulation is done to count pixel votes. It means that in the "v-disparity" image, the pixel value corresponds to vote number.

In order to illustrate the process of constructing the "v-disparity" image, we show an example using synthetic images (see Figure 2.3). These images (see Figure 2.3(a) and 2.3(b)) are synthesized by the driving simulator SiVIC [61]. The purpose of such simulator is to reproduce realistically 3D dynamic scenes that include roads, vehicles, their embedded sensors and pedestrians. It means that input ground truth information like egomotion and scene depth are already known. When using these image sequences, we do not need any matching method since disparity values are directly computed from Equation 2.6. Once we get the disparity map I_Δ (see Figure 2.3(c)), the "v-disparity" image $I_{v\Delta}$ (see Figure 2.3(d)) is built by accumulating the pixels of the same disparity

8. *As explained before, pixels that belong to the same plane and that are located in the same image raw will be projected on the same position in the "v-disparity" image*

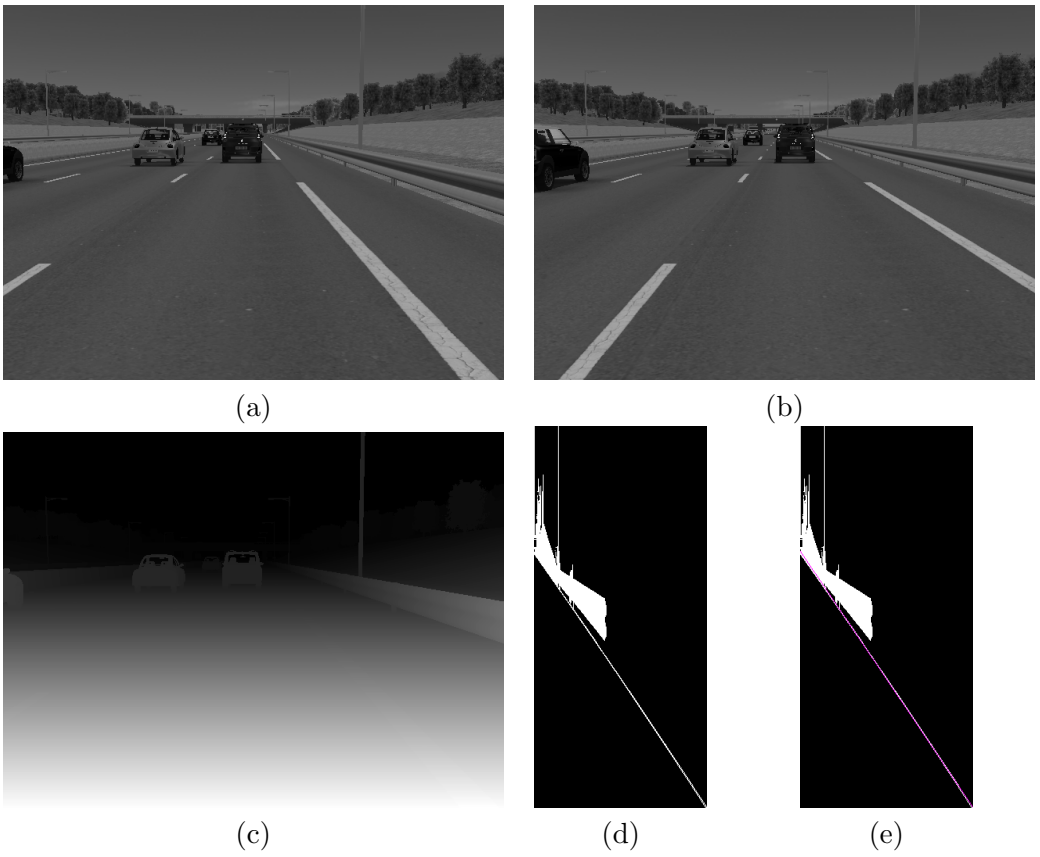


FIGURE 2.3: A synthetic example used to build the "v-disparity" image : (a) Left synthetic image ; (b) Right synthetic image ; (c) Disparity map ; (d) "v-disparity" map ; (e) The red line shows the detected line by Hough Transform

in I_Δ along the $\overrightarrow{v}$ axis. In this example, the road is considered as an horizontal plane, so a large number of votes comes from this road in each raw and reveal an emerging point from the background [9]. All the road points from different raws form a white line which can be easily detected by 2D Hough Transform (see the red line in Figure. 2.3(e)).

2.2.3 From stereo vision to monocular vision, from "v-disparity" to "c-velocity"

Let us recall that the basic idea of the "v-disparity" image is to project pixels of the disparity map along the rows using accumulation. Interesting information about urban scenes can be obtained from this "v-disparity" image. For example, the horizontal road surface appears as a slopping line in the "v-disparity" image while frontal obstacles appear as vertical lines. This idea is also generalized to the other image coordinate by using the u-disparity [62] by several teams, including ours on our autonomous car [63].

9. *The background is black and this point is very white since the vote number is very large*

The problem now is whether this concept can be translated to monocular vision and how to use this idea. The authors in [64] gave us the answer. A monocular approach what was called "c-velocity" is proposed. The idea of this approach is to build a 2D histogram "iso-velocity **c**urve & **velocity**" ("c-velocity" image) by accumulating the pixels with the same (*c*, *velocity*). Here "velocity" represents the image motion and is equivalent to "disparity" in stereovision since both are the apparent shift of pixels between images resulting from the camera position change. "c", corresponding to "v" in the "v-disparity" approach, is also a variable that depend on pixel coordinate and that is call "iso-velocity curves". Our thesis research is based on this method, so the next section will give a complete and exhaustive presentation about this "c-velocity" approach.

2.3 C-velocity : a monocular approach for plane detection

The "c-velocity" approach is, in fact, inspired by two studies : from the first study[65], motion vectors of certain lengths and directions lie on the image at particular loci. Their location and shape depend solely on 3D motion parameters. If motion fields are considered, equal vectors are shown to lie on conic sections. The other one is what we presented in the last section - "v-disparity" approach. According to the above presentation, the disparity of a horizontal plane is constant along a line of stereo pair of rectified images. These make us think about if we can also exploit the iso-velocity curves in the case of monocular vision. Then "c-velocity" approach is proposed by [57].

2.3.1 Assumptions and models

To present the "c-velocity" approach, we should start by the computation of 2D motion field. The motion field is defined as the 2D vector field of velocities of the image points, induced by the relative motion between the viewing camera and the observed scene [66]. In other words, motion field is the projection of 3D motions on the image plane (see Figure 2.4). So motion field can be represented as a function of 3D motions, image coordinates and intrinsic camera parameters (see Equation 2.1).

To avoid depth estimation, we assume that the 3D scene is a set of 3D planes. In fact, planes are largely used as scene approximation in many studies. In the "v-disparity" approach, roads are considered as horizontal planes and obstacles [10] as frontal vertical planes. Such plane approximation allows to remove depth parameter from the motion vector equation. From Appendix B.1, the motion vector of a static planar surface is expressed by Equation 2.12 where the plane equation is $\mathbf{n}^T\mathbf{P} = d$ with $\mathbf{n} = (n_X, n_Y,$

10. *Vehicles in front of the camera*

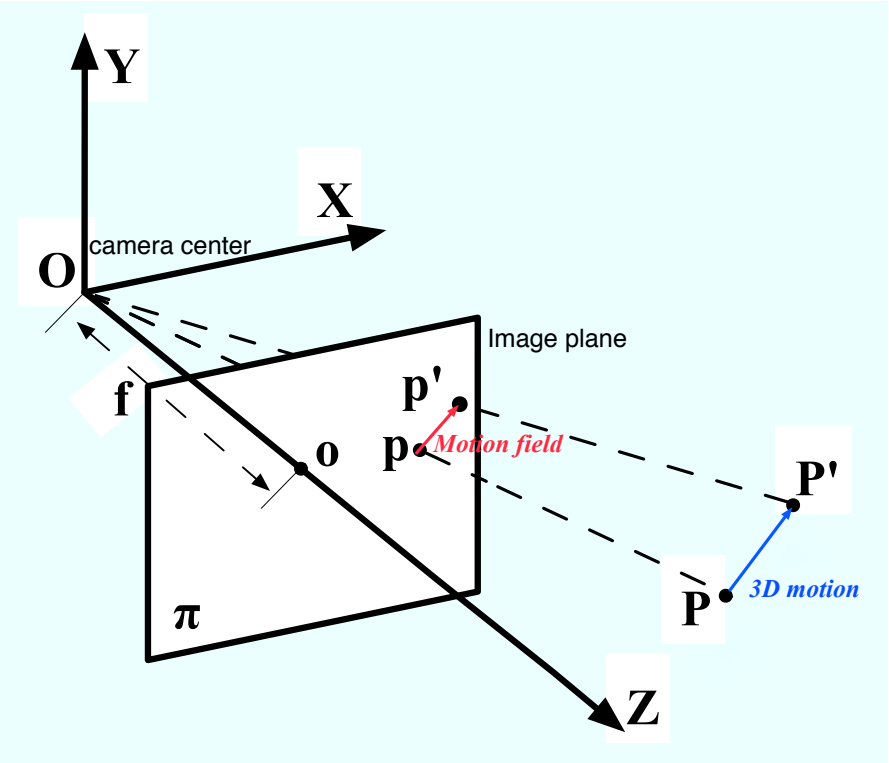


FIGURE 2.4: The motion field of a 3D point **P** when considering a pinhole camera model

n_Z) the unit vector normal to the plane, d the distance "plane to origin" and **P** an arbitrary point (X, Y, Z) in the plane .

$$\begin{cases} u = \frac{1}{fd}(a_1x^2 + a_2xy + a_3fx + a_4fy + a_5f^2) \\ v = \frac{1}{fd}(a_1xy + a_2y^2 + a_6fy + a_7fx + a_8f^2) \end{cases} \tag{2.12}$$

$$\begin{aligned} a_1 &= -d\Omega_Y + T_Zn_X \quad & a_2 &= d\Omega_X + T_Zn_Y \\ a_3 &= T_Zn_Z - T_Xn_X \quad & a_4 &= d\Omega_Z - T_Xn_Y \\ a_5 &= -d\Omega_Y - T_Xn_Z \quad & a_6 &= T_Zn_Z - T_Yn_Y \\ a_7 &= -d\Omega_Z - T_Yn_X \quad & a_8 &= d\Omega_X - T_Yn_Z \end{aligned}$$

We can see that eight parameters (a_1 .. a_8) should be estimated to obtain the motion field of a general plane. But some simplifications could be done under some assumptions. In most driving situations, the longitudinal component of movement is dominant for moving vehicles. So we can ignore or neglect the rotational component as a first approximation. This means $\Omega_X = \Omega_Y = \Omega_Z = 0$. Moreover, objects that compose urban scenes are generally buildings, road, vehicles and pedestrians. They could be considered as planes with specific orientations :

(1) Road → Horizontal plane

FIGURE 2.5: An urban road scene in the city of versailles in France

(2) Buildings in each side of the road → Lateral planes

(3) Obstacles → Frontal planes

  (a) Vehicles in front of the camera → Fleeing/approaching frontal planes

  (b) Crossing pedestrians → Crossing frontal planes

Three different orientations of planes can be considered to represent the above objects that compose urban road scenes. The road is of course considered as horizontal plane, the buildings in both sides of the road are supposed to be lateral planes. Moving obstacles [11] like vehicles, pedestrians are also assumed to be frontal planes. According to the different motions of the obstacles, we divide them into two classes : 1) fleeing or approaching obstacles (usually : vehicles) that have longitudinal motions in direction Z and 2) crossing obstacles (usually : pedestrians) that have a X direction of motion. In order to compute motion vectors for these different orientations of planes, we should first define their plane parameters : the 3D relative motions of the camera and the distances of these planes to origin. They are all given in Table 2.1 where motions of the fleeing/approaching obstacles are defined as (0, 0, T_{Z1}) and motions of the crossing pedestrians are defined as (T_{X1}, 0, 0). With all these parameters, the corresponding motion fields for these planes can be obtained by injecting the respective $\mathbf{T}$ and $\mathbf{n}$ into Equation 2.12. These equations are shown in Table 2.2. For all computation details, one can refer to Appendix B.2.

11. *Obstacles are considered as the objects on the road that will influence the navigation of the vehicle*

TABLE 2.1: Plane parameters of four relevant cases of moving planes

Plane type	n	3D motion	Dist. to origin
(1)	(0,1,0)	$\mathbf{T} = (T_X, T_Y, T_Z)$	d_r
(2)	(1,0,0)	$\mathbf{T} = (T_X, T_Y, T_Z)$	d_b
(3.a)	(0,0,1)	$\mathbf{T} = (T_X, T_Y, T_Z + T_{Z1})$	d_o
(3.b)	(0,0,1)	$\mathbf{T} = (T_X + T_{X1}, T_Y, T_Z)$	d_o

TABLE 2.2: Motion fields of four relevant cases of moving planes

Plane type	Motion field equations
(1)	$u = \frac{T_Z}{fd_r} y(x - f\frac{T_X}{T_Z})$ $v = \frac{T_Z}{fd_r} y(y - f\frac{T_Y}{T_Z})$
(2)	$u = \frac{T_Z}{fd_b} x(x - f\frac{T_X}{T_Z})$ $v = \frac{T_Z}{fd_b} x(y - f\frac{T_Y}{T_Z})$
(3.a)	$u = \frac{T_Z+T_{Z1}}{d_o}(x - f\frac{T_X}{T_Z+T_{Z1}})$ $v = \frac{T_Z+T_{Z1}}{d_o}(y - f\frac{T_Y}{T_Z+T_{Z1}})$
(3.b)	$u = \frac{T_Z}{d_o}(x - f\frac{T_X+T_{X1}}{T_Z})$ $v = \frac{T_Z}{d_o}(y - f\frac{T_Y}{T_Z})$

From Equation **??**, camera motion $\mathbf{T}$ is required to compute the motion field of a point. But as far as we know, estimating of $\mathbf{T}$ is a very tough task and the results are sometimes unsatisfactory in despite of many efforts in this domain [67] [68]. However, this estimation can be avoided from the plane equations in Table 2.2, because only the scale motion factors $\frac{T_X}{T_Z}$ and $\frac{T_Y}{T_Z}$ are needed. Fortunately, the Focus Of Expansion (FOE) point contains these information. In fact, in the case of a camera moving straight (rotations are negligible), the motion field of all static points converge/diverge to a particular point in the image and this is the Focus Of Expansion point. Its expression is shown in Equation 2.13.

$$\begin{cases} x_{FOE} = f\frac{T_X}{T_Z} \\ y_{FOE} = f\frac{T_Y}{T_Z} \end{cases} \tag{2.13}$$

Since each motion direction induces a unique FOE, in the case of moving obstacles (fleeing/approaching or crossing obstacles), the FOE points are computed using the relative motion between the camera and obstacles :

$$\begin{cases} x_{FOE1} = f\frac{T_X}{T_Z+T_{Z1}} \\ y_{FOE1} = f\frac{T_Y}{T_Z+T_{Z1}} \end{cases} \qquad \begin{cases} x_{FOE2} = f\frac{T_X+T_{X1}}{T_Z} \\ y_{FOE2} = f\frac{T_Y}{T_Z} \end{cases} \tag{2.14}$$

TABLE 2.3: Motion fields of four relevant cases of moving planes after adding FOE point

Plane type	Motion expressions with FOE
(1)	$u = \frac{T_Z}{fd_r} y(x - x_{FOE})$ $v = \frac{T_Z}{fd_r} y(y - y_{FOE})$
(2)	$u = \frac{T_Z}{fd_b} x(x - x_{FOE})$ $v = \frac{T_Z}{fd_b} x(y - y_{FOE})$
(3.a)	$u = \frac{T_Z + T_{Z1}}{d_o}(x - x_{FOE1})$ $v = \frac{T_Z + T_{Z1}}{d_o}(y - y_{FOE1})$
(3.b)	$u = \frac{T_Z}{d_o}(x - x_{FOE2})$ $v = \frac{T_Z}{d_o}(y - x_{FOE2})$

TABLE 2.4: Motion fields of four relevant cases of moving planes

Plane type	Motion module	Motion direction
(1)	$\|w\| = \|\frac{T_Z}{fd_r}\| * \|y\| \sqrt{(x - x_{FOE})^2 + (y - y_{FOE})^2}$	$\frac{u}{v} = \frac{x - x_{FOE}}{y - y_{FOE}}$
(2)	$\|w\| = \|\frac{T_Z}{fd_b}\| * \|x\| \sqrt{(x - x_{FOE})^2 + (y - y_{FOE})^2}$	$\frac{u}{v} = \frac{x - x_{FOE}}{y - y_{FOE}}$
(3.a)	$\|w\| = \|\frac{T_Z + T_{Z1}}{d_o}\| * \sqrt{(x - x_{FOE1})^2 + (y - y_{FOE1})^2}$	$\frac{u}{v} = \frac{x - x_{FOE1}}{y - y_{FOE1}}$
(3.b)	$\|w\| = \|\frac{T_Z}{d_o}\| * \sqrt{(x - x_{FOE2})^2 + (y - y_{FOE2})^2}$	$\frac{u}{v} = \frac{x - x_{FOE2}}{y - y_{FOE2}}$

If we include these FOE equations into the motion equations in Table 2.2, we get the new motion equations that depend with FOE in Table 2.3.

From the above motion field equations, we can deduce the corresponding motion vector direction $\tan\theta = \frac{u}{v}$ and motion vector amplitude $\|w\| = \sqrt{u^2 + v^2}$ as shown in Table 2.4. From the direction equations,only the basic known equation of FOE could be deduced. This equation shows that the FOE is the convergence point of all the velocity vectors. However, the motion amplitude equation, whatever the plane model, is the product of two terms. The first term ($|\frac{T_Z}{fd_r}|$, $|\frac{T_Z}{fd_b}|$, $|\frac{T_Z + T_{Z1}}{d_o}|$ or $|\frac{fT_Z}{d_o}|$) is constant for a given plane that depends only on 3D motions (T_Z, T_{Z1}), camera intrinsic parameter (f) and plane geometric parameter (d). The second term depends only on pixel coordinates (curve variable). If we denote the first constant term as K and the second one as c, we can rewrite the velocity amplitude equation by :

$$\|w\| = \sqrt{u^2 + v^2} = K \cdot c \tag{2.15}$$

We can see that there is a linear relationship between the velocity amplitude $\|w\|$ and the pixel coordinates function c (that we call as "c-value") for pixels that belong to the same plane. In other words, the projection of the pixels of a plane onto the "c-velocity" image [12] is a straight line of slope K. The idea of the "c-velocity" approach is to transform the plane detection in the image into a line detection in the "c-velocity" frame.

2.3.2 Discussion about the different inputs

In oder to determine the "c-velocity" image, we should at first compute the "c-value" c and the motion amplitude $\|w\|$ of each pixel. In this subsection, we will explain how to obtain these parameters.

Motion amplitude In case of perspective projection, the 2D motion field which is the projection of the 3D motion field on the image plane can be computed from Equation **??**. But in our case, the real 3D motion is not known. The only input is an image sequence. Unfortunately, the 2D motion field is just a purely geometric concept and is not directly measurable from the image sequence. However, we can estimate the apparent motion of the brightness pattern that is called optical flow. So, we chose to approximate the 2D motion field by the optical flow and then the 2D motion amplitude by the flow amplitude. In most of the cases, optical flow is a good approximation of the 2D motion field, but not always. For example, the motion field and optical flow of a rotating barber's pole are different, as illustrated in Figure 2.6. According to [66], the optical flow can be considered as the approximation of the motion field under some assumptions :

— Lambertian surfaces
— Point-wise light source at infinity
— No photometric distortion
— Rigid motion

12. *In this image, the y-axis is the "c-value" and the x-axis is the velocity module* $\|w\|$

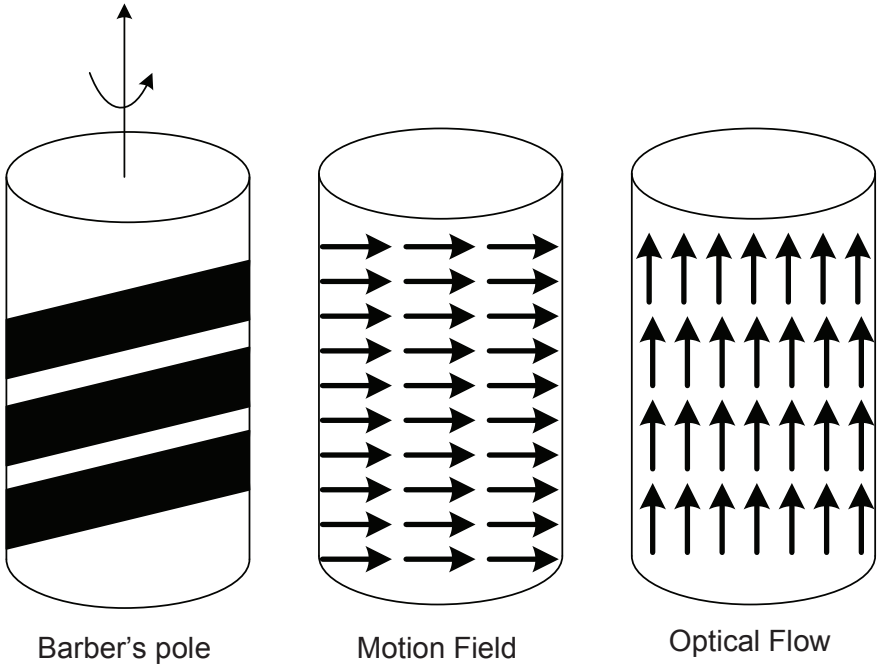


FIGURE 2.6: The motion field and optical flow of a barber's pole

Various approaches are proposed to estimate optical flow such as block-matching, parametric motion approaches and optical flow approaches. But all of them are based on the brightness constancy assumption. That is when a pixel moves, its brightness does not change. But with only this constraint, we can not estimate the two components (u, v) of the flow vector. So, other various assumptions could be added, each of them leads to a different optical flow estimation method. In block matching approaches, image is divided into fixed blocks and the pixels in the same block are supposed to have the same optical flow. So, only a single motion vector is computed in each block. While in parametric motion approaches, the motion vector is induced by the 3D motion of the camera. So, the adjacent pixels may have the same motion parameters. Finally, optical flow approaches are also called "differential approaches" as they are based on partial derivatives of the image signal and/or the sought flow field and higher-order partial derivatives. We can yet subdivide them into local approaches and global approaches.

The most representative method of local approaches is the Lucas-Kanade method [69]. It considers a locally constant optical flow as the additional constraint. Then the problem becomes a linear system and can be solved by any resolution method like gradient descent algorithm.

Many global approaches have been proposed since the famous Horn-Schunck algorithm. All these approaches consider the problem of estimating velocity vectors as the optimization of a global energy function which is a weighted sum of two terms. The first one is

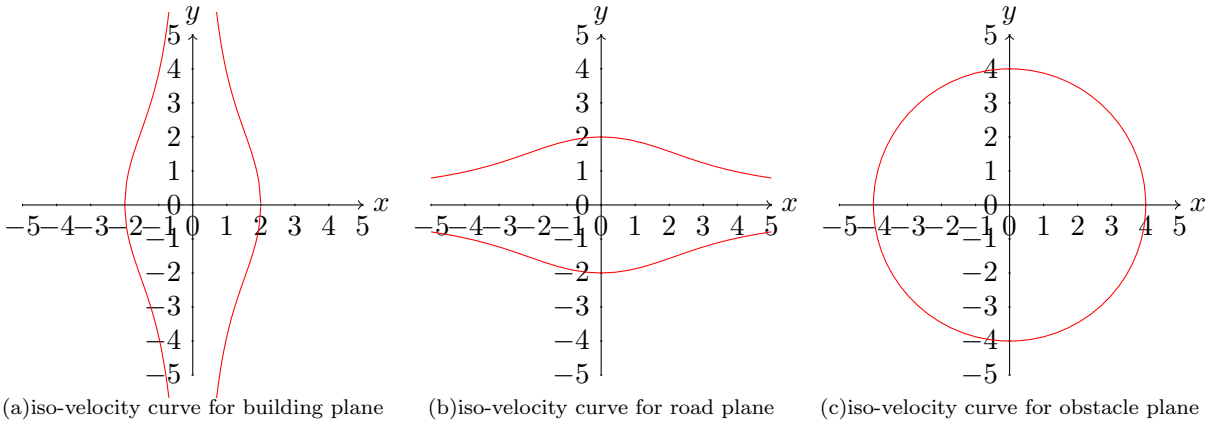
based on the brightness constancy constraint. The second is the regularization term that correspond to the additional constraint. In the Horn-Schunck algorithm, such constraint is the smooth flow assumption.

Since the optical flow is one of the two main inputs of the "c-velocity" method, a detailed presentation of the different estimation methods including evaluation will be given in the next chapter.

"C-value" Besides Optical flow, the other input that is necessary to construct the "c-velocity" image is the "c-value" that is :

$$c = \frac{\|w\|}{K} = \begin{cases} |y|\sqrt{(x - x_{FOE})^2 + (y - y_{FOE})^2} & \text{for road} \\ |x|\sqrt{(x - x_{FOE})^2 + (y - y_{FOE})^2} & \text{for building} \\ \sqrt{(x - x_{FOE_o})^2 + (y - y_{FOE_o})^2} & \text{for obstacles} \end{cases} \tag{2.16}$$

Since K is constant for a given plane, pixels in a plane that have the same "c-value" also have the same velocity. So, the "c-value" is also an iso-velocity curve. The shape of the iso-velocity curve is different depending on the plane model. An example can be found in Figure 2.7 where iso-velocity curves for $c = 4$ are traced with different types of planes knowing the FOE point at the origin of the frame. From these figures, two curves are symmetrically located on both sides of the y axis (for building planes) or the x axis (for road planes). For obstacle planes, these two curves intersect and turn into a circle.

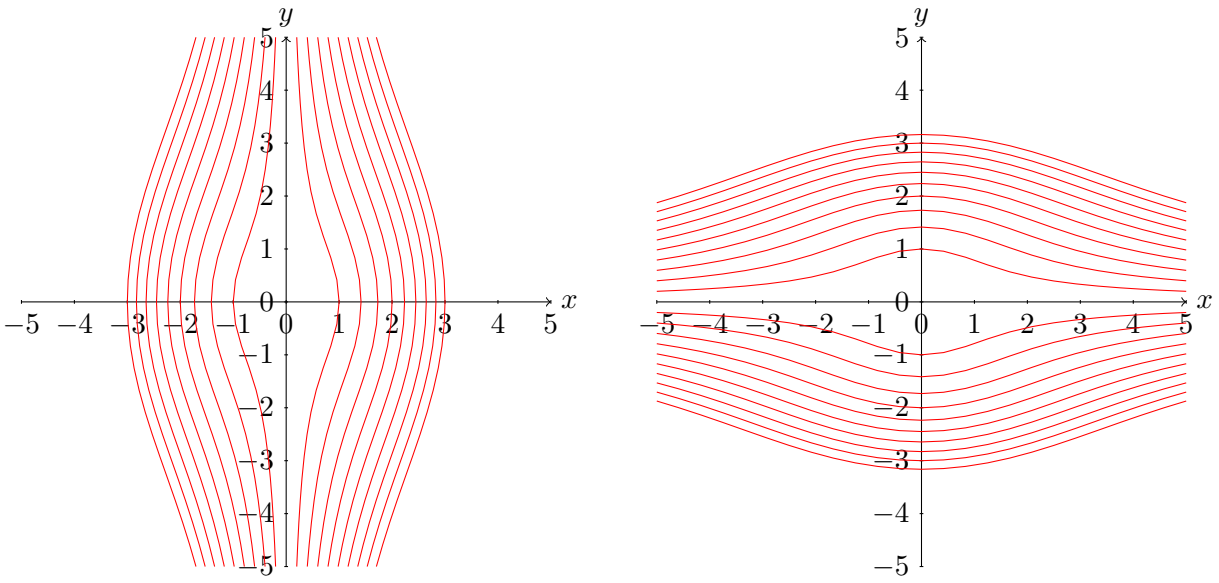


(a)iso-velocity curve for building plane (b)iso-velocity curve for road plane (c)iso-velocity curve for obstacle plane

FIGURE 2.7: Iso-velocity curves for c = 4 and different types of planes considering the FoE at the origin

We can also draw a set of curves with different c-values. For example, a set of curves in the case of road model and building model are shown in Figure 2.8. The interval of "c-value" between two adjacent curve is 1. From these curves, we can conclude that : (1) Iso-velocity curves do not intersect ; (2) Each curve intersects the x axis (for the

road model) or y axis (for the building model) in the image plane within : $x = \pm\sqrt{c}$ or $y = \pm\sqrt{c}$, respectively ; (3) The distance between two adjacent curves is different and depends on the current "c-value". The interval distance is smaller for the higher "c-value".

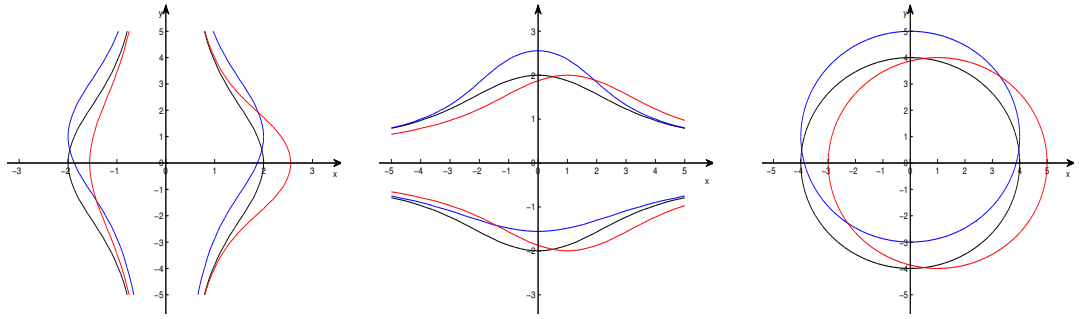


(a)iso-velocity curves for building plane

(a)iso-velocity curves for road plane

FIGURE 2.8: Iso-velocity curves with step 1

In practice, the FOE point is not exactly at the origin. We will discuss now the new iso-velocity curve equations when considering FOE position. For building planes, a change in the FOE position along y axis only leads to a shift along y axis (see the shift from the black curves to the blue curves in Figure 2.9(a)). The curve shape do not change. In contrast, a change of the FOE position along x axis results in a deformation of the curves. We can see that the two curves are pulled toward the movement of FOE position (see the deformation from the black curves to the red curves in Figure 2.9(a)). These changes are completely different for the curves of road planes (see Figure 2.9(b)). In case of the obstacle planes where the curves are circles, a change in the FOE position just changes the circle center.



(a)iso-velocity curve with different FOE position for building plane

(b)iso-velocity curve with different FOE position for road plane

(c)iso-velocity curve with different FOE position for obstacle plane

FIGURE 2.9: Iso-velocity curve with different FOE positions : FOE(0, 0) for black curves, FOE(0, 1) for blue curves, FOE(1, 0) for red curves

The estimation of FOE position is an important step to compute the iso-velocity curves. In many studies, optical flow field is usually used to locate the FOE position. A summary on the state of the art methods will be given in Chapter 4, and we also propose three different estimation methods to locate the FOE point.

Once the FOE position is known, the c-values are computed from Equation 2.16 for each pixel. For a standard image size of 240×320, if we consider the FOE position in the image center, the c-values vary from 0 to 96000 for the road model and vary from 0 to 128000 for the building model. Such large variation is not suitable for implementation purposes (computing accuracy) and homogeneity ($c \approx length^2$), so we translate the models into the relations between $\|w\|$ and $\sqrt{c}$. Then the linear relationship between $\|w\|$ and c becomes a parabolic relationship between $\|w\|$ and $\sqrt{c}$.

$$\|w\| = K * c = K * \sqrt{c}^2 \tag{2.17}$$

2.3.3 Construction of the "c-velocity" images

From the above presentation, objects (buildings, road, obstacles) are approximated by planes of different orientations. The projection of pixels from the same plane (object) on the corresponding[13] "c-velocity" image is a straight line. Using this property, the detection of objects can be transformed into the extraction of lines in the "c-velocity" image. This subsection describes how to construct the different "c-velocity" images. The construction process is actually a pixel voting process. Each pixel will give a vote in the three "c-velocity" images. The votes from a pixel will help the emergence of a line only

13. *The "c-values" are different for different plane types*

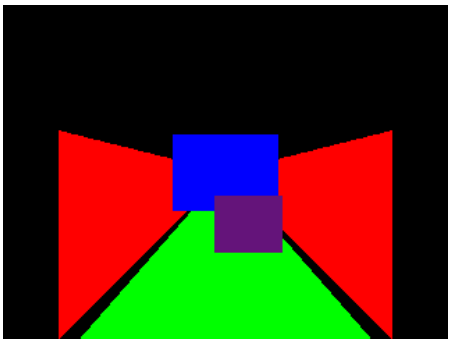
in the correct "c-velocity" image. In the non-correct "c-velocity" spaces, these votes are just considered as background noises.

We will give below a toy example to explain how to construct the "c-velocity" images and also show the resulting images. Figure 2.10(a) is an urban road scene that is composed with different objects. The green plane is the road. On this road, two obstacles moves with different motions : the violet plane is the crossing obstacle and the blue plane is an approaching obstacle. On both sides of the road, two buildings are displayed as red planes. The geometric parameters of these objects, the camera's intrinsic and extrinsic parameters are defined a priori. So, the 2D motion vector of each pixel is computed directly from Equation 2.12. The FOE position is also directly computable from Equation 2.13. Then three "c-values" (c_b, c_r, c_o) are obtained from Equation 2.16. So each pixel has four values ($\|w\|$, c_b, c_r, c_o). Then, the pixel will vote for the different "c-velocity" images by incrementing the value of point ($\|w\|$, c_i [14]). The detailed algorithm of this cumulative voting process can be found in Algorithm 1. After the voting process, we obtain three "c-velocity" images shown in Fig. 2.10(c)(d)(e) : Two buildings (red planes in Figure 2.10(a)) are projected as two lines in the building "c-velocity" image ; the votes from the road pixels (green plane in Figure 2.10(a)) becomes a line in the road "c-velocity" image. The obstacle lines are not visible in the obstacle "c-velocity" image because they are submerged by the votes of the building pixels and the road pixels. To further know this, we can look at the corresponding Hough histogram in Figure 2.10(h) : the approaching obstacle (the blue rectangle in Figure 2.10(k)) is presented as a small peak (peak ① in Figure 2.10(h)). The crossing obstacle is hidden in peak ③ of Figure 2.10(h)(see Figure 2.10(y)). Both obstacle peaks are submerged by the peak (peak ② in Figure 2.10(h)) voted by the building and road pixels(see Figure 2.10(n)).But we can still extract them by our iterative histogram splitting method presented in [70].

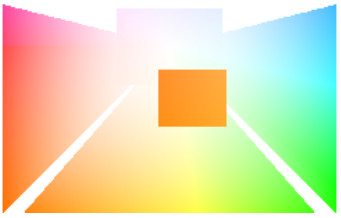
2.3.4 Post-processing of the "c-velocity" image for object detection

Once the 3D planes are transformed into lines in different "c-velocity" images, the next step is to extract these lines from the "c-velocity" images. Line extraction can be performed using Hough transform. As these lines are crossing the origin, a 1D hough transform is needed to exhibit the histogram of the line slope (see the histograms in Figure 2.10(f)(g)(h)). K-means or other clustering methods could be applied to detect peaks in the histograms. In fact, it is not easy to correctly extract the lines. Many factors will influence the quality of extraction. For instance, motion imprecision or bad FOE position will affect the projection of the 3D planes on the "c-velocity" images. For example,

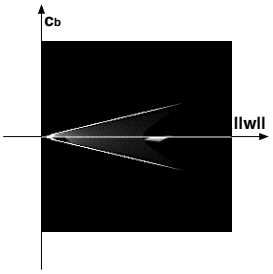
14. $c_i = c_b$ *for the building "c-velocity" image ;* $c_i = c_r$ *for the road "c-velocity" image ;* $c_i = c_o$ *for the obstacle "c-velocity" image*



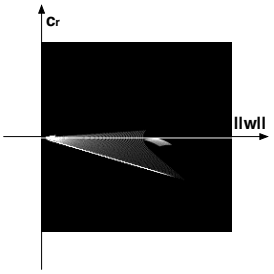
(a) synthetical image scene (red : buildings ; green : road ; blue : approaching obstacle ; violet : crossing obstacle)



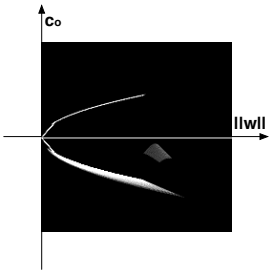
(b) 2D motion field of the image



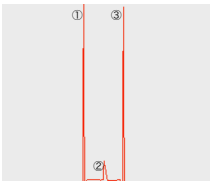
(c) Building "c-velocity" image



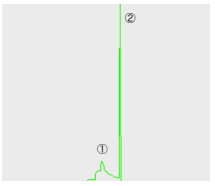
(d) Road "c-velocity" image



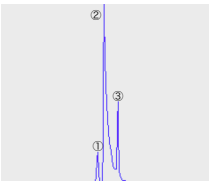
(e) Obstacle "c-velocity" image



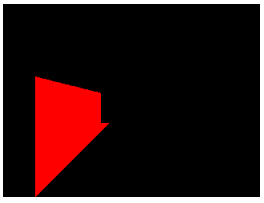
(f) 1D Hough histogram for the building "c-velocity" image



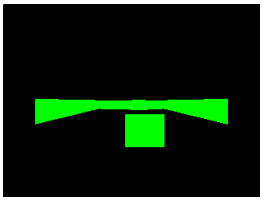
(g) 1D Hough histogram for the road "c-velocity" image



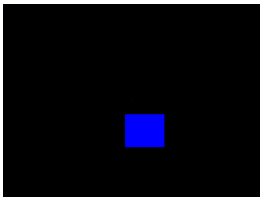
(h) 1D Hough histogram for the obstacle "c-velocity" image



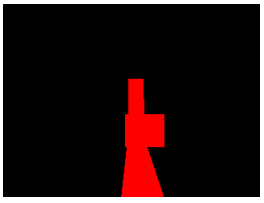
(i) Pixels corresponding to peak ① in building histogram



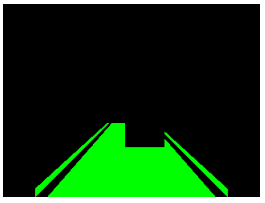
(j) Pixels corresponding to peak ① in road histogram



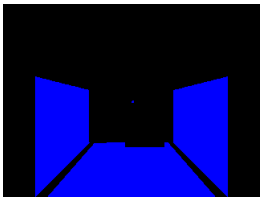
(k) Pixels corresponding to peak ① in obstacle histogram



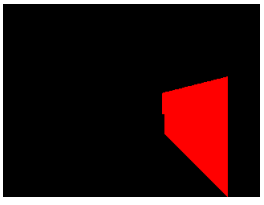
(l) Pixels corresponding to peak ② in building histogram



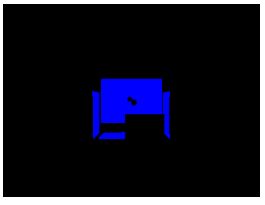
(m) Pixels corresponding to peak ② in road histogram



(n) Pixels corresponding to peak ② in obstacle histogram



(x) Pixels corresponding to peak ③ in building histogram



(y) Pixels corresponding to peak ③ in obstacle histogram

FIGURE 2.10: An synthetic example for the construction of the "c-velocity" images and the corresponding Hough histograms for line extraction

Input: optical flow (u, v), FOE (x_{foe}, y_{foe}) and image
Output: Building_Space ($\|w\|$, c_b), Road_Space ($\|w\|$, c_r), Obstacle_Space ($\|w\|$, c_o)

begin
  Initialize Building_Space ($\|w\|$, c_b) ;
  Initialize Road_Space ($\|w\|$, c_r) ;
  Initialize Obstacle_Space ($\|w\|$, c_o) ;
  for *Each pixel* $(x_0, y_0) \in$ *image* **do**
    $\|w_0\| = \sqrt{u(x_0, y_0)^2 + v(x_0, y_0)^2}$;
    $c_{b0} = |x|\sqrt{(x_0 - x_{foe})^2 + (y_0 - y_{foe})^2}$;
    $c_{r0} = |y|\sqrt{(x_0 - x_{foe})^2 + (y_0 - y_{foe})^2}$;
    $c_{o0} = \sqrt{(x_0 - x_{foe})^2 + (y_0 - y_{foe})^2}$;
    Increment ($\|w_0\|$, c_{b0}) cell in Building_Space ;
    Increment ($\|w_0\|$, c_{r0}) cell in Road_Space ;
    Increment ($\|w_0\|$, c_{o0}) cell in Obstacle_Space ;
  end
end

Algorithm 1: C-velocity voting process

the "line" width maybe very large or even the representation is no longer a "line". Besides these influences, the inter-perturbation between the different spaces also makes the detection very difficult. From the toy example in Figure 2.10, there is no error in the motion and the FOE position estimation as they are computed directly from the equations. But we still can not extract the obstacle "lines" from the obstacle "c-velocity" image because the number of pixels for an obstacle (vehicle or pedestrians) is smaller compared with that of a building or a road. So the votes from the obstacles are not significant compared with the votes of other large size objects. In fact, any "c-velocity" image suffers from the perturbation of other objects from other models.

2.3.5 Optimizations

In order to use the "c-velocity" approach in ADAS applications for object detection, we should overcome the real-time constraint. So, in this subsection, we propose some optimizations of our method in different ways : algorithmic modification and implementation using multithread.

Direct transform without constructing c-velocity images The "c-velocity" approach consists of two-steps (see Figure 2.11) : a "c-velocity" frame construction then a line detection using Hough transform.

Since the construction of "c-velocity" frames is just an intermediary step (see Figure 2.11), we show that it is possible to build directly the Hough histograms after computing

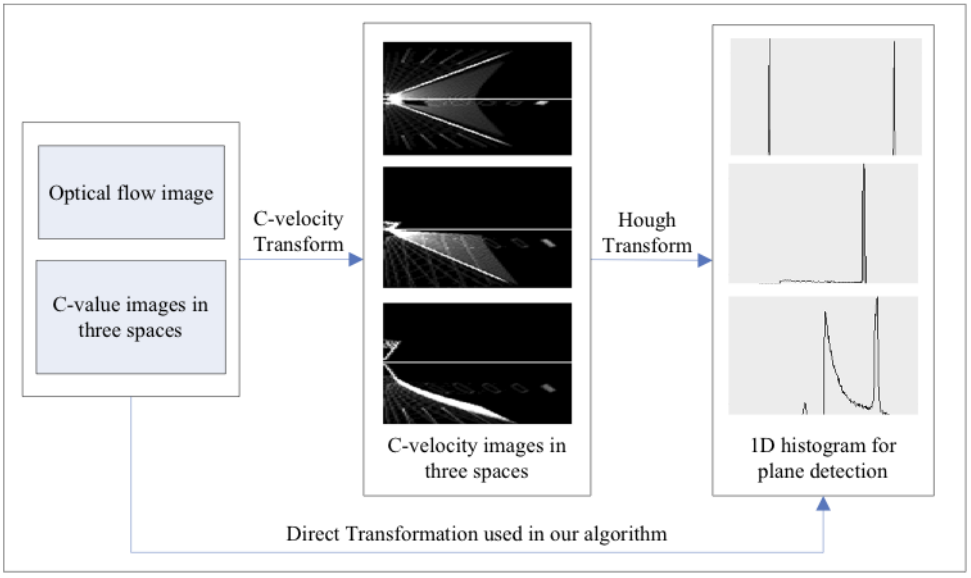


FIGURE 2.11: c-velocity algorithm flow

the optical flow and determining "c-values". On the one hand, this modification reduces the computation time by simplifying the detection process, and on the other hand avoids a double quantification.

The main difficulty of this direct method is to correctly count the vote number and avoid redundant votes. Let us consider the following example to reveal this problem. In Figure 2.12, three pixels vote for the same cell $(c_0, \|w_0\|)$ in the "c-velocity" image. So three votes are added to the cell $(c_0, \|w_0\|)$. But when we build the line slope histogram, the cell $(c_0, \|w_0\|)$ give only one vote to $K = \frac{\|w_0\|}{c_0}$. This means that the pixel numbers is not equal to vote numbers in the histogram. It is possible that several pixels may give only one vote in the histogram. The question is : how to know the vote numbers without the "c-velocity" images ? The solution is to group the pixels with the same motion module $\|w\|$ and the same "c-value" c before building the histogram. All the pixels in the group (c, w) give only one vote for $K = \frac{\|w\|}{c}$ in the histogram. With a fixed $\|w\|$, $c \to K$ is bijective. So, we can also use the $(\|w\|, K)$ couple to group pixels. Algorithm 2 is an example of the direct method specifically for building planes.

From the original "c-velocity" approach, the construction of the "c-velocity" images needs to quantify $\|w\|$ and c. The slope K is computed by the digitized value of $\|w\|$ and c and then another digitization to build the line slope histogram. However, in the direct transform, the computation of K use the original $\|w\|$ and c. Then K is digitized only for building 1D histogram. So the voting results seems more correct using the direct transform.

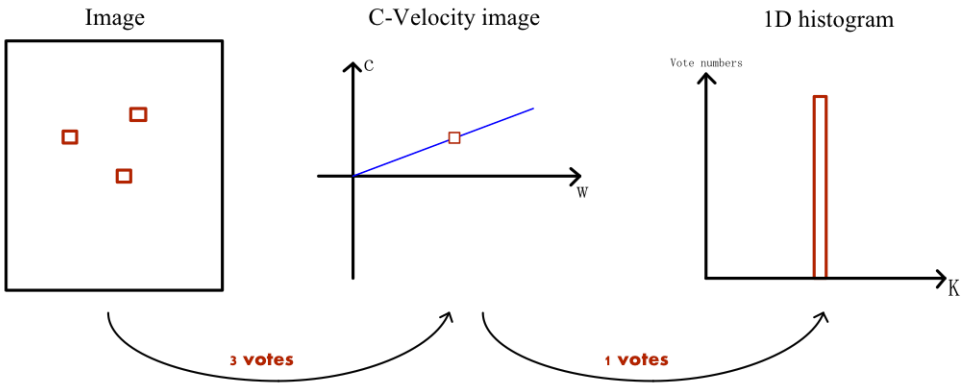


FIGURE 2.12: Two voting processes in the original "c-velocity" approach

Input: optical flow (μ, ν), FOE (x_{FOE}, y_{FOE}) and image
Output: Building histogram h_b

begin
  Initialize Building histogram k_b ;
  for *Each pixel* $(x_0, y_0) \in$ *image* **do**
    $\|w_0\| = \sqrt{u(x_0, y_0)^2 + v(x_0, y_0)^2}$;
    $c_{b0} = |x|\sqrt{(x_0 - x_{foe})^2 + (y_0 - y_{foe})^2}$;
    $K_{b0} = \frac{\|w_0\|}{c_{b0}}$;
  end
  Save the pixel information (x, y, w, K_b) in an 1D table *Tab* ;
  Sort *Tab* by the order of w and K;
  Group the pixels with the same w and K ;
  for $i = 0$ **to** *size of Tab* **do**
    if *Tab[i].PixelNumbers* $>$ *threshold* **then**
      $h_b[Tab[i].K_b] + +$;
    end
  end
end

Algorithm 2: Direct *c-velocity* process

Another advantage is the computational acceleration. To illustrate this result, we run these two algorithms in the same laptop of Mac OS with an Intel i7 Core 2. CPU and compare their execution time. The image size is 640×480. The computation time is measured using two methods. *T(s)* measures the time in second by function *mach_absolute_time*, which is a CPU/Bus dependent function that returns a value based on the number of "ticks" since the system started up. The number of cycles is obtained by reading directly the value of the register Time Stamp Counter (TSC), which counts the number of cycles after reset. We can see from Table 2.5 that these two numbers are coherent with the CPU frequency. We can conclude from the results, the direct "c-velocity" method is at

TABLE 2.5: computation speed test for two algorithm of *c-velocity* process

Algorithms / Times	Old algorithm		Modified algorithm	
	T(s)	number of cycles	T(s)	number of cycles
1	0.290146	781607562	0.118234	318497650
2	0.277248	746862387	0.111056	299164911
3	0.313165	843619656	0.112126	302044550
4	0.287641	774859146	0.112892	304109682
5	0.282248	760330440	0.112202	302252325
6	0.277852	748491420	0.110819	298525383
7	0.289525	779933592	0.110547	297791556
8	0.277402	747277443	0.111624	300693888
9	0.288778	777923826	0.110522	297726720
10	0.305127	821964654	0.112925	304197298
mean	0.2889132	778285013	0.1122947	302500396

TABLE 2.6: computation speed test for multi-thread algorithm of *c-velocity* process

	T(s)	number of cycles
1	0.0753531	202985970
2	0.0777929	209558034
3	0.0758184	204239355
4	0.0747896	201467637
5	0.0757412	204031359
6	0.0740406	199450257
7	0.0738521	198938229
8	0.072257	194647038
9	0.0751435	202421949
10	0.0768012	206888535
mean	0.07515896	202462800

least twice faster than the original one.

Multi-thread implementation Since the construction of the voting spaces ("c-velocity" images and 1D histograms) is necessary for each plane type. It is essential to implement the multi-thread to accelerate the computation. Here, we use the boost thread library to create a thread for each voting spaces. Therefore, there are on the whole three threads. And the code is executed on the laptop with an Intel Core i7 of 2.7 GHz. Table 2.6 shows the computational time result when a multi-thread implementation is applied to the "c-velocity" process. We can compare these results with those in Table 2.5 as they are tested on the same laptop with the same images. Although the voting process for the three different orientations of plane is designed by three separate threads, the sharing data makes the computational time not too much accelerated.

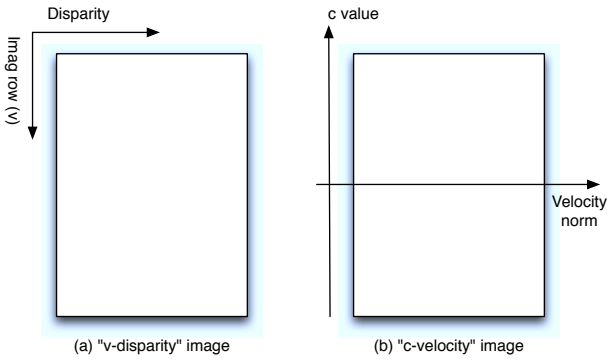
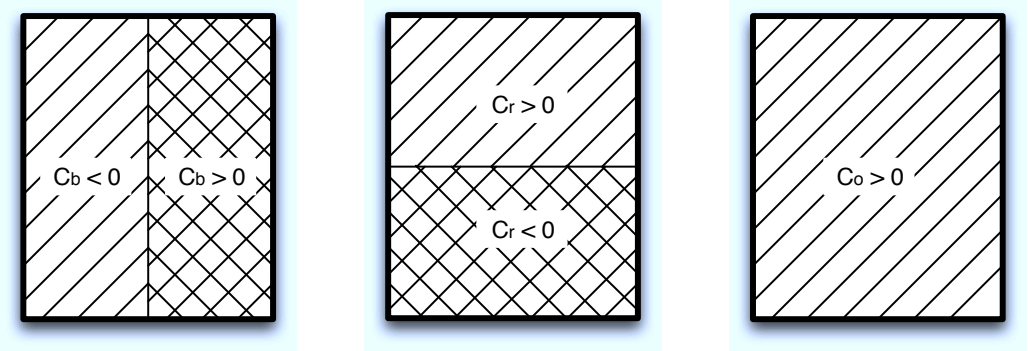


FIGURE 2.13: Comparison of "c-velocity" image and "v-disparity" image

2.3.6 Comparison between "c-velocity" and "v-disparity"

By studying the "v-disparity" and the "c-velocity" approaches, we showed that both methods share a common principle : facilitating object detection by projecting them into a new 2D space where the representations of objects are simple forms (ex. lines) that are easily detectable. The comparison between the two cumulative spaces is straightforward : in the "v-disparity" image, the x axis is the disparity and the y axis is the image row. The origin is located on the top left of the image (see Figure 2.13(a)) ; while in the "c-velocity" image, the x axis is the velocity norm and the y axis is the "c-value" (see Figure 2.13(b)). Each plane type has a specific "c-value" equation (see Equation 2.16). All the "c-values" are positive but we factiously give them a sign according to their position in order to distinguish pixels from different planes that have same "c-value" (see Figure 2.14). So, the origin of the "c-velocity" image is on the left middle of the image. If we add the sign to the "c-values", the line will also have a sign to indicate the position of the plane in the image. An example of "c-velocity" image is shown in Figure 2.15. In this "c-velocity" image, two lines correspond to two building planes that are respectively located on each side of the road since their distances to the image center are the same, the sign of the "c-value" (or slope K) can distinguish them in the "c-velocity" image (or in the 1D histogram).

When computing the disparity map in "v-disparity" approach, the search of the correspondent point is performed along the epipolar line. All epipolar lines converge on a unique point (epipolar point). After camera calibration and rectification, these epipolar lines become horizontal and parallel. It means that disparity estimation is simplified into a 1D search along an horizontal line. In contrast, in the case of "c-velocity" approach, no simplification could be done for motion that is two-dimensional by nature except when it is purely translational (see Figure 2.16). In this case, all corresponding epipolar lines radiate from a single epipole, that is FOE. So FOE position should be known at first in order to simply the two dimensional estimation to a 1D search along epipolar lines.



(a) Sign of the "c-value" for building planes

(b) Sign of the "c-value" for road planes

(c) Sign of the "c-value" for obstacle planes

FIGURE 2.14: Distribution of the sign of the "c-value" in the image for different plane types

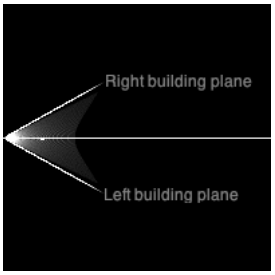


FIGURE 2.15: An example of "c-velocity" image for the building espace

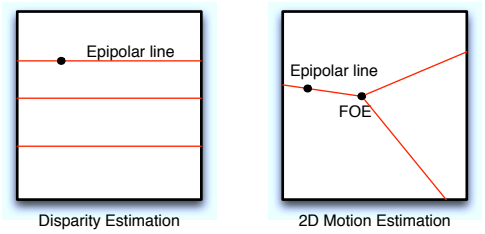


FIGURE 2.16: Epipolar lines for disparity estimation and 2D motion estimation

If we consider that "disparity" is similar in a certain sens to "velocity" norm, then the image row "v" corresponds to the iso-velocity curve "c". Both depends on pixel coordinates, "v" is a line index while "c" is a curve index (see Figure 2.17). And the latter also depends on the FOE position.

In ADAS applications, cooperation between stereovision and motion is largely used for object detection. Generally, objects are detected by the stereo system, and then confirmed by exploiting motion. Such kind of cooperation could be considered for "v-disparity" and "c-velocity". The "c-velocity" approach can on the one hand give motion information and on the other hand enhance decisions for object detection. While the combination of "v-disparity" and "c-velocity" is not straightforward because it is not

(a) Image raw (line index) for the "v-disparity" (b) "C-value" (iso-velocity curve) for the "c-velocity"

FIGURE 2.17: Comparison of the image row "v" with the iso-velocity curve "c"

easy to find the bijection between the "v-disparity" image and the "c-velocity" image. To facilitate such integration, we propose to unify the writings by modifying our "c-velocity" image to the "v-velocity" image.

2.3.7 From "c-velocity" to "v-velocity"

In the "c-velocity" approach, two assumptions are defined : (1) the motion is purely translational ; (2) objects are approximated into planes with specifical orientations. For instance, the road plane is modeled as a horizontal plane. At the same time we consider that the camera is horizontally mounted on the camera. So we always define that the camera axis is parallel with the road plane. The motion equations in the "c-velocity" approach is based on this assumption. But the parallel relation between the road surface and the camera orientation is too strict. If we define a small angle θ between the road plane and the camera axis, we can rewrite the motion equations of the road plane by

$$\begin{cases} u_r = \frac{T_Z}{fd}(x - x_{FOE})(y\cos\theta + f\sin\theta) \\ v_r = \frac{T_Z}{fd}(y - y_{FOE})(y\cos\theta + f\sin\theta) \end{cases} \tag{2.18}$$

Then the motion module can be expressed by

$$\|w\| = |\frac{T_Z}{fd}||y\cos\theta + f\sin\theta|\sqrt{(x - x_{FOE})^2 + (y - y_{FOE})^2} \tag{2.19}$$

From the motion module equation in Equation 2.19, the "c-value" becomes more complex and difficult to compute as θ is unknown. So the definition of the iso-velocity curves and the construction of the "c-velocity" image requires to know θ. But how about we regard the different motion components u_r and v_r separately ? From the expression of v_r in Equation 2.18, we can remark that it is a second order equation of y (image row) and that it can be expressed as the product of two terms that depend only on the first order of y :

$$v_r = \mathbf{a_1}(y - \mathbf{a_2})(y - \mathbf{a_3}) \tag{2.20}$$

with

$$\begin{cases} \mathbf{a_1} = \frac{T_Z}{fd} \cos\theta \\ \mathbf{a_2} = y_{FOE} \\ \mathbf{a_3} = -f \tan\theta \end{cases}$$

From the above equation, if we construct a "v-velocity" image [15], the road plane is represented by a second order curve. Such curve can be extracted using 3D Hough Transform by estimating three parameters $\mathbf{a_1}$, $\mathbf{a_2}$ and $\mathbf{a_3}$. This "v-velocity" approach is more robust than the original "c-velocity" approach as the angle between the road surface and the camera orientation is taken into account. In addition, for practical considerations, "v-velocity" and "v-disparity" images could be combined for road detection as they have the same y-axis.

We will now use a toy example to show the construction of the "v-velocity" image and then compare it with the "v-disparity" image. This is the same example that we used before for the construction of the "v-disparity" image (see Figure 2.3). The 3D scene (see the left image in Figure 2.18) is generated by the SiVIC simulator. The disparity map is computed directly from depth and from Equation 2.6. The motion field is also computed directly using 3D motions and depth information from Equation 2.1. After projecting all the pixels of the scene into the "v-disparity" image and the "v-velocity" image, we get the image in the middle of Figure 2.18 ("v-velocity" image) and the right image of Figure 2.18. They have the same y axis - image row. In the "v-velocity" image, we can find the projection of the road plane - a second order curve. Since the velocity value v_r can be negative, we draw the zero velocity line $velocity = 0$ (see the red line in the middle image in Figure 2.18). This line intersects the road curve at point ($velocity = 0$, $y = y_{FOE}$). If we draw an horizontal line along this intersection point (see the yellow line in Figure 2.18), we can find the horizon line in the scene. And this horizontal line intersects the road line in the "v-disparity" image at point ($disparity = 0$, $y = y_{or}$). We can see that y_{or} is an important parameter to compute the angle between the optical axis of the cameras and the horizontal (see Equation 2.10).

In the "v-disparity" image, the projection of the road plane - "line" - can be easily detected thanks to Hough Transform. We express the line equation by its definition in the polar coordinate system :

15. *"v" is the image row, here means y ; "velocity" is the optical flow in y direction v_r*

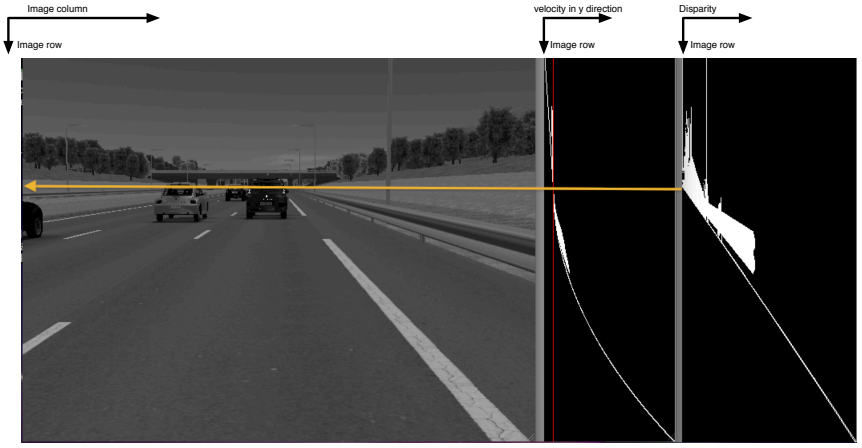


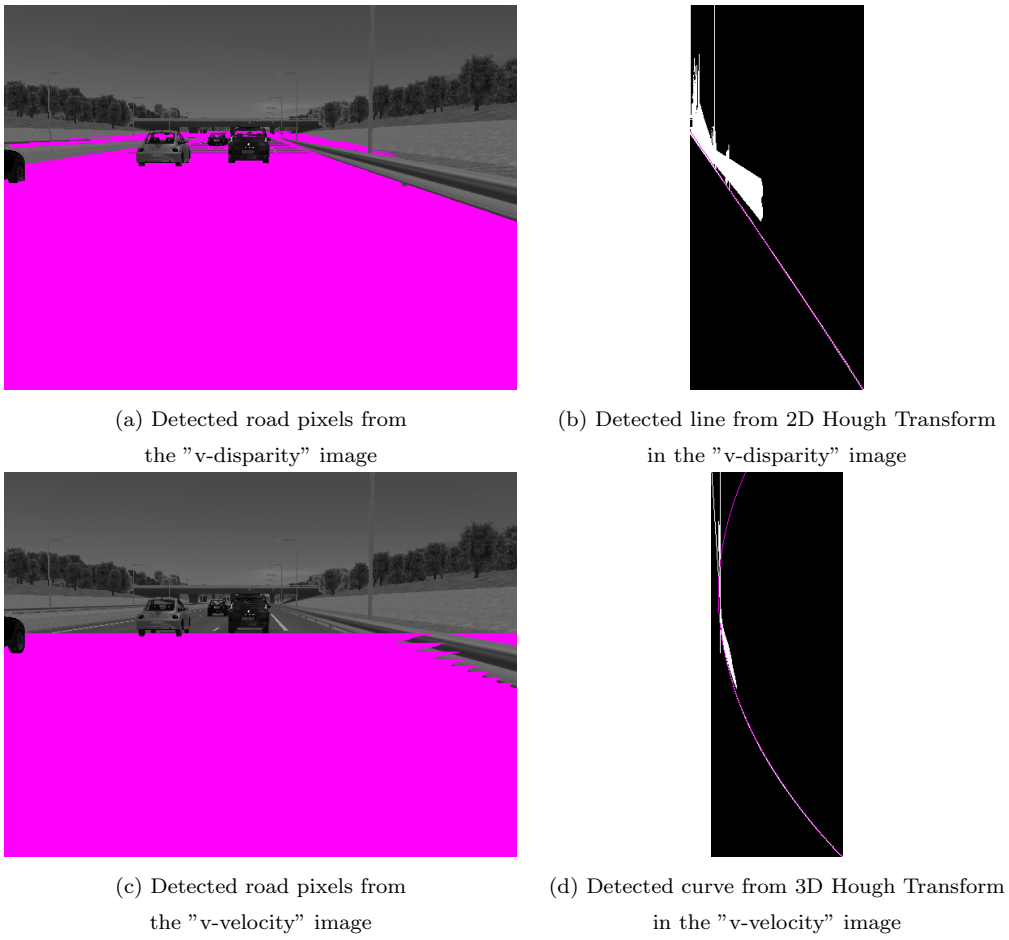
FIGURE 2.18: "V-velocity" image and "v-disparity" image comparison

$$x\cos\theta + y\sin\theta = r \tag{2.21}$$

In Figure 2.19, the road "line" is detected (see the violet line in Figure 2.19(b)) from Hough Transform using (θ, r) as parameters. Corresponding road pixels are extracted from the image (see the violet regions in Figure 2.19(a)).

The projection of the road in the "v-velocity" image - that is a second order curve - can be also detected using Hough Transform. However, three parameters (a_1, a_2, a_3) are needed to represent a second order curve (see Equation 2.20). It is possible to define a 3D hough transform to estimate the three parameters but this possibility is time-consuming and takes a lot of memory spaces. We show an example of detection in Figure 2.19(d) and the violet curve is the detected Hough Transform. The corresponding pixels are shown in Figure 2.19(c) as violet regions. We should mention that the detection results depend on the quantification of the three parameters. Detection results are better when we reduce the quantification interval for the different parameters (especially for a_1). But this largely increases the required memory space. Moreover, this 3D Hough Transform is not really adapted for real images since optical flow and FOE estimation will introduce a lot of noise in the 3D space. So, the detection results are given only for synthetic images (see Figure 2.19).

From the above presentation, a horizontal plane (or quasi-horizontal plane) such as road surface can be extracted from the "v-velocity" approach by using the velocity information in y direction v_r. If we consider a vertical plane with plane equation $X\cos\theta + Z\sin\theta = d$, we can rewrite the velocity equations by



(a) Detected road pixels from the "v-disparity" image

(b) Detected line from 2D Hough Transform in the "v-disparity" image

(c) Detected road pixels from the "v-velocity" image

(d) Detected curve from 3D Hough Transform in the "v-velocity" image

FIGURE 2.19: Road detections from "v-disparity" and "v-velocity" images

$$\begin{cases} u = \frac{T_Z}{fd}(x - x_{FOE})(x\cos\theta + f\sin\theta) \\ v = \frac{T_Z}{fd}(y - y_{FOE})(x\cos\theta + f\sin\theta) \end{cases} \tag{2.22}$$

In this case, we can find that the motion in x axis is the product of two terms that depend only on the first order of x (image column). This means that if we construct a "u-velocity" image [16], the vertical plane becomes a second order curve in the "u-velocity" image. This image can be integrated with the "u-disparity" approach to exploit vertical planes such as buildings ($\theta \approx 0$) and obstacles ($\theta \approx \frac{\pi}{2}$).

2.4 Conclusion

In this chapter, we have presented the state of the art concerning vision-based obstacle detection methods for the ADAS. In this kind of applications, obstacles are vehicles or pedestrians that move on the road, most of these methods start from locating the road

16. *the x axis is the image column x and the y axis is the velocity component u*

and then detecting obstacles on the resulting road region. Among all the approaches, we have studied the "v-disparity" approach. This method exploit the so-called "v-disparity" frame by projecting all pixels in this frame. The road plane is projected onto a slanted line in this new space and obstacles (frontal vertical planes) becomes vertical lines that intersect the road line. Such method is efficient thanks to its cumulative nature and because it transforms a complex plane detection onto a simple line detection. It is then natural to try to adapt it to monocular vision. It was the main motivation of the "c-velocity" approach. This method assumes that objects (buildings, road, obstacles) in urban road scenes could be approximated by planes of different orientations (horizontal, lateral vertical and frontal vertical). These planes are projected on the corresponding "c-velocity" frame and their projections become easily detectable lines. In this chapter, we have also presented how to construct the "c-velocity" frames for different plane orientations and how to detect lines by Hough Transform. For real-time considerations, we also made some optimizations to accelerate the computation, including algorithmic optimization and implementation of multi-threads. Finally, we try to modify the "c-velocity" approach and define the "v-velocity" frame that could be easily combined with the "v-disparity" image to enhance object detection.

3

Model-based 2D motion estimation as a preliminary step for c-velocity

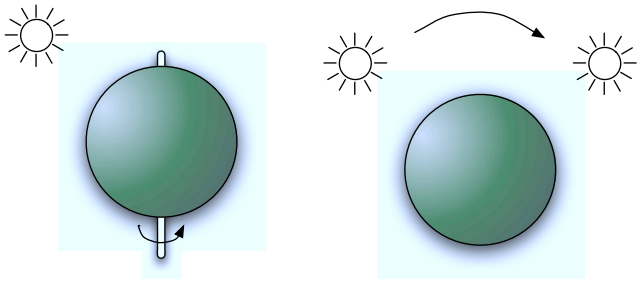
Contents

3.1 Introduction

When we use a camera to record the environment, the whole 3D scene is projected into an image plane. While computer vision is an inverse process, different techniques are designed to reconstruct explicit, meaningful descriptions of the structures and properties of the 3D world from image sequences. Recovering the whole environment with insufficient information from the image sequences is a really rough task. So we should exploit as much as possible information from 2D images. From one image, only the color or intensity information and the texture information are useful. When we compare the difference between the images, the displacements are exploitable. These displacements are called "disparities" if the images are captured from different viewpoints (eg. stereo). In such case, the displacements can be very large and be estimated by matching strategies. And then the 3D geometry and the camera pose are estimated simultaneously from the disparity information. This is the so-called "Structure from Motion [71]". Another type of displacement is the apparent motion from two successive images captured by a monocular camera. This is called the "optical flow". The estimation of optical flow is, for instance, used for motion compensation in video compression. It is also part of researches in robotics and in many domains such as object detection and tracking, robot navigation, egomotion recovery, etc. In the frameworks of ADAS, these apparent motions are due to moving objects as well as camera motion. So, we can exploit this optical flow to help obstacle detection and intelligent navigation. In our case of object detection from the "c-velocity" approach, an accurate motion estimation is important to construct an exact "c-velocity" image. So, this chapter will present different estimation strategies and also show different benchmarks to evaluate the motion estimation results. At the same time, we propose a new model-based optical flow estimation method that is specially designed for motion detection with large displacement and homogeneous regions. This method is very suitable for the road optical flow estimation in ADAS applications.

3.2 Motion representation and models

Motion has an unambiguous definition in three dimensional world. However, when the 3D motion is projected onto the image planes, the motion definition is divided into two concepts : optical flow and 2D motion field. Optical flow is considered as apparent motion of the brightness pattern in the image. This means that it describes the spatio-temporal variations of pixel intensities. While the 2D motion field is the projection of the 3D motion onto the image plane. From the next discussions, we will explain their difference as in Figure 3.1. In the left case, a perfectly uniform sphere is rotating in front of the camera. Since there is no brightness change, the optical flow is zero, but the motion field



(a) (b)

FIGURE 3.1: Difference between optical flow and 2D motion field

is not zero and follows the surface points. In the right image, the sphere is static but the light source is moving from left to right. The motion field in this case is zero but the optical flow is not zero because of the shade change. In both situations, the motion field is totally different from the optical flow. However, these cases are rare. Generally, optical flow is an approximation of the motion field. And motion field is only a geometric concept since optical flow can be estimated directly from the images. So motion field is generally approximated by optical flow and we will consider that motion estimation is equivalent to optical flow estimation.

3.2.1 An introduction about optical flow

As explained before, spatial changes of the cameras lead to disparities between two images while temporal changes (sometimes spatial changes also) of the cameras result in optical flow. Techniques that exploit disparity information are defined as stereoscopy. In these techniques, two cameras are set up to capture the scene simultaneously. The first step is the calibration that allows to determine the geometric configurations between two cameras. Then, the images are rectified using intrinsic and extrinsic parameters of the camera that are obtained from the calibration step. Then, feature matching is performed to find similar primitives from two images. Thanks to image rectification, finding similar primitives between two images is performed in 1D only along horizontal lines (epipoles). Finally, depth information can be exploited from the disparity map. Stereovision is a widely used technique for many applications thanks to its reliability. Unfortunately, the matching method that is used in stereovision is difficult to transpose to estimate optical flow. One obvious reason is that we can not rectify and calibrate cameras in case of motion to simplify matching since the amplitude of motion is not known, the matching must be performed in 2D which largely increases the search complexity. Moreover, robust matching is generally performed using sparse primitives, if we transpose it to motion, we get only a sparse optical flow. In our case, we chose to exploit differential methods to estimate a dense optical flow. Since, disparity information can be used to estimate

depth map, and since optical flow contains motion information, if we combine them, we can directly compute the 3D motion of each point. This combination will lead to the so-called "scene flow". Recently, many researchers estimate the scene flow from the stereo image sequences [72] [73] [51] [74] [75].

From Chapter 2, the plane-model-based object detection method - "c-velocity" - exploits image sequences from an on-board moving camera in the context of ADAS. This method exploits optical flow to construct the "c-velocity" image in order to simplify object representation and detection. So, the quality of optical flow estimation will directly affect the construction of the "c-velocity" image. In the rest of this chapter, we will focus on optical flow estimation and analyze different estimation methods.

The concept of optical flow was first introduced by the psychologist J.J. Gibson. In [76], he first brought forward the idea of deriving 3D motion and shape of the scene from the measurements of feature displacement on the retina, which he defined as "optical flow". A more official definition about the optical flow is : "The motion of the brightness pattern in the image plane which arises from the relative motion between the objects and the camera" is called optical flow. In other words, optical flow is the apparent motion between two consecutive frames.

According to J.J Gibson, the optical flow was defined at first to recover the scene structure and 3D motion. Many researchers provided a mathematical framework for this idea. The detail works to realize this idea are specified into two steps by [71]. In the first step, the accurate image displacements between two consecutive frames are computed. Then the 3D motion and the structure of the scene are computed from the equations relating the 2D image velocity to the 3D parameters. Besides of this "structure from motion" application, the optical flow is also applied for interpolation in the video compression field. Recently, the optical flow is largely opted as an important information for the automatic navigation such as obstacle detection, focus of expansion calculations and time-to-contact information.

3.2.2 Motion models

Motion estimation methods are usually based on motion models. For example, to estimate the apparent motion of the brightness pattern (optical flow), we should use the brightness constancy model. The motion that arise in the image sequences is due to the 3D motions of the objects and the camera. Although the motion of each pixel is different to others but they may be arise from the same 3D motion parameters if these pixels are from the same object. We can then design parametric motion models to describe these properties.

3.2.2.1 Brightness constancy model

As explained before, optical flow is the motion of the brightness pattern in the image plane, therefore for each pixel, if we could find how it moves from one image to another, we know the optical flow. To find the correspondence between images, we consider the brightness constancy assumption. That means the pixel intensity or color does not change in the different images when the pixel moves. Such assumption can be mathematically presented in two equations. Either the total derivative of the image brightness I with respect to time t is zero (see Equation 3.1), or the brightness value in the first image is equal to the one in the second image (see Equation 3.2).

$$\frac{\mathrm{d}I(x,y,t)}{\mathrm{d}t} = 0 \tag{3.1}$$

$$I(x,y,t) = I(x+u, y+v, t+1) \tag{3.2}$$

From Equation 3.1, the total derivative is rewritten by the expression of the partial derivatives of x, y and t. So, we obtain the new expression (see Equation 3.3).

$$\frac{\partial I}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial I}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{\partial I}{\partial t} = 0 \tag{3.3}$$

In this equation, $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x, y, t). We rewrite them as I_x, I_y and I_t. And the derivatives of the positions with respect to time ($\frac{\mathrm{d}x}{\mathrm{d}t}$, $\frac{\mathrm{d}y}{\mathrm{d}t}$) are replaced by the optical flow expression (u, v). So we get the brightness constancy equation (see Equation 3.4). This equation can also be obtained from Equation 3.2 by a first-order taylor approximation.

$$uI_x + vI_y + I_t = 0 \tag{3.4}$$

Most of the optical flow estimation methods are based on this brightness constancy assumption. But this assumption is easily violated under non ideal visual conditions. So some more complex descriptions about the brightness change are also proposed [77].

Aperture problem and normal optical flow In fact, it is impossible to compute the motion vector (two unknowns) using only the brightness constancy equation (one equation). We can only compute its component in the direction of the brightness gradient (see Equation 3.5 and Figure 3.2). This is the so-called aperture problem. Such problem

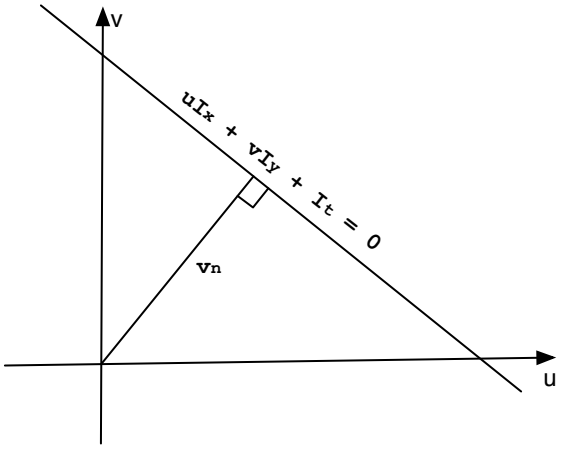


FIGURE 3.2: Mathematical representation of optical flow component v_n

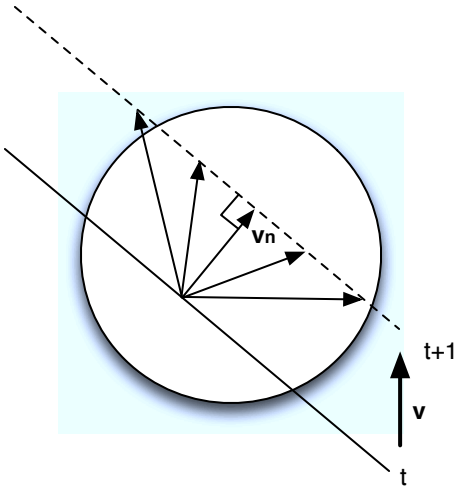


FIGURE 3.3: A demonstration of the aperture problem

can be explained from the example in Figure 3.3. When we look at the moving line from a small round aperture, we can not deduce how it moves correctly since we only know the motion component in the perpendicular direction of line.

$$v_n = -\frac{I_t}{\sqrt{I_x^2 + I_y^2}} = \frac{uI_x + vI_y}{\sqrt{I_x^2 + I_y^2}} \tag{3.5}$$

Difference between motion field and optical flow We have seen that the brightness constancy equation is the starting point for optical flow estimation. We will now discuss how the motion field is estimated by comparing the normal flow (Equation 3.5) - computed from the brightness constancy equation - with the normal component of the

motion field. From Appendix A.2, the image brightness (or the image radiance) I that is reflected by a Lambertian surface s is expressed by :

$$I = \frac{\rho}{\pi}\mathbf{E}^T\mathbf{n} \tag{3.6}$$

Where ρ is the surface's albedo, $\mathbf{E}$ defines the source intensity and the incident light direction and $\mathbf{n}$ is the normal unit vector of the surface s.

When we compute the total temporal derivation of the pattern brightness $\mathbf{I}$ from Equation 3.6, in the right side of the equation, only the normal vector $\mathbf{n}$ depends on time, so the total derivative is given by Equation 3.7.

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{\rho}{\pi}\mathbf{E}^T\frac{\mathrm{d}n}{\mathrm{d}t} \tag{3.7}$$

If the relative motion between the surface and the camera is composed by a translational motion $\mathbf{T}$ and a rotational motion $\mathbf{\Omega}$, then the temporal derivative of $\mathbf{n}$ is expressed by the vector product of $\mathbf{\Omega}$ and $\mathbf{n}$ (see Equation 3.8).

$$\frac{\mathrm{d}n}{\mathrm{d}t} = \Omega \times n \tag{3.8}$$

By injecting this expression into Equation 3.7, we obtain :

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{\rho}{\pi}\mathbf{E}^T(\Omega \times n) \tag{3.9}$$

In the left side of Equation 3.9, the total deviation can be rewriten as a function of the partial derivatives (see Equation 3.10). So we can obtain the expression of the difference of the normal vector Δv_n by comparing Equation 3.10 with Equation 3.4.

$$uI_x + vI_y + I_t = \frac{\rho}{\pi}\mathbf{E}^T(\Omega \times n) \tag{3.10}$$

$$\Delta v_n = \frac{\rho}{\pi}\frac{|\mathbf{E}^T\mathbf{\Omega} \times n|}{\|\nabla I\|} \tag{3.11}$$

The Equation 3.11 is valid under very restrictive assumptions like : lambertian surface, pointwise light source at infinity and no photometric distortion. But we cannot consider that Δv_n is equal to zero, which means that motion field is usually different from optical flow.

If we want to use optical flow as an approximation of motion field, Δv_n should be as small as possible or equal to zero. From Equation 3.11, Δv_n can be equal to zero under two conditions. Either there is a purely translational motion ($\boldsymbol{\Omega} = \vec{0}$), or there is a rigid motion and the illumination direction is parallel to the angular velocity ($\boldsymbol{\Omega} \times n| = \vec{0}$) [66]. Moreover, the value of Δv_n is inversely proportional to the magnitude of the spatial gradient of the image $\|\nabla I\|$. It means that motion field is well estimated on pixels with large gradient amplitude.

3.2.2.2 Parametric motion models

Since 2D motion depends on depth, object motion and camera motion, it means that pixels that belong to the same object at the same depth will have the same 2D motion. In a small neighborhood, we can make the hypothesis that this statement is true. For instance, we could consider that the optical flow of the pixels in small window is constant [78]. In practice, this hypothesis is not valid and leads to a wrong 2D motion approximation. In fact, we can only suppose that these pixels may share the same 3D motion parameters. In this case, instead of using two unknowns to represent the motion vector of each pixel, the pixels are grouped with common motion parameters. Then the optical flow of each pixel is expressed as a function of these motion parameters. Finally, we obtain the optical flow by estimating these motion parameters. The methods that are based on these parameter motion models are called "Parametric motion estimation" methods.

In this section, we present two current models and discuss the appropriate conditions to use them.

Affine model Affine motion model is given by Equation 3.12. Six parameters ($a_1..a_6$) are needed to describe this model. It can describe image translation, dilation, rotation and shear. This model is well appropriated in the case of large distances between the observed surface and the camera. Many classic estimation approaches can be applied using this model to compute the model parameters. When using this method, the affine motion parameters are considered the same for all the pixels of the image or in a specific region, then a least-square estimation of the parameters is performed using spatio-temporal gray-level gradients. This is an extension of the well-known Lucas-Kanade method [78].

$$\begin{cases} u(x,y) = a_1 + a_2 x + a_3 y \\ v(x,y) = a_4 + a_5 x + a_6 y \end{cases} \tag{3.12}$$

This affine model is a first-order function of the image coordinates. To deal with more complex motion models, we can use the second-order function of the image coordinates. This is the planar surface model.

Planar surface model If we consider that objects are sets of piecewise planar surfaces and that 3D motion of these planar surfaces is rigid with a translation motion $\mathbf{T}$ and a rotational motion $\mathbf{\Omega}$, we obtain motion equations with 8 coefficients (see Equation 2.12). These equations are second order functions of image coordinates and these 8 coefficients ($a_1..a_8$) are function of $\mathbf{T}$, $\mathbf{\Omega}$ and the plane parameter $\mathbf{n}$. We can notice that the affine motion model is a simplified version of the planar surface model. Since 8 parameters should be estimated, we should at least consider 8 pixels with the same motion parameters.

In fact, we can also use the 3D motion parameters T and Ω and the surface parameters n to express the 2D motion as a function of motion parameters 3.13. Now, 9 unknowns (T, Ω, n) must be estimated.

$$\begin{cases} \dot{x} = \frac{xy}{f}\Omega_X - (\frac{x^2}{f} + f)\Omega_Y + y\Omega_Z + \frac{1}{fd}(n_X x + n_Y y + n_Z)(xT_Z - fT_X) \\ \dot{y} = (\frac{y^2}{f} + f)\Omega_X - \frac{xy}{f}\Omega_Y - x\Omega_Z + \frac{1}{fd}(n_X x + n_Y y + n_Z)(yT_Z - fT_Y) \end{cases} \tag{3.13}$$

The most important advantage of these parametric motion models is that they help to find the internal link between pixels in the whole image or in small regions. For instance, the pixels from the same planar surface have the same motion parameters and the same plane parameters. Thinking it from a different way, if we consider that pixels with the same motion parameters results always from the same object, it means that parametric motion estimation should help object segmentation [79]. The segmented regions can be reused to improve the estimations of the parameters, and a joint motion estimation and segmentation algorithm can be designed [80].

3.3 Motion estimation approaches

In this section, we classify motion estimation approaches into three categories : Optical flow approaches, Block matching approaches and Parametric motion approaches. All these approaches are based on the brightness constancy assumption. Optical flow approaches use differential techniques to estimate the motion vector for each pixel. Block matching approaches search the similar block between images to estimate the motion

vector for each block. Parametric approaches estimate the motion parameters for each small region or for the whole image.

3.3.1 Optical flow approaches

Motion estimation techniques are designed to estimate optical flow (apparent motion) from image sequences, but here the "optical flow approaches" are defined specifically for the differential methods that use brightness constancy equation as starting point to solve optical flow. These differential methods are ulteriorly classified into global approaches and local approaches. Global approaches attempt to minimize a global energy function to get optical flow, while in local approaches the optical flow computation is based on local information.

This section will give an overview on these two approaches. In fact, both approaches have their advantages and drawbacks. For instance, global approaches allow to compute dense optical flow but local approaches are more robust to noise. According to this, [81] proposed an optical flow estimation method that combines global and local approaches.

3.3.1.1 Global approaches

Global approaches compute optical flow by minimizing a global energy function. The first global approach was proposed by Horn Schunk in [82]. Since then, many extensions are proposed to deal with different estimation problems and provide good results. For instance, [83] proposed a novel extended coarse-to-fine refinement framework to deal with fine motion structures, which can not be correctly estimated in the commonly multi-scale frameworks. Its effectiveness is demonstrated by Middlebury optical flow benchmark marking. So, this method will be compared with ours in the next sections of this chapter. We will call it later the " MDP" (Motion Details preserving) method. Although many methods are proposed, the typical formulation of this global energy function has slightly changed. It is expressed as the weighted sum of two terms E_d and λE_r :

$$E = E_d + \lambda E_r \tag{3.14}$$

$$E_d = \sum_{xy} \rho_d(I_1(x, y) - I_2(x + u, y + v)) \tag{3.15}$$

(a) Original image (b) Structure image (c) Texture image

FIGURE 3.4: Structure-Texture decomposition example

$$E_r = \sum_{xy} \rho_r(\frac{\partial u}{\partial x}) + \rho_r(\frac{\partial u}{\partial y}) + \rho_r(\frac{\partial v}{\partial x}) + \rho_r(\frac{\partial v}{\partial y}) \tag{3.16}$$

The first term E_d minimizes the brightness difference of the pixel in the different images. In [82], this difference is expressed by the brightness constancy equation. The penalty function is the L2 norm function $\rho(x) = x^2$. Due to the aperture problem, an additional assumption should be considered for the flow vector estimation. This assumption is contained in the second term E_r, which is also called the regularization term. The flow smoothness constraint is considered in [82] and the penalty function is still quadratic. The results from [82] is not satisfactory. Then, various modifications are made to improve results : different assumptions about the data term [84–88] or the regularization term [85, 88–94], choice of a penalty function or use of optimization algorithms.

Data term Although many state of the art algorithms use the brightness constancy assumption for the data term, this assumption is easily violated in natural scenes due to sensor noise, illumination changes, reflections, and shadows. The structrure-texture decomposition is applied by [84, 85] to deal with intensity changes. Since the image can be decomposed into a structural image, corresponding to low-frequency components and a textured image, containing fine scale-details, the texture image will be used for the estimation of optical flow. An example of such a structure-texture decomposition is shown in Figure 3.4 [1]. In our application, we find that the shadows on the road are only visible in the structure image.

Another way to cope with brightness changes is the use of photometric invariants to replace luminous intensity [86, 87]. But these methods require color images as input. A common approach in literature is to use another more robust assumption based on the constancy of the illumination [88].

1. *The structure-texture decomposition algorithm is presented in [95]*

$$\nabla I(x, y, t) = \nabla I(x + u, y + v, t + 1) \tag{3.17}$$

Such constraint is very helpful for translational motion, but the classical brightness constancy assumption suits better for more complicated motion patterns like local scale changes, rotations, etc.

Regularization term Apart from the spatial flow smoothness assumption that is defined in [82], [88–90] also propose to add a temporal smoothness term $\frac{\partial u}{\partial t}$ and $\frac{\partial v}{\partial t}$. Since smoothness constraint is not observed for object boundaries, we can weight the spatial smoothness assumption using a function of the image gradient ∇I (see Equation 3.18).

$$E_r = \sum_{xy} w(\nabla I)[(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2] \tag{3.18}$$

This weight should be inversely proportional to the image gradient, because the flow smoothness is always violated at the edges (high $|w(\nabla I)|$). However such a weight is isotropic. We can also use anisotropic weight functions to distinguish different directions. For example, a common solution is to weight the direction along the image gradient less than the direction orthogonal to it [91, 92].

Besides of smoothness assumption, the rigid-body motion assumption are also applied in [85, 93, 94]. This constraint is observed for the static scenes captured by a moving camera or for rigid objects moving in the scene.

Penalty functions Robust estimation methods were first introduced for optical flow by [96]. Since all the assumptions that are considered for flow estimation are not realistic and can be easily violated, and the least square methods are well known to be sensible to outliers, some authors propose robust estimation. The basic idea is to design a structure that can best fit the majority of the data while identifying and rejecting "outliers". Mathematically, the penalty function will be used to "weight" all the data. A good penalty function should make the influence of the outliers tend to zero. If we look for the penalty function of the least square estimation (see Figure 3.5(a)), the outliers (when x is far from zero) are highly weighted by this quadratic $\rho-$function. To understand this statement, we can look for the *influence function* $\psi(x)$, the derivative of $\rho(x)$. In the case of least square estimation, the influence of the data points increases linearly and without bound (see Figure 3.5(b)). In order to be more robust, L1 norm penalty function is also used in [97] [84] both for the data term and the regularization term. Comparing with the quadratic function, the influence is constant for all the data points (see Figure

3.5(c)(d)). Since the L1 norm is not derivable, a similar function like the Charbonnier (see Figure 3.5(e)(f)) is widely used [81]. A non-convex penalty function - Lorentzian - is used in [96]. From Figure 3.5(g)(h), the influence increases at first, then it starts decreasing for deviant points - the true outliers. So, this function should be very robust. According to [98], the performance of this Lorentzian penalty is worse than the less robust Charbonnier penalty because this non-convex function makes the optimization difficult to find the global optimum. Moreover, [92] make use of the Huber penalty (see Figure 3.5(i)(j)) for the regularization term and the results are also remarkable.

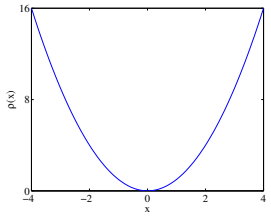
Optimization methods The minimization of the global energy function can be treated using Euler-Lagrange equations. To solve the linear Euler-Lagrange equations, a variety of techniques can be used like the Jacobi method, the Gauss-Seidel method, successive Over-Relaxation, and the Conjugate Gradient algorithm. From the non-linear Euler-Lagrange equations, an iterative method is always proposed to transform the non-linear equation of $\mathbf{u}$ into a linear equation of $\delta\mathbf{u}$ [2].

However, all these optimization methods need high computational cost, and sometimes, they can easily fall into the local minimum. To cope with these problems and also to estimate large motion, the coarse-to-fine strategies are used. The principle of these strategies is to build image pyramids. The flow estimation starts from the top level image (the coarsest image with the fewest pixels), the results are then oversampled and used as initial estimates for the lower level image. To estimate larger motion, we should increase the number of levels. But this makes detail information blurred or even disappeared in the upper levels. For instance, texture is completely absent on upper levels when dealing with images of a road that is filmed from a moving camera mounted in a vehicle. The road is often textureless and the egomotion due to the moving camera produces large displacements. Coarse-to-fine strategies do not work well in this case. For this reason, we propose a model-based optical flow method which is a good answer to deal with this kind of situations [99]. This method will be presented later in this chapter.

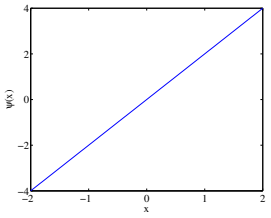
3.3.1.2 Local approaches

The first local approach for optical flow estimation was proposed by Lucas Kanade [78] in the same year of the publication of the first global approach [82]. It is actually an image alignment method. To estimate optical flow, images are divided into different sub-regions and the flow vector is assumed to be the same for all the pixels in the same sub-region. Then the alignment is made on these sub-regions between two successive

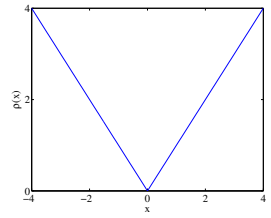
2. *The flow is represented as* $\mathbf{u_i} + \delta\mathbf{u}$. $\mathbf{u_i}$ *is known from the initialization or the previous iteration, and* $\delta\mathbf{u}$ *is the unknown vector*



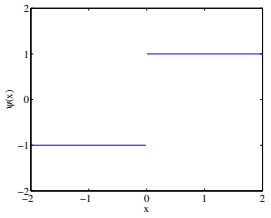
(a) L2 norm, $\rho(x) = x^2$



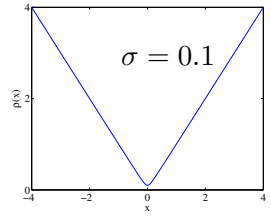
(b) Derivative function of L2 norm $\psi(x) = 2x$



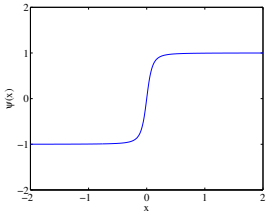
(c) L1 norm, $\rho(x) = |x|$



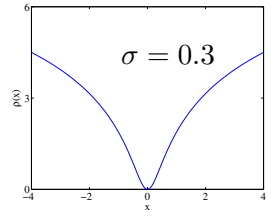
(d) Derivative function of L1 norm $\psi(x) = sgn(x)$



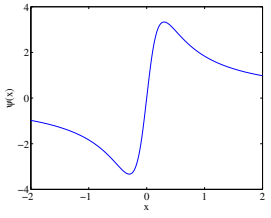
(e) Charbonnier, $\rho(x) = \sqrt{x^2 + \sigma^2}$



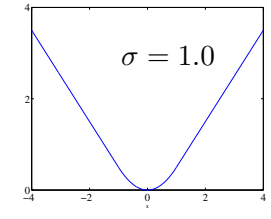
(f) Derivative function of Charbonnier $\psi(x) = \frac{x}{sqrt(x^2+\sigma^2)}$



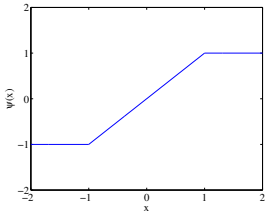
(g) Lorentzian, $\rho(x) = log(1 + \frac{x^2}{2\sigma^2})$



(h) Derivative function of Charbonnier $\psi(x) = \frac{2x}{x^2+2\sigma^2}$



(i) Huber, $\rho(x) = \begin{cases} \frac{x^2}{2\sigma} & |x| \leq \sigma \\ |x| - \frac{\sigma}{2} & \text{else} \end{cases}$



(j) Derivative function of Huber $\psi(x) = \begin{cases} \frac{x}{\sigma} & |x| \leq \sigma \\ sgn(x) & \text{else} \end{cases}$

FIGURE 3.5: Five commonly used penalty functions for optical flow estimation and their derivative functions

images. Mathematically, the flow vector on pixel x is computed by minimizing the sum of squared error between two images : the previous image I_1 and the current image I_2 :

$$\sum_{\mathbf{x}' \in S} [I_1(\mathbf{x}') - I_2(\mathbf{x}' + \mathbf{u})]^2 \tag{3.19}$$

Where $\mathbf{S}$ is the sub-region centered on $\mathbf{x}$. From this equation, the estimation of the optical flow at pixel $\mathbf{x}$ requires only the information of the pixels neighbors. That is why we call it a local approach.

An iterative algorithm is used to estimate $\mathbf{u}$ from Equation 3.19. In each iteration, the estimation $\mathbf{u}$ from the previous iteration is used as an initial guess, then only the increment flow $\delta\mathbf{u}$ is estimated. The equation is then rewritten :

$$\begin{aligned} &\sum_{\mathbf{x}' \in S} [I_1(\mathbf{x}') - I_2(\mathbf{x}' + \mathbf{u} + \delta\mathbf{u})]^2 \\ =&\sum_{\mathbf{x}' \in S} [I_1(\mathbf{x}') - I_2(\mathbf{x}' + \mathbf{u}) - \Delta I(\mathbf{x}' + \mathbf{u})\delta\mathbf{u}]^2 \end{aligned} \tag{3.20}$$

Since the problem becomes a linear system, we can compute δu using a gradient descent algorithm. Of course, other numerical algorithms such as difference decomposition [100] and linear regression [101] can also be used, but gradient descent is the standard and easy to implement. Once the increment δu is computed, we update the flow vector for the next iteration :

$$\mathbf{u} \leftarrow \mathbf{u} + \delta\mathbf{u} \tag{3.21}$$

This is the original formulation proposed by [78]. A wide variety of extensions have been proposed. For example, [102] proposed a compositional approach to replace the original incremental approach and the results were significantly improved. Many other algorithms use the different approximations (Gauss-Newton, Newton, steepest-descent or Levenberg-Marquardt) in each gradient descent step. These methods were presented and compared in [103]. Moreover, a real time and dense flow implementation was proposed by [104]. This method as an extension of the Lucas Kanade method will be compared with ours. We will refer to it as the FOLKI method.

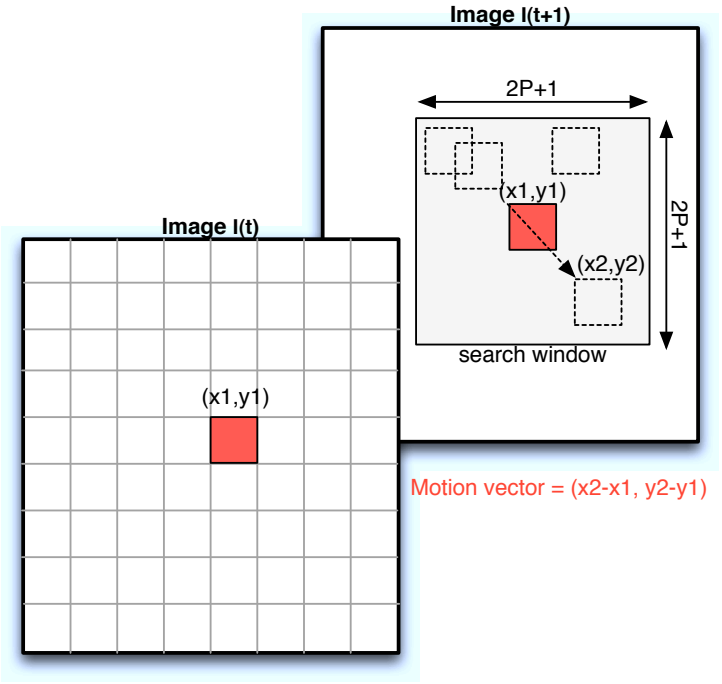


FIGURE 3.6: Block matching method for motion estimation

3.3.2 Block matching approaches

Optical flow approaches are also called pixel-based algorithms as they try to estimate motion vectors on each pixel in the image. While block matching approaches are referred as block-based motion estimation approaches because the image is divided into many blocks (for example 8*8 pixels or 16*16 pixels). Only a single motion vector is computed for each block. Block-matching methods are more easy to implement with real time constraints than dense methods. It is the reason why these methods are widely used in video coding standards and also for stereo vision systems.

We use Figure 3.6 to show how a block matching technique works. Images are divided into many square blocks and the matching compares these blocks between successive images (I_t and I_{t+1}). For example, for the block (x_1, y_1) in I_t (see the red block in the image I_t in Figure 3.6), a search is performed on the image I_{t+1} to find the best match based on a similarity measure. Then, the motion vector of this block is the displacement from the position of the current block (x_1, y_1) to the best matched block (x_2, y_2) in the image I_{t+1}. In order to confine the search of the similar blocks in a limited area, a search window is defined. In the example of Figure 3.6, search window size is $(2P+1)\times(2P+1)$. So, the maximum displacement that we can estimate from this example is P pixels on both horizontal and vertical directions.

Although the technique is easy to understand, the matching performances depend on many factors more particularly the following ones :

— The block size and shape
— The search strategy
— The similarity measure

The next sections will discuss the influence of these factors on the matching performances. Then, the optimization on the sub-pixel accuracy will be presented.

3.3.2.1 Block size

Standard block matching algorithm divide the image into square blocks of same size. Pixels in the same block are considered to undergo the same translation. This assumption is true when the block does not overlap objects with different motions. In fact, the determination of the block size is very important.

Small blocks are better for distinguishing between several motions in the block, but it is not possible to built enough discriminant information for computing similarity measures. On the other side, different motions may co-exist when the block size is large. According to [105], the choice of the block size depends on many factors such as motion vector accuracy, scene texture and inter-frame noise etc.

Since a fixed block size is not suitable for the whole image, [106] proposed a variable-size block matching algorithm to adapt different image areas. They divide the image into large blocks at first, then matching is performed on each block. The matching error of each block is compared to a threshold to decide whether the block should be further split. This top-down process continues until the matching error is under a threshold or until a prescribed minimum block size is reached.

3.3.2.2 Search strategies

After defining the block size, a search is performed in the search window to find the best match. For seek of simplicity, we compare all the possible blocks in the search window with the reference block. This full search strategy can obtain optimum results, but the computational cost could be very large. For instance, in the case of a block size of $n \times n$ and a search window of $m \times m$, the search complexity is proportional to $m \times m$. Therefore, the choice of the search window size is a compromise between the computational cost and the estimation range. However, to accelerate the search process, different search strategies are proposed as an alternative to the exhaustive search. For

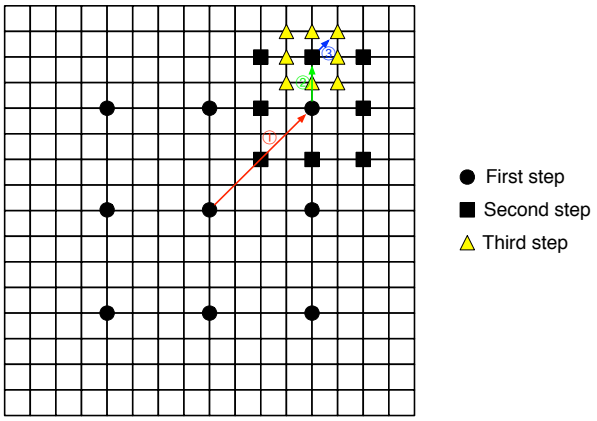


FIGURE 3.7: Three step search algorithm

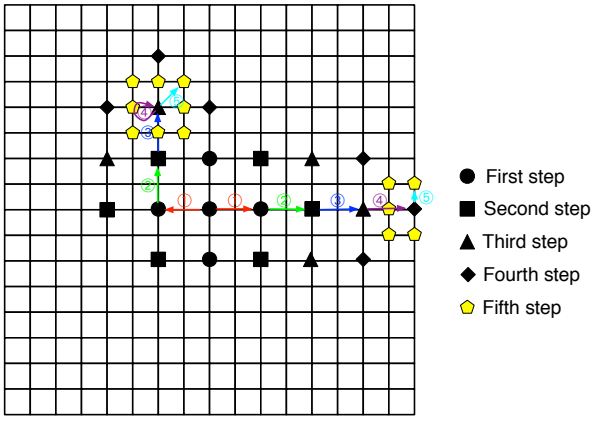


FIGURE 3.8: 2D-logarithmic search algorithm

example, three step search algorithm [107] is a coarse to fine search mechanism. The idea is to halve the search size after each step until the search size reaches 1 (see Figure 3.7). 2D-logarithmic search algorithm [108] also starts with a rough search, then the step size is reduced by half only when the best match is in the center or the best match reaches the search window boundary (see Figure 3.8). Diamond search [109] starts the search along a large diamond (9 pixels) in the center of the search window. When the best match is in the center, we reduce the diamond size to 5 pixels (see Figure 3.9).

3.3.2.3 Similarity measures

When measuring the similarity of two blocks, we should at first choose the appropriate attributes for performing the comparison. Generally, the intensity value or the color are used as a measure. This is under the assumption that the illumination has not changed between the images. We know that this assumption is often violated. So, block textures are also used as comparison element. For instance, the local binary pattern (LBP) [110]

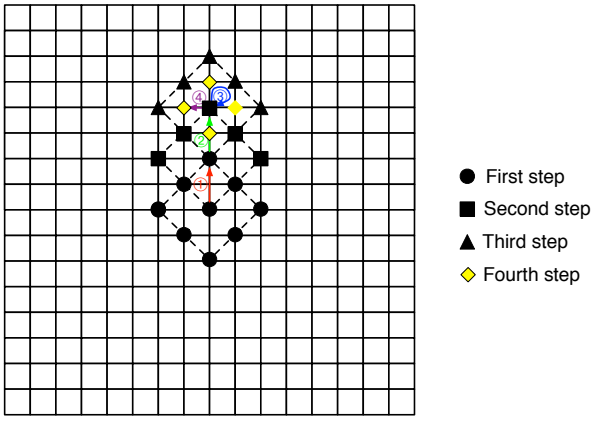


FIGURE 3.9: Diamond search algorithm

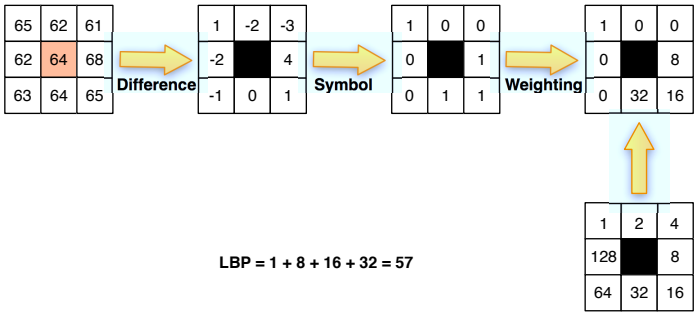


FIGURE 3.10: Local Binary Pattern computation

is widely used to describe the texture due to its computational simplicity. As shown in Figures 3.10, the LBP operator forms labels for each pixel by thresholding the 3×3 neighborhood with the center value and then considering the result as a binary number.

Besides, there are also many possibilities for choosing the matching criteria like : sum of squared difference (SSD), mean squared difference (MSD), sum of absolute difference (SAD), mean absolute difference (MAD) etc.

3.3.2.4 Sub-pixel accuracy

In order to obtain a sub-pixel accuracy, a simple way is to interpolate gray levels between two successive images. Different interpolation techniques such as Nearest-neighbor interpolation, Bilinear interpolation or Bicubic interpolation can be used. Nearest-neighbor interpolation is the simplest interpolation since the value of an interpolated pixel is the copy of the nearest pixel. To obtain smooth results, we can use the bilinear interpolation. In this method, the interpolation value depends on the four neighbors. An example is given in Figure 3.11. The value of the interpolated pixel P is computed using the four neighbors P_1, P_2, P_3 and P_4. Their contributions are inversely proportional to the

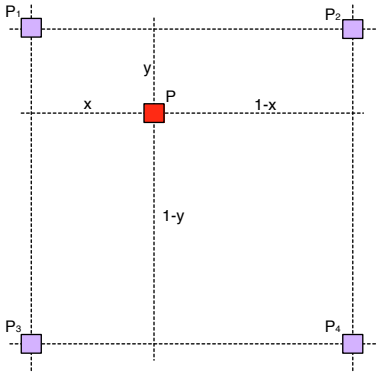


FIGURE 3.11: Bilinear Interpolation Schema

distances to P. The computation can be found in Equation 3.22, where x and y are normalized distances.

$$P = (1-x)(1-y)P_1 + x(1-y)P_2 + (1-x)yP_3 + xyP_4 \tag{3.22}$$

The interpolation results of the bicubic method are smoother than for the bilinear method, but the computation is more complex as it considers neighborhood of 16 pixels (4×4) for each pixel interpolation.

However, interpolation can only increase the pixel accuracy by two (1 pixel to 0.5 pixel). To obtain a higher resolution, an iterative interpolation should be used. We can also mix different interpolation techniques. For instance, the video compression format H.264/AVC uses a 6-tap filter for half-pixel interpolation and then a simple linear interpolation to achieve quarter-pixel precision from the half-pixel data.

3.3.3 Parametric motion approaches

Block matching approaches use a constant motion assumption for local pixels while optical flow approaches consider a smoothness assumption for local pixels. In fact, we can find more complex and accurate relations between local pixels. That is the principle of parametric motion models (see Section 3.2.2.2). It means that pixels are considered with their corresponding motion parameters and the estimation concerns these parameters.

In fact, we can use optical flow approaches to estimate these motion parameters. For example, in [111] [96], the local approach - Lucas-Kanade algorithm - is used to computer motion parameters by a least square method. Global approaches can also be used, but we should replace the flow smoothness constraint by the motion parameter smoothness constraint. The selection of the motion model is important. The model should be as

simple as possible but also describe the significant properties of motion. For example, in Section 3.2.2.2, we have presented two simple but popular motion models : affine model and planar surface model. Affine model is a first order function of image coordinate, there are fewer parameters to be estimated. The planar surface model can model more complicate situations but the parameter estimation is more difficult as 8 parameters should be estimated.

3.4 Introduction to the optical flow evaluation

Since there are so many different flow estimation methods, an appropriate benchmark evaluation is required to rank them and also to encourage the improvements. In fact, there is a long history on the evaluation researches. The first study is from [112]. It tested nine optical flow estimation methods on five synthetic image sequences and four real image sequences. As the ground truth was available for the synthetic images, the angular error was computed to each estimation method for comparison. Since there was no ground truth for real images and evaluating only synthetic images being not convincing, McCane et al. [113] proposed an approach to generate motion field from real sequences that contain polyhedral objects, at the same time more complex and realized synthetic sequence with ground truth were created for a quantitative evaluation. Finally, an exhaustive evaluation was proposed by [114] who asked for the participation of all the state-of-the-art estimation methods. In this benchmark, the measurements include the average angular error (AE), the endpoint error (EE), the interpolation error (IE) and the normalized interpolation error (NE). This benchmark has promoted the development of new flow estimation methods since its release in 2007. But there still exist some deficiencies. For example, no long sequence was provided in the database, most of them had only 8 frames or even image pairs. So all the methods that take advantage of history in long sequence can not be equally evaluated from this benchmark. Moreover, large motion that is very common in real scenes is not present in this benchmark. However, [115] proposed a new dataset MPI-Sintel to fill the gap. These data came from an animated short film Sintel. Many important features of natural scenes, like long sequences, large motion, specular reflections, motion blur, defocus blur, atmospheric effect, etc. are presented in these films. So we could test the robustness of different estimation methods toward these natural perturbations. Besides, some benchmarks were proposed specifically for ADAS applications [116] [117]. Data from these benchmarks came from real traffic outdoor scenes. Unfortunately, the ground truth was provided by using range sensing or stereo.

3.4.1 Ground Truth Computation

A key point for optical flow evaluation is to get the ground truth. Unfortunately, no sensor can provide a correct measure of optical flow. However, we have the possibility to generate synthetic images since all parameters are known to compute 2D motion field directly. Another advantage to use synthetic images is that we could render scenes under conditions of different complexities. Moreover, it is possible to generate various scenes in term of contain or quality or illumination conditions that allow to test the robustness of the estimation method. Although, new synthetic scene generators make synthetic image quite similar to real scene images. However, it is very difficult to reproduce exactly in synthetic images some illuminations and textures or the effect of real noise. It is the reason why the ultimate evaluation should be on real scene images.

3.4.1.1 Ground truth for synthetic images

In this section, we will present two synthetic examples and see how to obtain the ground truth flow for these synthetic images. The first database is MPI-Sintel presented in [115]. Data came from an animation film. The second database is called SiVIC as all the synthetic images are generated by the SiVIC simulator [61] designed specifically for ADAS applications.

MPI-Sintel dataset This data is derived from the animated short film Sintel [118] created by the software Blender [119]. Since all the source files about this film are released on open source, [115] modified this film both on the production data and the rendering to make the data scientifically interesting and suitable for the use of flow evaluation. The modification efforts can be found in [120].

Since it is computer-generated, the scene geometry and its change over time are known before. So we can track each pixel p_t from the image plane to its original 3D point P_t in the scene, then locate its new position P_{t+1} in the next frame and compute its projection p_{t+1} in the next frame. The optical flow ground truth is therefore given by $u_t = p_{t+1} - p_t$.

Although this is a synthetic dataset, the first-order image and motion statistics are very similar to those reported in the literature for natural scenes [121–123], and the use of the comparison set of "Lookalike" image sequence from real films makes this database very interesting to deal with real-world videos.

SiVIC dataset from SiVIC simulator SiVIC (Simulateur Vehicule-Infrastructure-Capteurs) [61] is a virtual sensor prototyping platform created by the French research

institute on transportation (IFSTTAR). It is designed for ADAS evaluations as it can accurately reproduce road situations, the vehicle's behavior and the functioning of sensors that are embedded in a vehicle like cameras, telemeters, inertial navigation system, etc. So we can create different scenarios with predefined parameters such as motions, textures, depth, etc. It is then possible to generate motion field from depth and camera motion information. So, this dataset is very suitable for optical flow evaluation in the context of ADAS applications.

3.4.1.2 Ground truth for real scene images

It is difficult to measure the optical flow ground truth from real scene images, as the situation is very different from other domains such as stereo, 3D reconstruction, segmentation or object recognition, where the ground truth is given from a sensor or by manual labeling. In order to obtain the optical flow ground truth, additional information should be used such as special texture patterns [114] or range imagery [116, 121]. The next part will present these two ground truth optical flow generation methods : (a) hidden fluorescent paint and (b) sensor fusion. However, we should recall that the estimated optical flow from these techniques are not the **real** flow. We call it ground truth only because these flow results are much more accurate than those from the under-tested estimation methods.

(a) Hidden fluorescent paint We know that UV (Ultraviolet) is more energetic than visible light. When an UV light shines in a hidden fluorescent paint, the UV will be 'stepped down' energetically into longer wavelengths and then the paint will be visible. This hidden fluorescent paint appears transparent under visible lighting. This property is then used to obtain the ground truth optical flow by [114]. Fluorescents patterns are painted onto objects. Then, pictures are taken simultaneously under ambient and UV light to capture both a natural image and an image with rich textures from the fluorescent patterns. Finally, a feature matching method is used to estimate the ground truth optical flow. Such methodology of painting fluorescent materials to track motion is only limited to indoor scenes and artificial motion.

(b) Sensor fusion In a static environment, optical flow can be computed from camera depth and motion information. This information could be directly measured from specific sensors. For example, Inertial Measurement Unit (IMU) and Global Positioning System (GPS) can provide accurate camera movement and position data. Depth information can be estimated by Laser Scanners like LIDAR or structured-like systems such as Microsoft Kinect. These two depth measurement methods will be detailed below. So, the ground

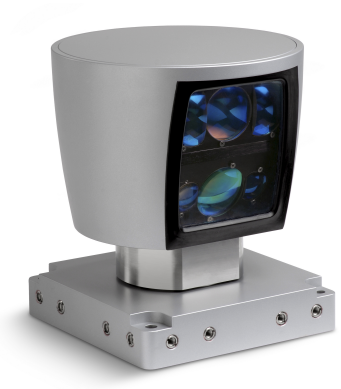


FIGURE 3.12: Velodyne HDL-64E LiDAR used by KITTI benchmark

truth optical flow can be estimated using sensor fusion. A method that fuses IMU and LIDAR can be used for outdoor scenes, but these sensors are very expensive. Moreover, calibration between different sensors is not straightforward. This method also suffers from occlusion, artifacts, noisy depth data, and interpolation error.

LIDAR LIDAR (**Li**ght **d**etection **a**nd **r**anging) is a very popular remote sensing technology. It measures the distance to a target by illuminating the target with a laser pulse and by analyzing the reflected light. The high precision (about several centimeters) and the large range (80 to 100 meters) make it also used for ground truth computation [116]. Unfortunately, the cost and the computation complexity with the collected points cloud make this technology difficult to integrate.

Structured-light system Structured-light system consists of a structured-light projector and a camera system that exploit depth information. The projector illuminates a known pattern (usually : stripes) onto a three-dimensionally shaped surface. Then, the camera captures the surface from a different viewpoint. From the captured image, the pattern is geometrically distorted due to the surface shape of the object. Depth information can be obtained by analyzing this the pattern distortion.

Microsoft Kinect [124] uses this technology to generate a robust depth map. The projected pattern is the infrared laser light points called optic codes. By reading the deformed points captured from the camera, a depth map with high precision is created. Despite of its high performance in an indoor environment, distance range limits its use to an outdoor situation (the maximum distance is about 4m).

FIGURE 3.13: Microsoft Kinect

3.4.2 Application-based optical flow evaluation

From the above presentation about the estimation of optical flow ground truth, it is obvious that the **real** ground truth can be measured only from synthetic images. For real scene images, the accuracy of the optical flow ground truth depends on computation technologies (for example feature matching for hidden fluorescent paint) or sensor precisions (for example sensor fusion methods). Therefore, the estimated optical flow can be only considered as a kind of reference data. The evaluation is then based on these reference data and some measure indicators (usually the angular error and the end-point error). If we can not improve the ground truth accuracy, maybe we should think about more significant indicators to judge the flow estimation.

Since optical flow is a set of 2D vectors, the evaluation of a vector is usually the evaluation of the magnitude and the direction. Therefore, the angular error and the end-point error are generally used as indicators in existing benchmarks. However, these measurements are incomplete as the optical flow reference is not a "real" ground truth. In computer vision applications, optical flow is usually used to recover the egomotion or segment objects. In another sense, if the estimated optical flow allows to recover a more accurate egomotion or better segment objects, this optical flow can also be considered as a good flow. Based on this point of view, we propose three error measurements depending on the application. Those that estimate the FOE error and speed error are used for egomotion recovery. Plane segmentation evaluation using "c-velocity" is suitable for object segmentation applications. We know that the "c-velocity" approach takes advantage of optical flow to segment planes and that the segmentation results depend on flow accuracy. Therefore, we can evaluate optical flow by analyzing the segmentation results from the "c-velocity" method.

Angular Error This indicator computes the angular difference in the homogeneous notation space between the estimated flow vector and the ground truth vector. We define the estimated flow as $(u,\ v,\ 1)$ and the ground truth flow as $(u_{GT},\ v_{GT},\ 1)$, then the angular error can be computed by

$$AE = \arccos\left(\frac{1.0 + u * u_{GT} + v * v_{GT}}{\sqrt{1.0 + u^2 + v^2}\sqrt{1.0 + u_{GT}^2 + v_{GT}^2}}\right) \tag{3.23}$$

This presentation can avoid the "divide by zero" problem for zero flows. But some bias still exists . For example, angular errors in large flow are less penalized than errors in small flow.

End-point Error As angular errors suffer from flow magnitude, an absolute error seems more accurate. Besides, the flow norm $\|w\|$ is used in the "c-velocity" approach. So, this indicator is very useful for estimating the flow that is used in the "c-velocity" approach. The computation is shown in Equation 3.24

$$EE = \sqrt{(u - u_{GT})^2 + (v - v_{GT})^2} \tag{3.24}$$

FOE Error Assuming that the camera moves straight (rotations are negligible), the flow field of all static points converge/diverge to the FOE point. So, the FOE position indicates the motion direction of the camera and thus provides an estimation of the camera motion. Since the FOE position can be estimated from the optical flow, we also use it as indicator to estimate optical flow. The ground truth FOE position is computed directly from the camera's motion (see Equation 3.25).

$$\begin{cases} x_{GT} = f\frac{T_X}{T_Z} \\ y_{GT} = f\frac{T_Y}{T_Z} \end{cases} \tag{3.25}$$

The FOE error is considered as the euclidean distance between the true FOE position (x_{GT}, y_{GT}) and the FOE position (x, y) estimated by optical flow.

$$FOEE = \sqrt{(x - x_{GT})^2 + (y - y_{GT})^2} \tag{3.26}$$

Many different methods can be used to estimate the FOE point from optical flow vectors (See Chapter 4) but here we use the same approach that is presented in [70].

Speed Error If we consider that the FOE position tells the observers which way they are heading, then the forward speed tells the observers how fast they are moving. These two parameters can help the interpretation of 3D motion of the camera. Since the FOE position is evaluated, the "Speed Error" relating to the forward motion error

of the camera can also be exploited. This forward speed can be simply computed from the optical flow of static pixels (for example, road pixels) by a least square method and then the speed error is the relative error with the real vehicle speed (see Equation 3.27).

$$SE = |T_Z - T_{GT}| \tag{3.27}$$

Evaluation of plane segmentation using "c-velocity" In Chapter 2, we have presented the "c-velocity" approach that could be considered as an object detection method that exploit optical flow norm information. In this method, the objects are approximated by planes of different orientations. Projections of these objects onto the corresponding "c-velocity" spaces are straight lines. All these lines pass through the origin. So, a 1D Hough transform can be used to extract these lines. At the same time, we find that the "accuracy" of the line shape in the "c-velocity" image depends on the accuracy of optical flow information. When the flow is not correct, the line in "c-velocity" image and the corresponding peak in Hough transform histogram become thicker. So we can not easily detect all the plane pixels from this "thick" peak. Using this property, we can use the detected plane pixel to evaluate the optical flow. In an intelligent vehicles application, the road is always considered as an horizontal plane to evaluate the optical flow.

3.5 Flow compensation method

Many state-of-the-art motion estimation methods gain excellent results in optical flow evaluation benchmarks but we find that they are not suitable for our "c-velocity" approach. After analyzing the estimation results on different objects, we get the following conclusions. First, possible objects arising in an urban road situation are sky, road, buildings, obstacles (pedestrians or vehicles). Motion estimation of sky pixels is meaningless as the sky is at infinity and the motion should be zero. Therefore, we should first filter sky region before motion estimation. Second, estimation results of buildings and obstacles by different methods are acceptable for "c-velocity" approach, but the motion results of the road pixels are very bad and sometimes totally wrong. Failures of these methods on road pixels are caused by the specificity of the road : poor texture and large displacements due to vehicle egomotion. Especially, for road points close to the camera, large motion problem is usually solved by coarse-to-fine strategies but these strategies can lead to a poor texture in the upper level, not to mention the originally poorly-textured scene. Therefore, we propose to assist the classic optical flow process by exploiting both a 3D

scene model and a rough velocity estimate from either other embedded sensors or ego-motion estimations from the previous frames. As using the available a priori knowledge allows to compensate the dominant flow to facilitate the estimation of the remaining part by a classical optical flow method.

In the rest part of this section, we will present the reason why we proposed this new method at first. Then an overview of this new approach will be presented. Finally, both synthetic and real image test will be shown and compared with other existing methods.

3.5.1 Background

Optical flow approaches are based on the brightness constancy equation (see Equation 3.2). Then, a first order Taylor expansion leads to the well-known optical flow equations (see Equation 3.3). This expansion is valid when the optical flow is very small. To estimate large motion, coarse-to-fine strategies are employed. These strategies allow estimating optical flow in an image pyramid. In the top level, a coarse scale image is used to estimate optical flow. Then, estimates are up-sampled as initial flows to estimate the increment flow in the level below. These steps are iterated until the original image scale level is reached. This ensures the validation of small motion assumption and also avoids to fall into a local minimum. In this strategy, the number of levels depends on the magnitude of motion. If we consider that the top level can estimate only one pixel-size motion, seven levels are then needed to estimate motions around 64 pixels. Supposing now that the original image size is 128×128, then the image size in the seventh level is 2×2. We can not get the accurate estimation in the image of size 2×2 without any texture information. This example shows that the coarse-to-fine strategy is not useful for very large motion.

Optical flow is rarely implemented solely on intelligent vehicle systems. It is combined with other sensors likes radars [14] or stereovision [15] and plays consequently just a supporting role. One can quote two principal reasons [125] : large displacements between consecutive frames and lack of textures in some regions. For example, the road is an essential object in the intelligent vehicles scenes and also a region that accords with the above properties. More precisely, near the car, road details could be used to estimate the relative displacement, but it can be very large (several tens of pixels) which makes most local strategies inefficient and hierarchical approaches cannot be used because details are blurred in the upper levels. At a longer distance, motion is much smaller, but the lack of texture in some regions always introduces a significant instability in the computation of spatial derivatives [126]. Therefore, we could refer ourselves to an interpolation frame to compensate the pixel displacement caused by the camera motion. Here, the interpolation

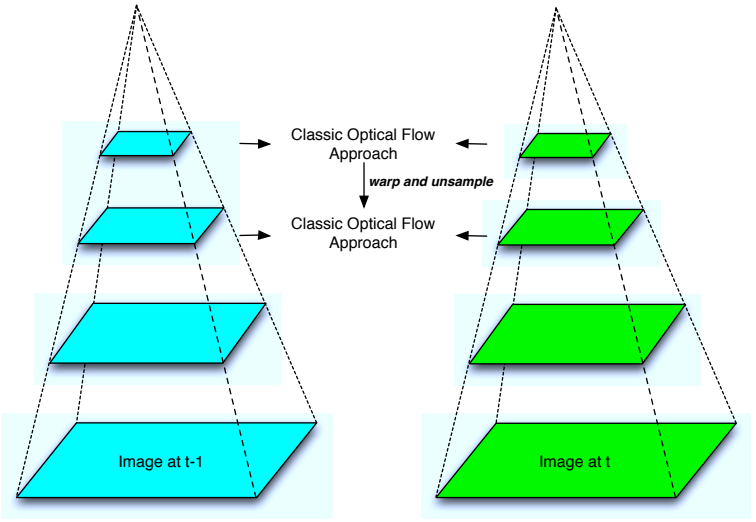


FIGURE 3.14: Coarse-to-fine optical flow estimation

is based on a plane model and a rough camera motion can be provided by other embedded sensors like Inertial Measurement Unit (IMU) or from motion estimation in the previous frames. Then, a classical hierarchical optical flow method is applied to estimate small displacements. We focus in this study on the road motion model (horizontal plane) but the approach could easily be generalized to other plane models. We compare our method with classical optical flow methods and show how results are improved.

3.5.2 Overview of our approach

Let us consider a camera mounted on a vehicle moving on a flat road. The optical axis is supposed to be parallel to the road surface which means that the road can be modeled as a horizontal plane. 2D motion vectors corresponding to the road can be expressed using the well-known 2D motion equation of 3D planes (see Equation 3.28).

$$\begin{cases} u = \frac{xy}{f}\Omega_X - (\frac{x^2}{f} + f)\Omega_Y + y\Omega_Z + \frac{xyT_Z - fyT_X}{fd} \\ v = (\frac{y^2}{f} + f)\Omega_X - \frac{xy}{f}\Omega_Y - x\Omega_Z + \frac{y^2T_Z - fyT_Y}{fd} \end{cases} \tag{3.28}$$

Where $\mathbf{T} = (T_X, T_Y, T_Z)$ and $\mathbf{R} = (\Omega_X, \Omega_Y, \Omega_Z)$ are the 3D translational and rotational motion of the vehicle, f the focal length and d the distance between the road and the camera (camera height). We assume first that the dominant motion of the vehicle is longitudinal with translational T_Z, which is true for most cases in this kind of applications. A rough estimate value T_{Z_0} of this translation can be provided from other sensors or from the estimation of a previous frame. Equation 3.29 gives the relation between the

translation and its estimate using the error ΔT_Z.

$$T_Z = T_{Z_0} + \Delta T_Z \tag{3.29}$$

One can add Equation 3.29 to Equation 3.28 and find that a motion vector is made up two parts $U_0 = (u_0, v_0)$ and $\Delta U = (\Delta u, \Delta v)$ (see Equation 3.30). The former part can be computed directly by Equation 3.31. The latter part can be estimated by any classical optical flow estimation method as it is very small.

$$\mathbf{U} = \mathbf{U_0} + \mathbf{\Delta U} \tag{3.30}$$

$$\begin{cases} u_0 = \frac{xyT_{Z_0}}{fd} \\ v_0 = \frac{y^2 T_{Z_0}}{fd} \end{cases} \tag{3.31}$$

$$\begin{cases} \Delta u = \frac{xy}{f}\Omega_X - (\frac{x^2}{f} + f)\Omega_Y + y\Omega_Z + \frac{xy\Delta T_Z - fyT_X}{fd} \\ \Delta v = (\frac{y^2}{f} + f)\Omega_X - \frac{xy}{f}\Omega_Y - x\Omega_Z + \frac{y^2\Delta T_Z - fyT_Y}{fd} \end{cases} \tag{3.32}$$

Let us consider two successive frames I_1 and I_2. Using the above decomposition, a rough motion vector estimate is computed directly from Equation 3.31 using a priori knowledge about vehicle speed. This estimate is used to construct an interpolated intermediate image I'_1 from I_1. Then, ΔU can be estimated by a classical optical flow estimation method between I'_1 and I_2 instead of using directly two successive frames. We can easily find the relation between I_1 and I'_1 (Equation 3.33). In our case, both bilinear and bi-cubic interpolations are tested.

$$I'_1(x', y') = I_1(x + u_0, y + v_0) \tag{3.33}$$

The model-based compensation using a priori knowledge about the vehicle speed allows to reduce frame difference and to estimate accurately the dense flow ΔU between I'_1 and I_2 using a classical optical flow estimation approach. One can find several techniques in the state-of-the-art that are generally classified onto two groups. Those that stem from the Horn & Schunk method relies on a global energy minimization that results from a brightness constancy term and a smoothing constraint (regularization term). They are known to be sensitive to noise but give a dense flow and are able to deal with homogeneous regions thanks to the regularization process. Another category stems from the Lucas & Kanade method, which is more robust to noise, and is based on the assumption that the flow is constant in a small neighborhood. The basic brightness constancy equations are solved in a local window (centered around each pixel) using a

least square criterion. In order to choose a suitable method to estimate ΔU, we selected on recent technique from each category : Motion Detail Preserving (MDP) [83] and FOLKI [104] as well as a classical block matching method for the test. Results and their comparisons are given in the next section.

3.5.3 Experimental results

The approach was tested and validated using both synthetic and real image sequences. All synthetic data stem from the French research institute on transportation (IFSTTAR). The image database results from a driving simulator [61] that allows to modify several parameters including egomotion but also scene depth, texture, etc. All the given parameters allow to compute a ground truth 3D motion as well as the corresponding 2D motion field. The real image sequence database is from the KITTI database [116] which provides various real outdoor sequences of car driving.

3.5.3.1 Results from synthetic images

We compare our method, called Model-based Motion Compensation (MMC), with some classical hierarchical optical flow methods (Horn&Schunck [127] and Lucas&Kanade [128]), two recent variants (MDP [83] and FOLKI [104]) and a full-search block matching method. Comparison criteria include traditional angular and endpoint errors and relative error of two flow vectors. Since the 2D velocity of the road is due to the egomotion of vehicle, we recompute the vehicle egomotion from the flow field and compare it with the real 3D motion of the vehicle. Assuming that the camera moves straight (rotations are negligible), the optic flow field of all static points converge/diverge to a particular point in the image called "focus of expansion (FOE) defined by Equation 3.34. As it plays an important role in many ADAS vision applications such as three-dimensional reconstruction, range estimation, time-to-impact computation, and obstacle avoidance, we also consider the error on its position as an important comparison criterion. In addition to this, another segmentation criterion is defined using the "c-velocity" method [70, 129, 130].

$$\begin{cases} x_{foe} = f\frac{T_x}{T_z} \\ y_{foe} = f\frac{T_y}{T_z} \end{cases} \tag{3.34}$$

Interpretation from statistics Table 3.1 shows the results using different methods including our approach. The above mentioned criteria are computed for each method. The "Speed Error" is related to the forward motion error of the vehicle. Since in our

considered scenes, the most important vehicle motion is the forward speed. Moreover, the optical flow of road is computed using the vehicle egomotion, then we recover the forward speed of the vehicle using the road flow field by a least square method and compute the relative error with the real vehicle speed. From optical flow, the FOE is also recomputed with the same approach in [70]. Its distance from the real FOE position is considered as the FOE Error.

TABLE 3.1: Comparison of different techniques

Technique	ARE Vx	ARE Vy	AAE (radiant)	AEE (pixel)	Speed Error	FOE Error (pixel)
Horn & Schunck	0.796245	0.867634	0.575541	21.7678	0.9139	9.85
Lucas & Kanade	1.06093	0.912752	1.33899	24.8179	0.8795	2.4638
MDP	0.940293	1.00278	1.10666	24.4643	0.9276	2.7192
FOLKI	0.559967	0.480318	0.202962	14.6121	0.3535	2.4958
Block Matching	1.15774	0.937277	1.49806	25.0439	0.8061	0.8764
MMC-MDP	0.0970013	0.0870454	0.0334058	1.71356	0.01994	2.92667
MMC-FOLKI	0.249	0.153	0.070	3.36146	0.0607	1.7368
MMC-BM	0.242834	0.159128	0.0838975	3.28049	0.0313	0.6992

From Table 3.1, we can see that combining any of the three selected methods (FOLKI, MDP or the Block matching) with our approach (MMC) gives better results than any of them alone. The direct MDP method, which is one of the most precise method using the Middleburry benchmark, shows less precise results than FOLKI using our test image sequences. However, when combining our method with the above three cited approaches, MDP is the best combination while FOLKI is the worst one. Moreover, FOLKI seems to be more robust with large displacement and MDP more accurate for small inter-frame difference. Here we use a full search block matching method with a kernel window size of 16 * 16 and a search window size of 10 * 10. In order to test the tolerance of our approach to intial speed variation (IS), we evaluate our approach with different IS. The relative errors and the egomotion errors are shown in Figure 3.15(a) and Figure 3.15(b). Apparently the combination with block matching or MDP is very tolerant to IS variation. and the combination with FOLKI has small working bandwidth.

(a) Relative error (b) Egomotion Error

FIGURE 3.15: Tolerance comparison of different classical methods

Evaluation of road Segmentation using "c-velocity" Let us assume that the 3D scene could be approximated by a set of 3D planes with different orientations : lateral planes (buildings), horizontal planes (the road) and frontal planes (moving cars or crossing pedestrians). The "c-velocity" method [129, 130] aims to detect 3D planes by considering optical flow information. Authors prove that for a specific plane orientation there is a linear relationship between the flow amplitude and a curve parameter called "c-value" that depends only on pixel coordinates (see Equation 3.35). A 3D plane is then represented in a cumulative space (w, c), called c-velocity, by a line of slope k that could be extracted using a Hough transform. Finally, a tough problem as the detection of a parameterized surface from a moving camera is reduced to an easy maxima finding in the Hough space. Authors prove also that an inaccurate optical flow produces a spread line in the c-velocity space instead of a thin line. As a consequence, we propose to use the line extent in the c-velocity space as an optical flow indicator of accuracy. Figure 3.16 shows the line (or the line slope k) histograms of the road after Hough transform using different flow methods. And the peak fineness reflects the using flow accuracy. Here, only the FOLKI method (Figure 3.16(b)) and our approach (Figure 3.16(a)) provide finer peaks and detect successfully the road plane (Figure 3.17) and their road pixel detection rates are shown in Table 3.2. Moreover, we can see that our results are closer to the ground truth (Figure 3.16(g)).

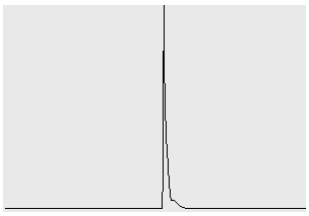
$$w = \begin{cases} k * c_{road} & c_{road} = |y|\sqrt{(x - x_{foe})^2 + (y - y_{foe})^2} \\ k * c_{building} & c_{building} = |x|\sqrt{(x - x_{foe})^2 + (y - y_{foe})^2} \\ k * c_{obstacle} & c_{obstacle} = \sqrt{(x - x_{foe})^2 + (y - y_{foe})^2} \end{cases} \tag{3.35}$$

TABLE 3.2: Road pixel detection rate from MMC-MDP flow and FOLKI flow

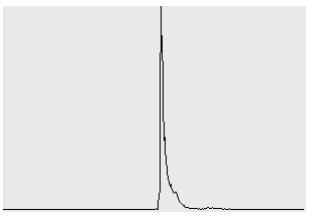
	detected road pixel rate (%)	error road pixel rate (%)
FOLKI	85.99	2.81
MMC-MDP	97.23	0.89

3.5.3.2 Results from real images

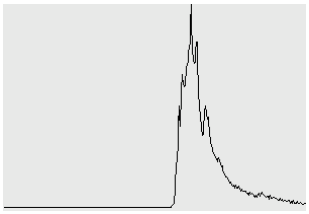
Real image data came from the KITTI database including challenging real-world computer vision benchmarks. All the sequences are captured by cameras mounted on a car moving in urban situations. The IMU system is also equipped to give the position and speed information. As we have no ground truth about the vehicle motion, we could not test our method with the same criteria than those used for synthetic data. However, we can compute an interpolation error as given by Equation 3.36.



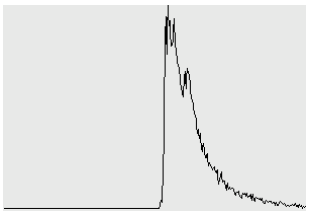
(a) "Road" histogram from MMC-MDP flow



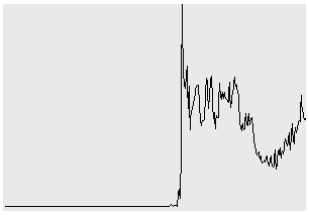
(b) "Road" histogram from FOLKI flow



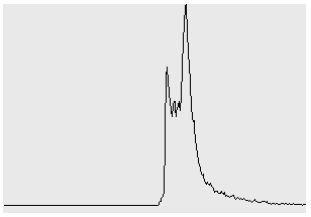
(c) "Road" histogram from Horn&Schunck flow



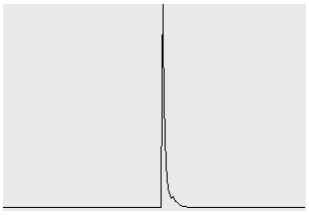
(d) "Road" histogram from Lucas&Kanade flow



(e) "Road" histogram from MDP flow

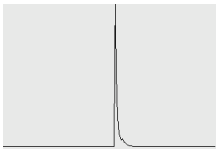


(f) "Road" histogram from Block Matching flow

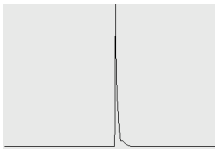


(g) "Road" histogram from ground truth

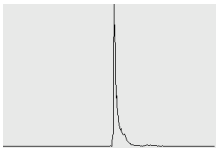
FIGURE 3.16: "Road" c-velocity histograms from different flows



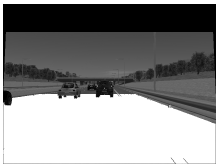
(a) "Road" histogram from ground truth



(c) "Road" histogram from our method

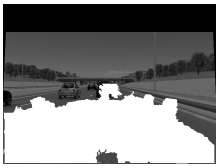


(e) "Road" histogram from FOLKI



(b) Segmented road from ground truth

(d) Segmented road from our method



(f) Segmented road from FOLKI

FIGURE 3.17: Segmented road from two methods and their comparisons with ground truth

$$Err = \frac{\sum |I_2(x,y) - I_1(x+u_x, y+u_y)|}{\text{pixel numbers belong to road}} \tag{3.36}$$

Using synthetic data, our method when combined with MDP gives the most accurate results. As a consequence, we chose to implement it and evaluate it with real images. Interpolation errors using different initial speed estimates are shown by the red curve in Figure 3.18. Best results (the smallest error) are obtained when the initial speed is about 13.5 m/s. From Figure 3.18, we can see obviously that our results depend on the initial speed estimate. We show that another iteration using the resulting speed as an initial guess allows to enhance the results and to reach the correct flow (see the blue flash in Figure 3.18). It means that our method leads to flow convergence even if the initial guess is far from the correct flow.

Results comparison with other existing methods are shown in Table 3.3 : our method gives the smallest interpolation error. When comparing with hierarchical Horn&Schunck (rank 2 from best to worst), their flow vectors are represented by red lines in Figure 3.19. Our method compute first the dominate flow using Equation 3.31 from the initial speed (Figure 3.19(a)) then interpolates to determine an intermediate image in order to compensate the large displacement flow. The remaining part is estimated by MDP (Figure 3.19(b)). The final resulting addition of the two flows is shown in Figure 3.19(c). One can see that it is more accurate than the Horn&Schunck results (Figure 3.19(d)), especially for the pixels corresponding to the road in front of the car. It is mainly due to the smoothness constraint in the Horn&Schunck approach that leads to false motion estimation on homogeneous regions like the road. This kind of inaccuracy is limited to very small regions in our approach when we combine it with MDP (Figure 3.19(b)). The dominate flow since it is computed directly by the flow equation is correct. As a consequence, it gives better results then the direct Horn&Schunck method.

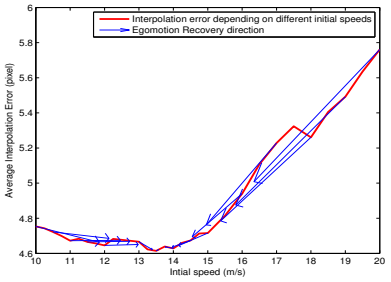
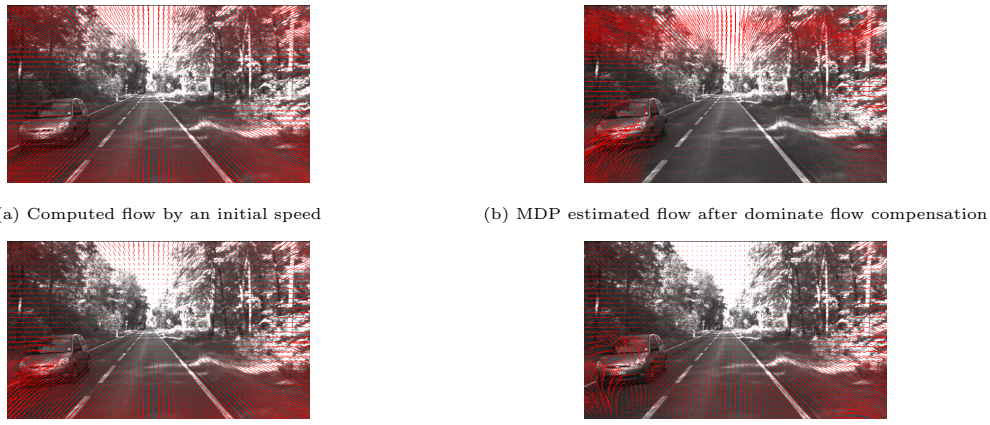


FIGURE 3.18: Average interpolation error depending on different initial speeds

TABLE 3.3: Interpolation error comparison of different techniques

	AIE
H&S	5.52088
MDP	18.7469
FOLKI	15.5334
MMC-MDP	4.61419



(a) Computed flow by an initial speed

(b) MDP estimated flow after dominate flow compensation

(c) The flow results computed by our method

(d) The flow results computed by Horn&Schunck method

FIGURE 3.19: Flow vectors of MMC-MDP and H&S

Here we also use "c-velocity" for road segmentation in order to evaluate optical flow accuracy. Since a road (considered as horizontal plane) can be transformed into a straight line in the c-velocity space (w, c) (see Equation 3.35) and we can extract it by Hough Transform. That means the road pixels will finally turn into a peak in the line (or the line slope k) histogram. And the more accurate the road optical flow is, the finer the peak will be. As a consequence, we applied our method combined with MDP flow method into the c-velocity process. The results are compared with which uses a direct MDP flow in two different scenes (see Figure 3.20). The results from scene 1 are shown in Figure 3.21. Apparently MMC-MDP flow provided a better segmented road, and It gives a fine peak in the road histogram (see the peak marked by violet circle in Figure 3.21(c)). Almost the whole road are detected except the small region near the center of the image as their flow magnitudes are too small so as to be filtered (see Figure 3.21(e)). While from the direct MDP flow result, a large road region near the camera was not extracted (see Figure 3.21(f)). And their votes in the road histogram are concentrated in the red regions (see Figure 3.21(d)) so they are not detected. The results from scene 2 are shown in Figure 3.22. Compared with scene 1, the camera movement in this scene is slightly larger. But the direct MDP method provides a really improper flow, especially in the center lane (see Figure 3.22(b)). Hence, most road pixels are not detected. While our method provides still a good flow (see Figure 3.22(a)), a fine peak (see Figure 3.22(c)) and a well segmented road (see Figure 3.22(e)).

3.6 Conclusion

As 2D motion field information is very important for the "c-velocity" approach, the estimation of the 2D motion is then an important preliminary step. In this chapter, we

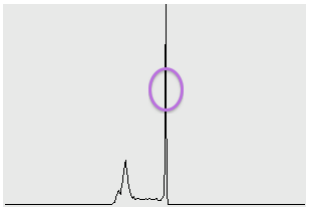
(a) scene 1

(b) scene 2

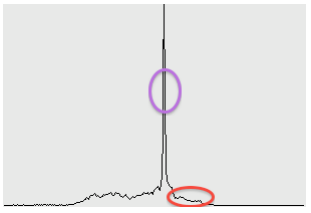
FIGURE 3.20: two scenes from Kitti database

(a) MMC-MDP flow result

(b) MDP flow result



(c) Normalized histogram of road with "c-velocity" by MMC-MDP flow



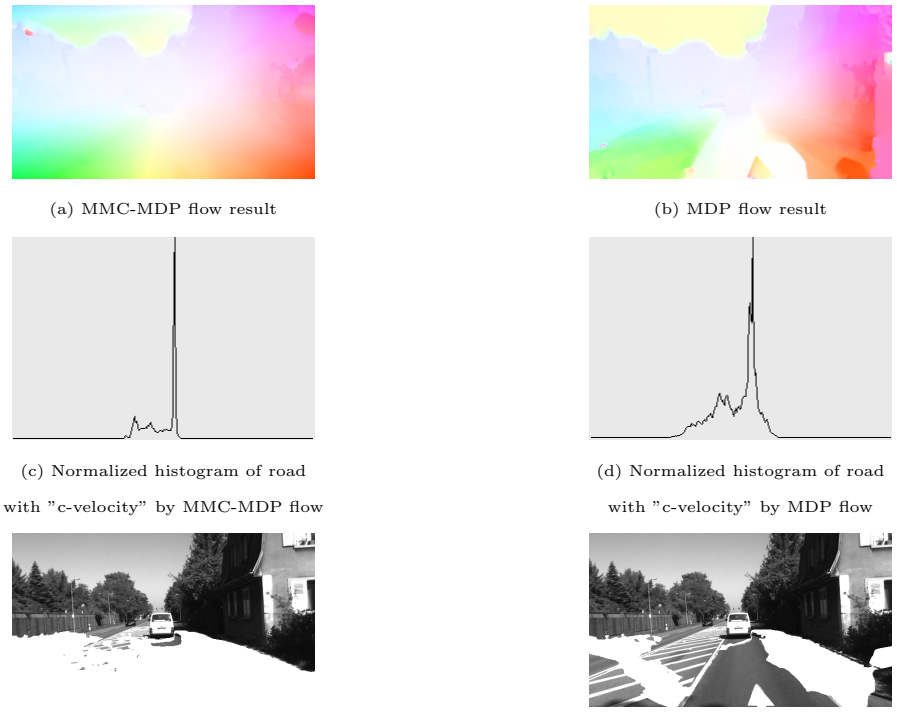
(d) Normalized histogram of road with "c-velocity" by MDP flow

(e) Road segmentation results (white region) by MMC-MDP flow

(f) Road segmentation results (white region) by MDP flow

FIGURE 3.21: road segmentation results of scene 1 by MMC-MDP and MDP methods

gave a presentation on existing motion estimation methods and compare them. Popular benchmarks to evaluate these methods are also presented. From the results of these state-of-the-art estimation methods, we found that none of them can be used for estimating large motion and on texture regions in the context of ADAS. So, we propose a new model-based optical flow method that exploits available a priori knowledge about the vehicle velocity. We focused on estimating the motion of horizontal planes which correspond to the road region in this kind of applications. The motion of this particular region is difficult to estimate with classical optical flow approaches since it is homogeneous and has a large relative displacement due to the moving vehicle. Our approach combines other sensors motion information to compute an estimate of the 2D velocity by using 3D/2D motion projection equations. After motion compensation, the remaining part of the velocity can be easily estimated by a classical optical flow method. We test our method on synthetic image sequences knowing the ground truth optical flow. Several comparison



(a) MMC-MDP flow result

(b) MDP flow result

(c) Normalized histogram of road with "c-velocity" by MMC-MDP flow

(d) Normalized histogram of road with "c-velocity" by MDP flow

(e) Road segmentation results (white region) by MMC-MDP flow

(f) Road segmentation results (white region) by MDP flow

FIGURE 3.22: comparison results of scene 2 between MMC-MDP and MDP methods

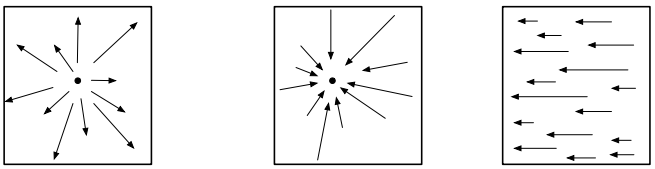
criteria were defined, including the traditional angular error and the endpoint error. We define a new comparison criteria based on the c-velocity method which provides 3D plane detection. We show how the optical flow is improved using our approach that was evaluated also on real image sequences. Moreover, the suitable convergence property of our algorithm allows to design an iterative motion estimation method that will be evaluated in a further study.

4 3D Translational Motion Estimation

Contents

4.1 Introduction

Egomotion is very important information in many applications of intelligent vehicle systems like obstacle detection, time-to-contact (TTC) estimation. Optical flow is a presentation of the egomotion inthe image sequence. From Equation 2.1, egomotion parameters can be computed from image movement information. In the case of purely translational egomotion [1], optical flow vectors form a radial pattern emanating from a center point, called Focus Of Expansion (FOE) (see Figure 4.1(a)). When the motion is backward (see Figure 4.1(b)), this point is called Focus Of Contraction (FOC). In the case of $T_Z = 0$, the radial pattern becomes parallel vectors (see Figure 4.1(c)). In the above situations with only translational egomotion, optical flow estimation can be reduced to that of locating FOE position [131–134]. Apparently FOE localization is useful to understand optical flow and egomotion, and it is also required for our "c-velocity" approach. So this chapter will focus on the FOE estimation.



(a) When moving forward (b) When moving backward (a) When $T_Z = 0$

FIGURE 4.1: The three types of optical flow pattern generated by translational motion

4.2 Background

FOE, as it contains egomotion information, plays a very important role in many vision applications such as 3D reconstruction of environment, visual attention, collision avoidance, image segmentation, etc. Especially for visual navigation applications, FOE is widely used to estimate the time-to-contact (TTC) [135–137]. In fact, FOE as a strong cue for visual attention has been presented in [138]. Moreover, FOE is also helpful for motion segmentation to distinguish static environments from moving objects as the FOE position only depends on the motion vectors of static objects.

Since Gibson [139] popularized the usage of optical flow for explaining human navigation behavior, computer vision researchers have devoted much work to estimating FOE from optical flow. A variety of estimation methods have been proposed. These methods can be generally classified into direct methods and optical flow based methods. Using the property that optical flow vectors due to the translational egomotion of the camera

1. Vehicles move forward in majority situations except for the swerves

converge to FOE, many conventional methods take advantages of optical flow for FOE estimation. Since these methods rely on optical flow information, they are suffered from the errors from the estimation of optical flow. Therefore, people seek to estimation FOE immediately from image information and avoid optical flow estimation. Such type of method is called direct method [132–134].

Optical flow based methods are further divided into two categories according to flow densities. In the early years, the estimation methods for dense optical flow are not robust and efficient. Optical flow is well estimated just on some interest points. For example, the Harris corner detector [140] is widely used to select some pixels with relevant information and flow estimation on these points is more robust. Early FOE estimation methods are based on these sparse flow vectors. These methods using matching and correspondence of feature points are then called discrete methods. However, these methods, as they use only local information, are not robust.

Then after the emergence of efficient optical flow algorithms, dense optical flow are introduced. Using dense flow for FOE estimation is that we called continuous method. Such method is more robust since global information (dense flow) is used for the estimation. But computational cost is higher, especially when we use least square minimization methods to deal with this over-determined problem [141]. As there exist many outliers (errors) in optical flow vectors and least square minimization methods are sensible to noise, FOE estimation will be severely affected. Therefore, algorithms such as RANSAC (**RAN**dom **S**ample **C**onsensus) [142] can be applied to deal with outliers [143]. Another way to cope with outliers is voting strategies. For example, Authors of [144] use a voting scheme to locate FOE after separating the camera rotation. Similar manner is also used in [37] by employing a matching filter to exploit the sign change of optical flow around the FOE point to detect it.

Many direct FOE estimation methods are also proposed. Even one of them has ever been implemented in hardware [145]. For instance, [146] uses brightness change by imposing the constraint that the camera is in front of the image scene. The method used in [147] is interesting : it considers some FOE candidates at first, then optical flow vectors are obtained from the hypothesis of the FOE position and the global constraint of optical flow. Using these optical flow vectors, a warping image is built. The differences between the warped image and the real image are used to evaluate FOE.

In our case, FOE estimation is a preliminary step of the "c-velocity" approach. Computational cost and accuracy are two important factors to select a suitable estimation algorithm. Since these two parameters are usually incompatible, we should find a method that gives a good compromise. Here, we designed three different FOE estimation methods and tested them on real images.

The first method makes use of optical flow to realize a voting process to choose an adequate FOE position from several candidates. Such voting approach is resistant to optical flow noise, but it takes a lot of time to make every pixel in the image vote FOE position.

In the second method, FOE estimation is transformed into linear least square minimization. Then Singular Value Decomposition (SVD) is used to solve this linear equation system. As there exist many outliers on optical flow vectors, and the SVD method is very sensible to outliers, we firstly filter the flow and select the highly confident flow vectors. Then RANSAC will be used to deal with outliers.

The last one is an interesting method that using the "c-velocity" concept. From the "c-velocity" approach in Chapter 2, scene structures are estimated using flow norm and FOE information to construct "c-velocity" images. Here we do it inversely : if we know scene structures and optical flow information, we can locate FOE. So this method is also called inverse "c-velocity" method.

4.3 Voting based FOE estimation

4.3.1 Principle introduction

Under a purely translational moving camera, optical flow vector due to the egomotion can be expressed by Equation 4.1 :

$$\begin{cases} u = \frac{xT_Z - fT_X}{Z} = \frac{T_Z}{Z}(x - f\frac{T_X}{T_Z}) = \frac{T_Z}{Z}(x - x_{FOE}) \\ v = \frac{yT_Z - fT_Y}{Z} = \frac{T_Z}{Z}(y - f\frac{T_Y}{T_Z}) = \frac{T_Z}{Z}(y - y_{FOE}) \end{cases} \tag{4.1}$$

If we divide the two flow components, we have the equation like

$$\frac{u}{v} = \frac{x - x_{FOE}}{y - y_{FOE}} \tag{4.2}$$

From Equation 4.2, motion vectors emanate from the FOE point. In other words, FOE is located on the reverse extension line of the motion vector and all these reverse extension lines intersect in FOE. So when each motion vector votes for all the points located on its reverse extension lines as FOE candidates. The real FOE point should have the most votes from all the motion vectors. This is the basic idea of the proposed voting method. The accumulation framework is resistant to the noise from optical flow estimation.

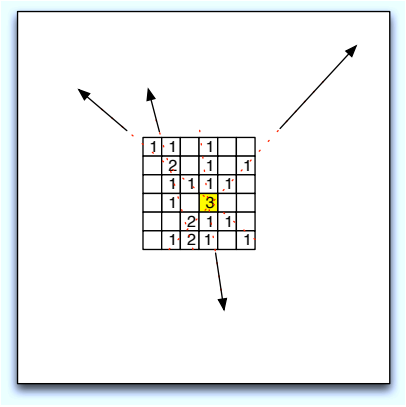


FIGURE 4.2: Voting process for FOE estimation

Here we use a simple example to understand how it works in detail. In Figure 4.2, only four flow vectors participate in the vote. And a candidate region is already defined at first. This region is used to restrain of a region for real FOE. It is usually a rectangle window located in the image center. When the host vehicle is moving straightly forward and the camera is well rectified to be parallel to the road plane ($T_X = T_Y = 0$), FOE should be exactly in the image center. Through the above conditions are too strict, in most cases, T_X and T_Y are not equal to zero but they are still small values. The FOE position is therefore not far from the image center. So the FOE region is defined in the image center. In the example, it is a square window with 6×6 candidate pixels. Once the FOE region is determined, we start the vote process : for each flow vector, the inverse extension line is drawn to cross the candidate region, pixels located on this line are voted. The votes are accumulated until all the vectors are processed. The pixel with the largest number of votes is considered as the location of FOE (the yellow pixel in Figure 4.2).

4.3.2 Results on synthetic optical flow images

In this section, we create synthetic optical flow vectors to estimate the FOE position. Figure 4.3 shows the synthetic flow vectors computed directly from Equation 4.1. The image size is 640×480 and the real FOE position is (320, 250).

It is meaningless to estimate the FOE using these synthetic flows, as the real FOE position will be estimated with no doubt. In fact, for all the optical flow based FOE estimation methods, difficulties lie on the ability to eliminate the "false" flow vectors. These "false" flow vectors can be generally classified into two categories :

— The badly estimated optical flow from an estimation method of optical flow (ex. optical flows of the homogeneous regions or due to the large egomotion)

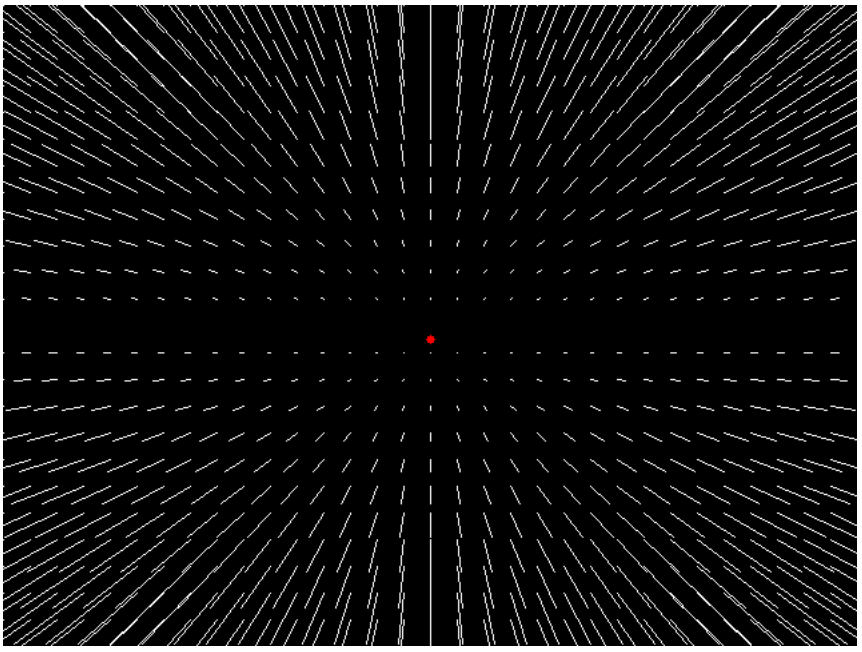


FIGURE 4.3: A synthetic optical flow example with FOE colored as red point

— Optical flow of moving objects

"False" flow from optical flow estimation method Although there are thousands of optical flow estimation methods, none of them can exploit the true flow vectors. Optical flow noises always exist on the whole image, especially for the homogeneous regions where little information can be used for detection. For the small motion regions, the relative error of optical flow is also very large. To study the influence of these noises on FOE estimation. We add the synthetic vectors (see Figure 4.3) with Gaussian white noises. These noises are added either on the component u or on the component v. For example, Figure 4.4 is an example of the FOE estimation based on the noised synthetic vectors. In this example, a Gaussian white noise with a variance of 5 pixels is added on the components v of all flow vectors (see the white flow vectors). Then the FOE position is estimated as the blue point. When we compare it with the real FOE (the red point), the shift is very small and only on the y direction.

$$\begin{aligned} u &= u_0 + \mathcal{N}(0, \delta) \\ v &= v_0 + \mathcal{N}(0, \delta) \end{aligned} \tag{4.3}$$

To study the robustness of this method to the Gaussian noise, we now change the noise variance δ and the estimation results are shown in Figure 4.5. The blue curve presents the changes of FOE error as a function of variance of Gaussian noise on u. The FOE error is considered as the euclidean distance between the real FOE point and the estimated

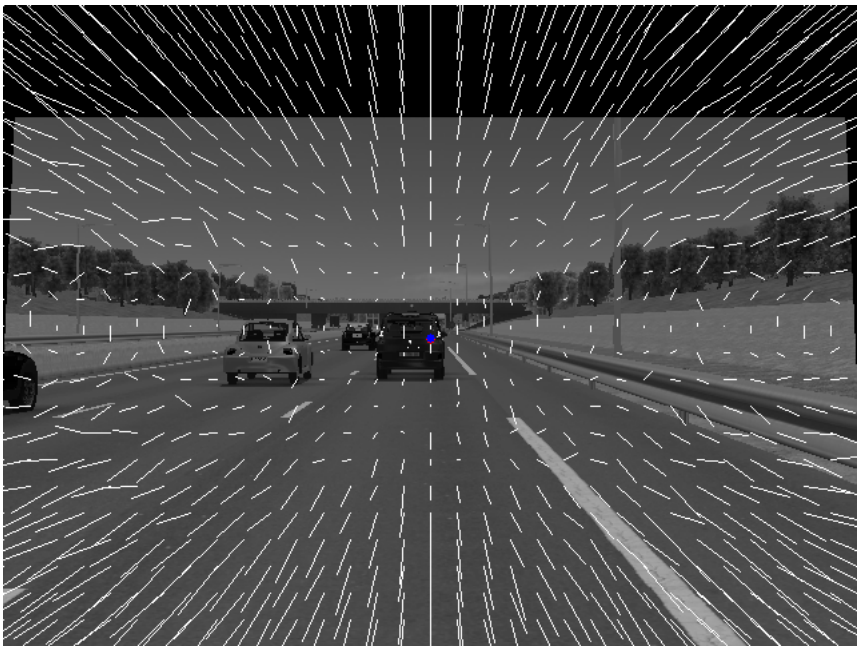


FIGURE 4.4: Synthetic optical flow vectors with a Gaussian white noise ($\delta = 5$) on the component of v

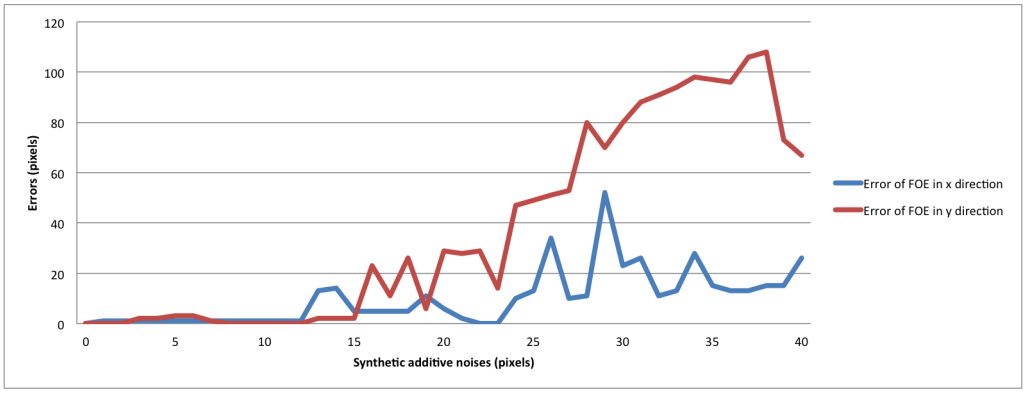


FIGURE 4.5: Impact of optical flow on FOE location

FOE point. The influences of the Gaussian optical flow noise on component v is shown by the red curve. From the curves, the FOE error is only 1 or 2 pixels when $\delta < 13$ pixels. This means that the voting based method is very robust to optical flow noise.

"False" flow from moving objects As this method is used to estimate FOE for the "c-velocity" approach in an urban road situation, and the moving objects are usually pedestrians and vehicles, optical flow of these moving objects are not only due to ego-motion. So these flow vectors are considered as "false" flow when we estimate the FOE position. In fact, the impacts of these flow vectors can be easily avoided by this voting based method. As all the moving objects have their proper FOE point, in the case of crossing pedestrians on both sides of the road, their flow vectors also converge to their own FOE points. But these points are usually located on the both side of the image,

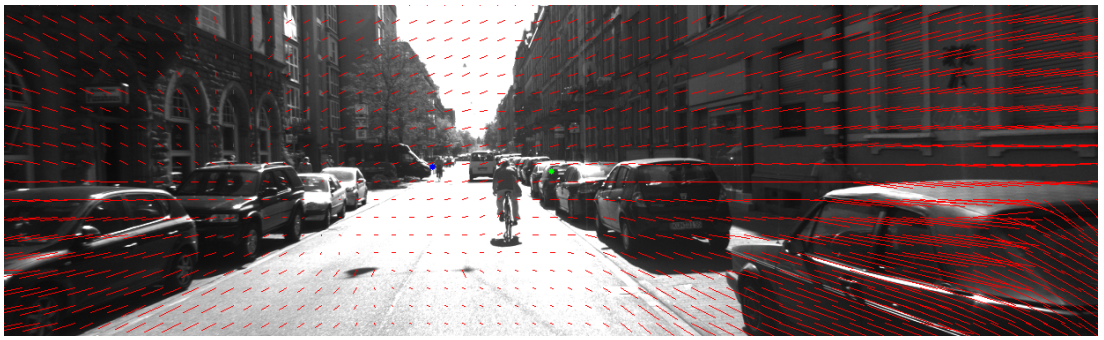


FIGURE 4.6: An example of the FOE estimation on real images

so they will be eliminated by the candidate region locating in the image center. Even if their FOE points are located in the candidate region such as the moving vehicles, the votes from these objects are too small compared with those from static pixels. So the voting strategy will eliminate these points.

4.3.3 Results on real images

This method is also tested on real images. Since we have no ground truth, we can not quantitatively evaluate this method. But we can still give a qualitative evaluation on the results. An estimation example is shown in Figure 4.6. The image is from KITTI database [116]. The used optical flow estimation method is from [148]. This method can deal with very large motion by integrating rich descriptors into the variational optical flow settings, which is suitable for our case as the egomotion from the camera is very large. From optical flow information (the red vectors), we can see that there is a change of the direction to the left. So we find that the estimated FOE (the blue point) is at the left part of the image (the green point is the image center) and the flow vectors are pointing to this point judging from naked eyes.

Another example is also from KITTI database (see Figure 4.7). In this figure, the vehicle is moving forward, and the estimated FOE position (the blue point) is very close to the image center (the green point).

The voting results of this example is also shown in Figure 4.8, where different colors represent different vote numbers. The black region is the place that the vote number is smaller than 1% votes (4657 pixels from 1242×375 pixels). Then the regions from the second fewer votes to the most votes are represented by blue, cyan, yellow and red. The estimated FOE is located as the intersection of two white line in the red region. This FOE position is voted by 4% pixels.

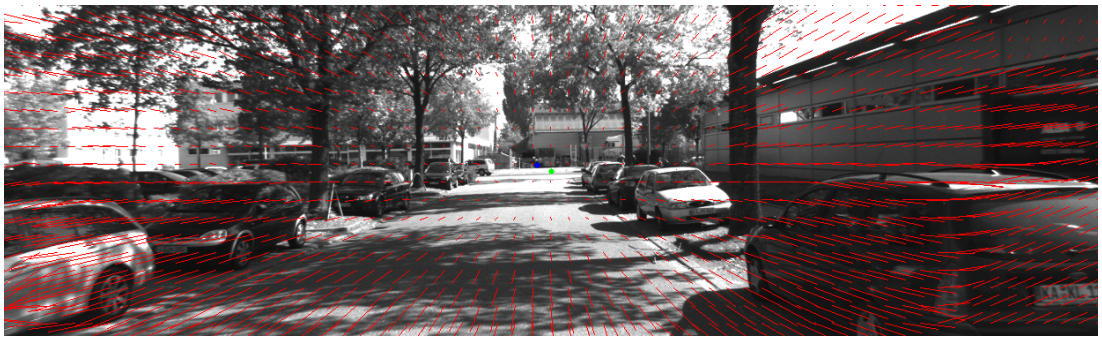
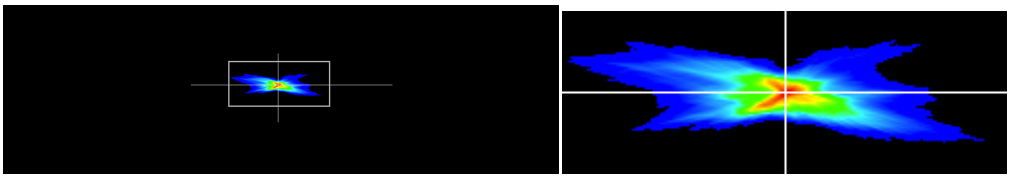


FIGURE 4.7: Another example of the FOE estimation on real images



(a) Voting results represented by the different colors (b) A zoom on the mostly voted regions

FIGURE 4.8: Color based voting space for FOE determination

Although this method is very robust and insensitive to the optical flow noise, the computation cost is also very high as all the pixels votes for more than one FOE candidates [2]. For the example shown in Figure 4.7, the image size is 1242375 pixels, and the candidate region size is 621187 pixels. The computational time is about 14 seconds. In fact, the computation cost can be reduced when we select a smaller candidate region. But a small candidate region risks to missed the real FOE. The choice of candidate region should be therefore a compromise between the computational time and the accuracy. In this method, as the candidates are pixels, the precision of the FOE position is therefore pixel-wise.

4.4 Least square FOE estimation with SVD solution

4.4.1 Principle introduction

From Equation 4.1, FOE can be computed from a linear equation :

$$\begin{bmatrix} v & -u \end{bmatrix} \begin{bmatrix} x_{FOE} \\ y_{FOE} \end{bmatrix} = xv - yu \tag{4.4}$$

2. *A flow vector votes for all the pixels located on its reverse extension line*

There are only two unknowns (x_{FOE}, y_{FOE}), but as many equations as velocity vectors in the image. Each optical flow vector (u_i, v_i) can write an Equation 4.4, then the linear equation system is written by

$$\mathbf{A}\begin{bmatrix} x_{FOE} \\ y_{FOE} \end{bmatrix} = \mathbf{B} \tag{4.5}$$

With

$$\mathbf{A} = \begin{bmatrix} v_1 & v_2 & \dots & v_n \\ -u_1 & -u_2 & \dots & -u_n \end{bmatrix}^T$$
$$\mathbf{B} = \begin{bmatrix} x_1v_1 - y_1u_1 & x_2v_2 - y_2u_2 & \dots & x_nv_n - y_nu_n \end{bmatrix}^T$$

Such an over-determined system can be solved by linear least square methods :

$$\begin{bmatrix} x_{FOE} \\ y_{FOE} \end{bmatrix} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B} \tag{4.6}$$

Since this is a huge system with a great number of equations [3], the inverse of matrix $\mathbf{A}$ can be solved by Singular Value Decomposition (SVD). Although SVD solution can accelerate computation, the results are very sensitive to the "false" flow vectors. We know that the "false" flow vectors that are either from moving objects or from the bad estimations should be eliminated as outliers. An efficient method to deal with these outliers is RANSAC algorithm [142]. It is an iterative method to estimate model parameters from a set of observed data which contains outliers. The algorithm is illustrated in Algorithm 3. So this robust algorithm is also used for the proposed methods. The estimation method is shown in Algorithm 3. It is based on RANSAC algorithm. The computation of FOE from the subset pixels (function *Resolve*) is from Equation 4.6. The error measurement of a model (FOE) is computed by

$$\varepsilon^2 = \sum (vx_{FOE} - uy_{FOE} - (xv - yu))^2 \tag{4.7}$$

4.4.2 RANSAC algorithm

Selection of RANSAC parameters The first step of RANSAC is to select the sample number m for the computation of FOE and the iterative number n. A good

3. *Equation number is equal to the pixel number in the image*

Input: Pixel database $\Re$, N_iterations, Threshold, ε, N_mini_samples
Output: Inliers, FOE

```
1  begin
2      ε = ∞ ;
3      iterations = 0 ;
4      Inliers = {∅} ;
5      while iteration < N_iterations do
6          samples = RandomlyChooseSamples(ℜ, 2) ;
7          FOE_hypotheses = Resolve(samples) ;
8          Inliers_hypotheses = {∅} ;
9          for Each pixel P(x, y) ∈ ℜ do
10             if NormalizedAngularError(FlowVector, Vector(FOE → P)) <
               Threshold then
11                 Inliers_hypotheses = Add(P) ;
12             end
13         end
14         if Size(Inliers_hypotheses) > N_mini_samples then
15             FOE_hypotheses = Resolve(Inliers) ;
16             Error_hypotheses = Error(FOE_hypotheses, Inliers) ;
17             if Error_hypotheses < ε then
18                 ε = Error_hypotheses ;
19                 FOE = FOE_hypotheses ;
20                 Inliers = Inliers_hypotheses ;
21             end
22         end
23     end
24 end
```

Algorithm 3: FOE estimation method with RANSAC

estimation requires that all the m samples should be inliers. We do n iterations, the probability P that at least one good estimation is obtained should be as large as possible. To compute this probability, we first define the probability of choosing an inlier from the database as w, and it is computed by

$$w = \frac{\text{number of inliers in database}}{\text{number of points in database}} \tag{4.8}$$

This value w is usually not known, but a rough value can be given from the statistics of the experiments. For example, if $w >= 10\%$, then we can compute the probability that at least one of the m points is an outlier (the case that a bad FOE will be estimated) is

$$1 - w^m \tag{4.9}$$

After n iterative estimations, P can be written by

Probability to get at least one good estimation	Iteration Numbers
0.9	230
0.99	458
0.999	687
0.9999	916

TABLE 4.1: Relationship Between the iteration number and the at least one good estimation probability

$$P = 1 - (1 - w^m)^n \tag{4.10}$$

We want a high probability P, so the sample number m should be as small as possible and the iteration number n should be as large as possible. In our case, at least two samples are needed to fix the FOE position and if we consider that $w = 10\%$, we can obtain the relation between the probability P with the iteration number n by Equation 4.11 and Table 4.1.

$$n = \frac{\log(1 - P)}{\log(1 - w^2)} \tag{4.11}$$

Bucketing When we use RANSAC algorithm for FOE estimation, the samples are randomly selected. But they can not be very close to each other as the FOE estimation from closing pixels will be highly unstable. However, a random selection method based on Bucking technique [149] can deal with this problem : The image is firstly divided into $b \times b$ buckets. To each buckets is attached a set of pixels and indirectly a set of flow vectors on these pixels. The two samples should be selected from different buckets. Although the flow vectors are calculated for each pixel, not all of them will be used for the estimation. For example, the flow vectors of sky pixels are theoretically zeros and can not be used. The small vectors should also be filtered as the FOE estimated from these pixels is not accurate. In this case, the pixel number in each buckets is not the same (see Figure 4.9), and random selection is not uniform for each pixels. In other words, a flow vector belonging to a bucket having fewer points has a higher probability to be selected. To balance this and make sure that each pixel has almost the same probability, the bucket with more pixels should have a higher probability to be selected. Then the following procedure is implemented : Consider that we have a total of L buckets, we divide the range [0 1] into L intervals such that the width of the i^{th} interval is equal to $\frac{n_i}{\sum n_i}$, where n_i is the number of flow vectors attached to the i^{th} bucket (see Figure 4.10). During the bucket selection, a number produced by a [0 1] uniform random number generator falling in the i^{th} interval implies that the i^{th} bucket is selected.

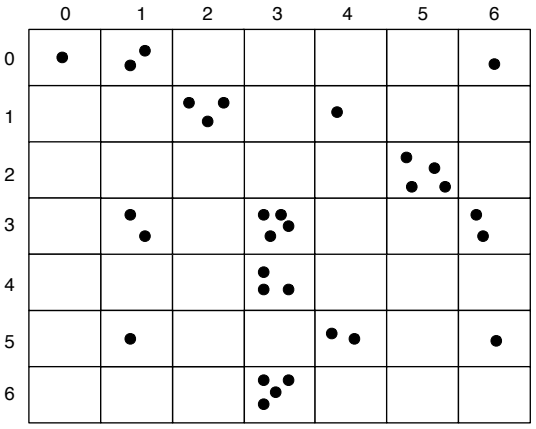


FIGURE 4.9: Illustration of Bucketings technique

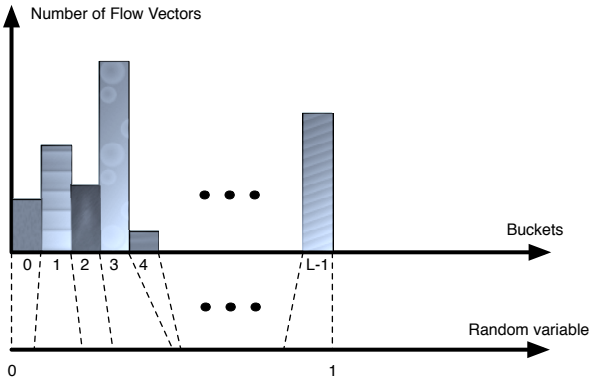


FIGURE 4.10: Interval and bucket mapping

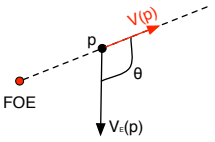


FIGURE 4.11: Inlier model estimation

Consensus model When a FOE model is estimated by the subset samples of optical flow vectors, we should define a measure to select the inliers. An accurate flow vector should be located on the extension line from FOE to the corresponding pixel. So we use angular error between the flow vector (see the black vector in Figure 4.11) and the ideal vector (see the red vector $\mathbf{V}(\mathbf{p})$ in Figure 4.11). The equation of this angular error is presented by Equation 4.12.

$$Err\_A(p) = \frac{v_x(p)(x(p) - x(FOE)) + v_y(p)(y(p) - y(FOE)) + 1}{\sqrt{1 + v_x(p)^2 + v_y(p)^2}\sqrt{1 + (x(p) - x(FOE))^2 + (y(p) - y(FOE))^2}} \tag{4.12}$$

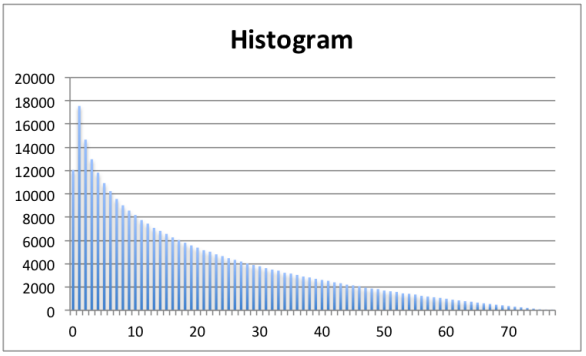


FIGURE 4.12: Histogram of the synthetic optical flow on $|u|$

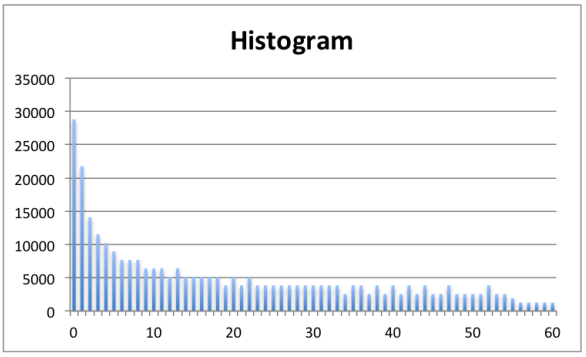


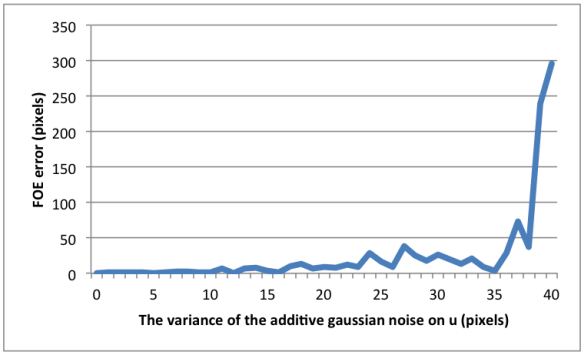
FIGURE 4.13: Histogram of the synthetic optical flow on $|v|$

4.4.3 Results on synthetic optical flow images

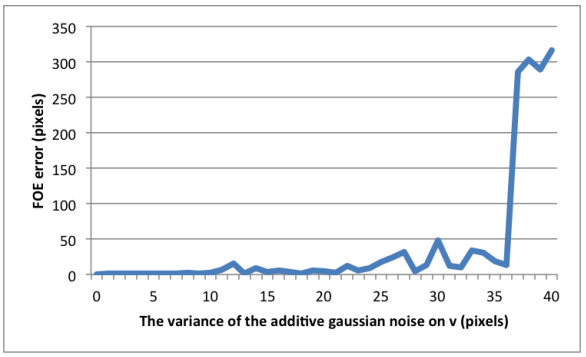
The estimation method is also tested on the synthetic optical flow image presented in Section 4.3.2. The original synthetic flow vectors are shown in Figure 4.3. Their histograms of synthetic flow vector components u and v are shown in Figure 4.12 and Figure 4.13. The maximum motion in the direction x is 77 pixels and the maximum motion in the direction y is 60 pixels.

To test the robustness, Gaussian white noises with different variances are added either on the flow component u or on the flow component v (eg. Figure 4.15). The estimation results are shown in Figure 4.14(a) and Figure 4.14(b).

When using the voting based method on the synthetic flow vectors, estimation error exists only on the same direction of the noised flow vectors. In other words, the noise on v leads to an estimation error only on y_{FOE}. While, the error exists on both components of FOE in this method. But the proposed method is very robust to the noise. From the results, FOE error is very small when the variance $\delta < 20$ pixels. Then the method failed when the variance $\delta > 39$ pixels for the component u or when the variance $\delta > 37$ pixels for the component v. If we go back to the synthetic flow histogram (see Figure 4.12 and Figure 4.13), 85% pixels in the image have a motion $|u|$ smaller than 39 pixels. And 81% pixels have a motion $|v|$ smaller than 37 pixels. We will lose all the inliers when



(a) FOE error changes along the augmentation of the variance of the additive Gaussian white noise on u



(b) FOE error changes along the augmentation of the variance of the additive Gaussian white noise on v

FIGURE 4.14: Estimation results on synthetic optical flow vectors with additive Gaussian white noise

the noises are too big. So RANSAC does not work well. As SVD solution is too sensible to noise, this estimation method will not work. Then, the estimation method will not work in this case. Figure 4.15 shows us the FOE estimation result when the variance of Gaussian noise is 5 pixels (see the white flow vectors). FOE is well estimated as there is only 1 pixel shift between the real FOE (the red point) and the computed FOE (the blue point).

4.4.4 Results on real images

We use the same real images that used for the voting based estimation method to test the proposed RANSAC based method. An estimation example is shown in Figure 4.16. From Figure 4.16(a), the estimated FOE is marked as the blue point, and the image center is marked as the green point. The result FOE in the x direction is very similar to that estimated by the voting based method (they are both located on the left of the image center), but the location in the y direction is different (see Figure 4.7 and Figure 4.16(a)).Although there is no ground truth for the comparison, we can still analyze the results through the inlier mask obtained by the RANSAC algorithm (see Figure 4.16(b)). The black regions are outliers. We can see that there are totally 5 big outlier regions (they are marked and numerated by the ellipses). If we find the corresponding regions

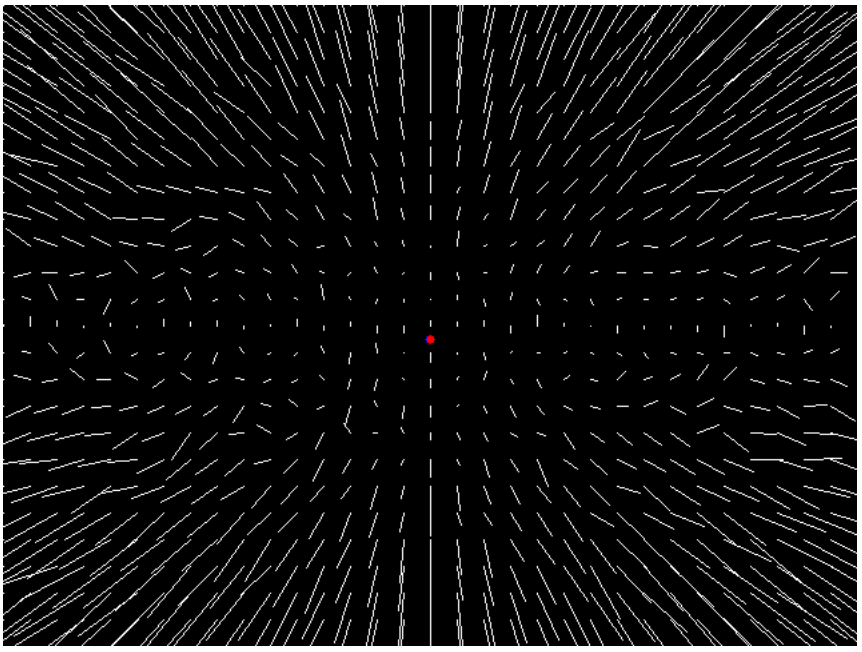


FIGURE 4.15: FOE estimation result from synthetic optical flow vectors with an additive Gaussian white noise ($\delta = 5$) on the component of v

in Figure 4.16(b) (they are also marked by the ellipses and numerated with the same order) : In region 1 (the center region), the flow vector is too small as they are very close to the FOE. So these vectors can not be used for the estimation. In region 2 (the left top region), the sky pixels have badly estimated flow vectors as they have no texture. So these pixels can not be used for the estimation. When looking at the flow vectors in region 3, 4 and 5, we find an evident difference both on the flow direction and the flow norm from their surrounding flow vectors. So they should also be considered as outliers. From the above discussion, the inliers are well estimated from the RANSAC algorithm. So we can conclude that the FOE from these inliers are also well estimated.

Another estimation example is shown in Figure 4.17 to compare with Figure 4.6. The estimated FOE positions (the blue point) from these two methods are very close to each other (they are both around the person on the bicycle). Contrast to the above example, the different in this case is mainly reflected on the x direction.

4.5 Inverse "c-velocity" FOE estimation

From the "c-velocity" concept in Chapter 2, an object (eg. Obstacle, Road, Buildings) in urban road situations can be modeled as a 3D plane. And its representation is a straight line when we project it into the "c-velocity" image (see Figure 4.18). Then [64] proposed a method based on this concept to extract the scene structure from optical flow

(a) FOE estimation results from the proposed method

(b) Inlier mask from the RANSAC algorithm

FIGURE 4.16: An example of FOE estimation results on the real image

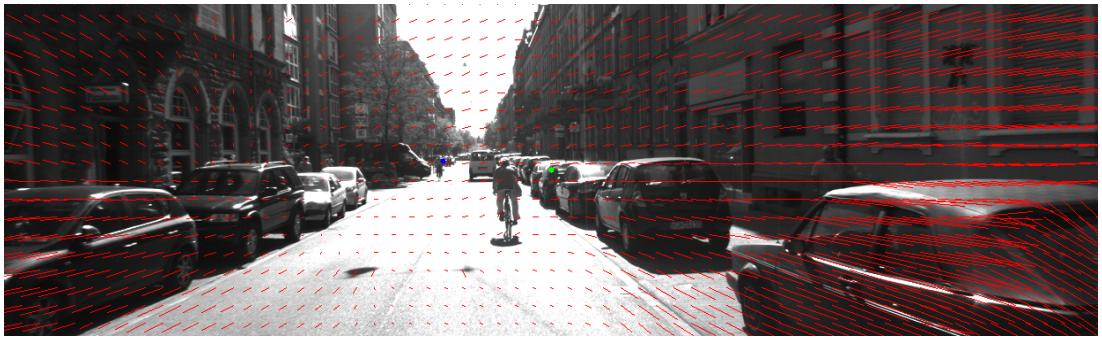


FIGURE 4.17: Another example of FOE estimation results on the real image

and FOE information [4]. Since the egomotion (FOE information) is as important as the scene geometry, another method based on the "c-velocity" concept was also proposed [150] to estimate FOE information from scene information. This method, as it is totally the inverse process (see Figure 4.19) of that proposed in [64], is called the inverse "c-velocity" FOE estimation method. For the simplicity, in the rest of this thesis report, we call this FOE estimation method for short as **inverse** "c-velocity" approach, and the original scene structure extraction method is called **direct** "c-velocity" approach.

The basis idea of this inverse "c-velocity" approach is to estimate FOE information by optimizing representation of the scene planes in the "c-velocity" images. When we project a scene plane into the corresponding "c-velocity" image, the representation is a straight

4. *FOE is required to compute the "c-value"*

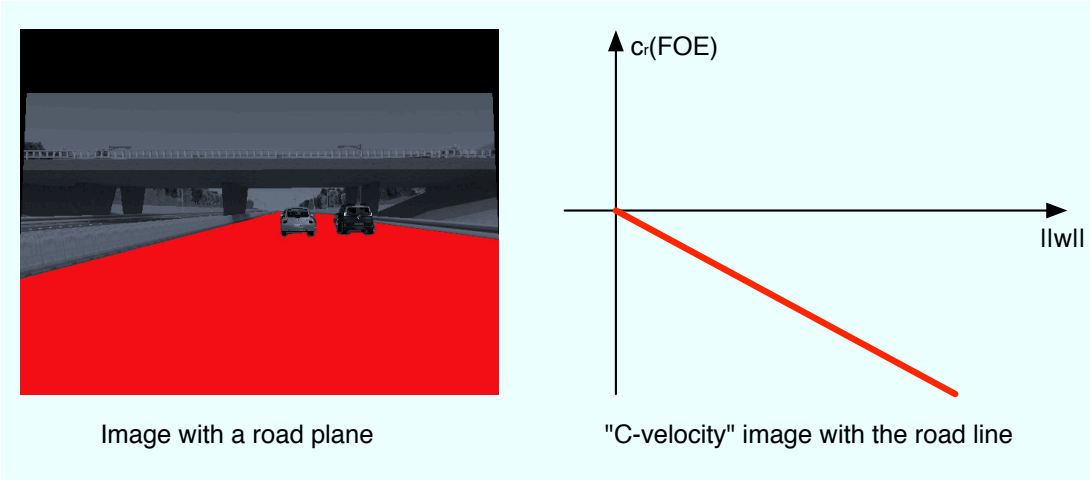


FIGURE 4.18: An schema expression of the "c-velocity" concept

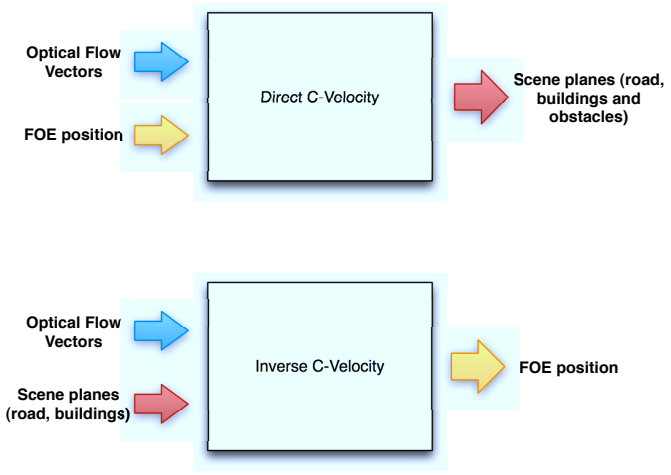


FIGURE 4.19: Schema blocks comparison of direct "c-velocity" and inverse "c-velocity"

and fine line with correct FOE information. But if the used FOE is badly estimated, the "line" will deform. And [150] concluded that the "line" is finer when the used FOE is closer to the real FOE. According to [150], the inverse "c-velocity" approach is then focused on the two issues. The first one is to extract several objects (planes) [5]. The second but more important issue is to find a way to quantify the appearance of objects projected "lines" by exhibiting a metric that reflects the distance between a supposed FOE and the real FOE. [150] proposed to extract road planes with "v-disparity" approach and to extract building planes with generalized Hough Transform. Then it stops after defining and proving the validity of the "line" dispersion.

In this section, we will propose more methods to extract scene structures and a complete FOE estimation process is also proposed by minimizing the "line" dispersion function.

5. *At least one object should be extracted*

4.5.1 Scene Structure Extraction methods

The first step for the inverse "c-velocity" approach is to extract scene structures. The idea is to identify the co-planar pixels in the image. Neither the plane equation nor the distance from plane to sensor are required. We should only provide a set of pixels labeled with different types of planes. In the direct "c-velocity" approach, three types of planes (buildings, road and obstacles) are considered. Since obstacles have their own motions and can not be used to estimate FOE only due to the egomotion, we use only static objects like roads and buildings. The buildings are considered as lateral vertical planes and the road is assumed to be horizontal plane.

Several systems can be used to extract these planes. For example LIDAR based sensor system [151] directly measures 3D information of the scene to extract the planes. The stereo-system [152] can estimate depth map to help the plane extraction. Then the monocular system - direct "c-velocity" framework [64] is designed specifically for the plane extraction.

LIDAR System for object plane extraction LIDAR is widely used by autonomous vehicles for obstacle detection and avoidance to navigate safely through environment as it provides robust 3D information on the environment. Then this 3D information can be mapped into the camera images if the LIDAR system is calibrated with camera. This calibrated system is exactly what we need to extract objects, since the 3D positions of all the pixels in the image are available. The road plane is just defined for the pixels with $Y = constant$, and a building plane equation is presented by $X = constant$.

This method is robust but the sensor is expensive and the calibration between sensor and camera is also a tough task. Fortunately, KITTI benchmark provides all the image data, LIDAR data and the calibration information. So we can easily extract different planes according to the 3D position of points, and project these point clouds into the image to obtain the corresponding pixel position.

Two examples are shown in Figure 4.20 and Figure 4.21. In Figure 4.20, a road plane is detected but not all the road pixels are detected (only the color pixels are extracted). In fact it is not necessary to extract all the road pixels but only to make sure that the selected pixels are from the same plane. In Figure 4.21, the left building pixels are extracted as well as the pixels not belong to the building (the circled pixels) since they are from the same lateral plane $X = \text{constant}$.

Stereo system for objects planes extraction Stereo systems can create disparity map and depth information can be computed directly from disparity map. Once depth

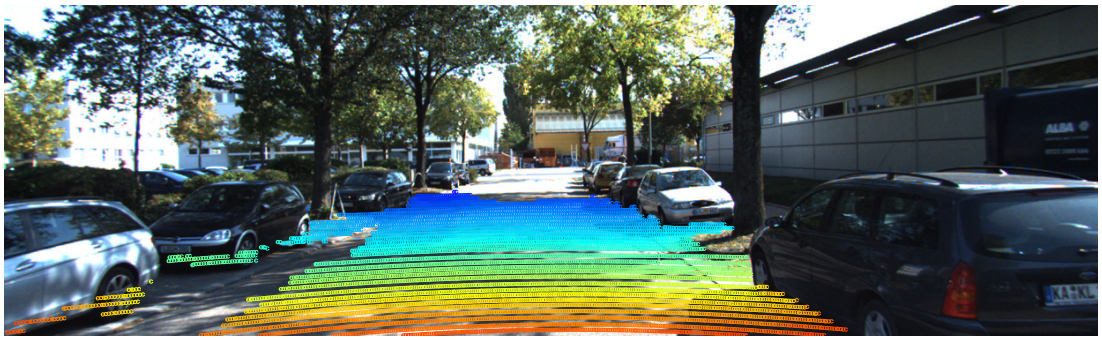


FIGURE 4.20: Detected road pixels from LIDAR data of KITTI benchmark

FIGURE 4.21: Detected building pixels from LIDAR data of KITTI benchmark

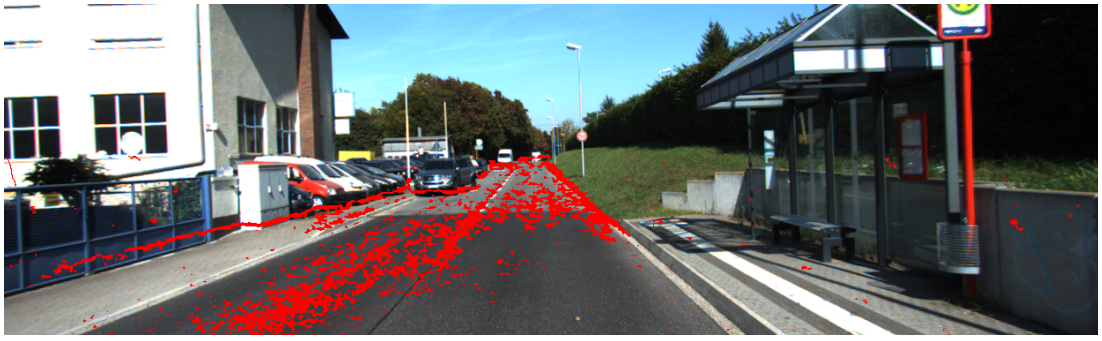


FIGURE 4.22: Detected road (red regions) from the "v-disparity" approach

information is known, the 3D position of each pixel is also available from their 2D positions. In our case, the road plane extraction is done by the "v-disparity" approach. This technique relies on the accumulation of disparity values along horizontal lines of the disparity map. In the cumulative image ("v-disparity" image), the road is presented as a straight line, which is easily detectable from classic Hough Transform. Figure 4.22 is an example of the road region extraction from the "v-disparity" approach. In this example, the disparity map is computed using a semi-global block matching method [58]. Since we consider the road as planar surface, a 2D Hough Transform is used to extract the "line" in the "v-disparity" image. But sometimes, the road is more like a succession of parts of planes. In this case, the road projection in the "v-disparity" image is a piece-wise linear curve. The extraction method of this piece-wise linear curve can be found in [2].

FIGURE 4.23: Detected building plane pixel by Direct "c-velocity"

An extension of this method - the "u-disparity" approach can be used for the lateral planes (buildings) extraction. In [150], the authors used a generalized Hough transform to extract directly the building planes (vertical planes). A vertical plane can be presented by two parameters : the distance to the origin (ρ) and the angle between the optic axis and the plane (θ). From this 2D vote space, we can easily extract the image pixels that belongs to the buildings ($\rho = 0$).

Direct "c-velocity" system for objects planes extraction We recur to the direct "c-velocity" approach as the purpose of this method is scene structure extraction. While this method requires FOE for the scene extraction, if we combine it with the inverse "c-velocity" method, it seems to become a chicken-egg problem. In fact, this combination allows to fulfill an iterative strategy to improve FOE estimation and further modify scene extraction. The procedure of this combination is like this : An initial FOE is estimated to employ the direct "c-velocity" approach, then a rough detection of different object planes is obtained. From this rough extraction of the objects, the inverse "c-velocity" approach is applied to estimate a new FOE. This FOE can be proved to be better than the initial one, and will be reused for the next iteration. Here we consider the direct "c-velocity" approach as only the pre-step of the FOE estimation. Figure 4.23 is a plane detecting example by the direct "c-velocity" with optical flow estimated by FOLKI. From the last chapter, we know that the most optical flow estimation methods can not well estimated the road because of large egomotion as well as a lack of texture. Therefore, the road plane is not detected. But the buildings on both sides of the road are detected and can be used for FOE estimation. While using our proposed method, the road flow is well estimated and the road is segmented by the direct "c-velocity" (see Figure 3.22(e)(f)). As our method only proposed the motion estimation for road model, the motion of other objects likes buildings or obstacles can not be correctly estimated.

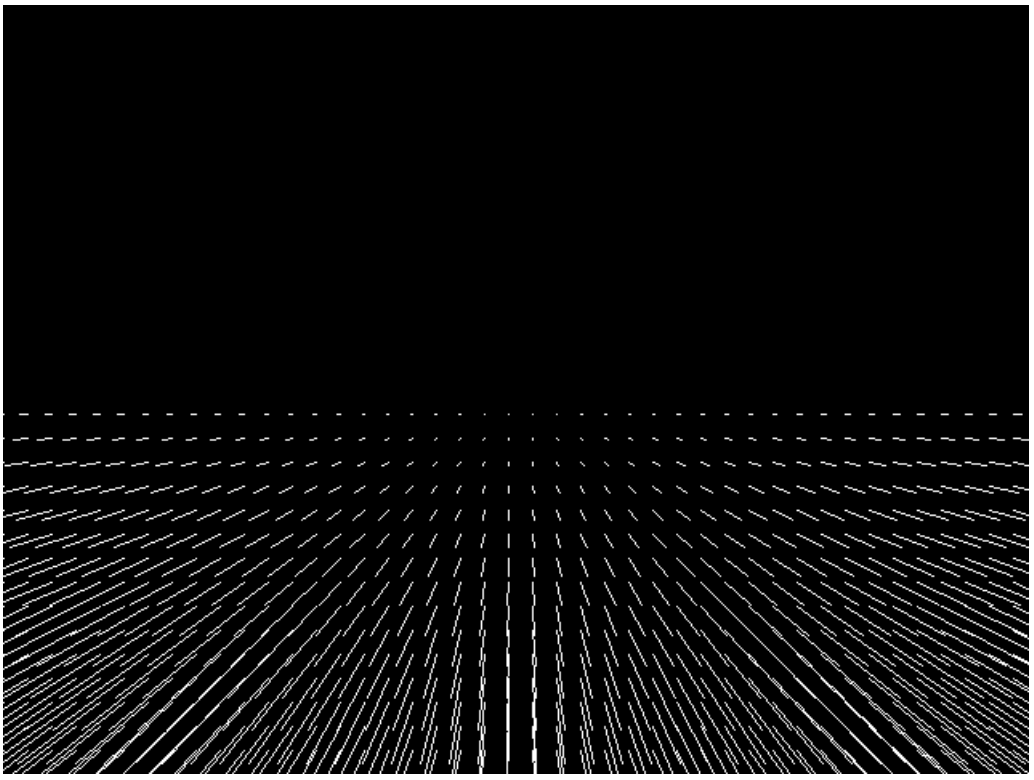
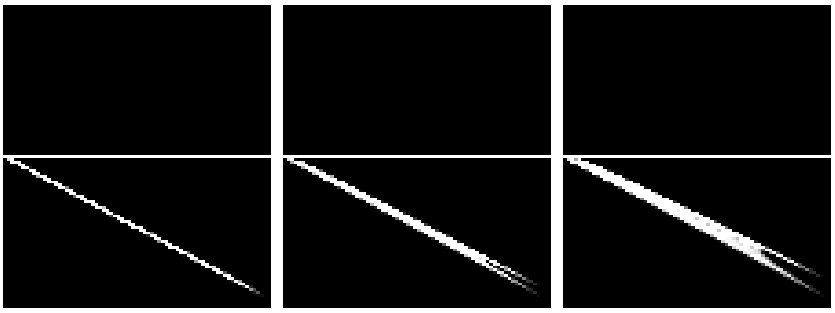


FIGURE 4.24: A synthetical optical flow example of a horizontal plane

4.5.2 FOE localization

From the above presentation, scene structures can be extracted by any of the three proposed methods. At least one object plane should be extracted (road or buildings). Each plane will be treated individually in the corresponding "c-velocity" space in order to avoid the noise from other plane models. When we project the plane to the "c-velocity" image, FOE position is required to compute the "c-values". According to the direct "c-velocity" approach, a plane should be presented as a straight line in the corresponding "c-velocity" image. But if the FOE position is badly estimated, the "c-values" are then badly computed. So there will be a dispersion of the "line" representation. To see how the precision of FOE will impact the "line" dispersion, we create a simple synthetic scene with only one horizontal plane to model road, the optical flow of this plane is computed directly from the motion equations to eliminate the impact of optical flow estimation method (see Figures 4.24).

The relation between the "line" dispersion and the distance from estimated FOE to real FOE is illustrated in Figure 4.25. Different FOE positions are used to construct the road "c-velocity" images. In Figure 4.25(a), the used FOE corresponds to the real FOE, so the "line" representation of this horizontal plane is very fine without any dispersion. In Figure 4.25(b), the used FOE is different from the real FOE ($\Delta x_{FOE} = 7$ pixels, $\Delta y_{FOE} = 7$ pixels), we can easily find a dispersion of the "line" representation in the



(a)Distance = 0 (b)Distance = 10 (c)Distance = 20

FIGURE 4.25: "C-velocity" images for a road plane in the road space when the FOE offset is different

"c-velocity" image. And this dispersion is larger in Figure 4.25(c) as the used FOE in this case is farther than the real FOE ($\Delta x_{FOE} = 14$ pixels, $\Delta y_{FOE} = 14$ pixels).

From the above example, we believe that the "line" dispersion increases when the used FOE is away from real FOE. But if the used FOE positions have the same distance to the real FOE (eg. distance = 10 pixels) and in the different directions, will the "lines" obtained from these different FOE positions have the same dispersion ? The answer can be found in Figure 4.26. We use the same horizontal plane scene for the test. But the used FOE points are with the same distance to the real FOE. If we consider that the real FOE is located in the circle center and the 8 under-testing FOE points symmetrically distributed on the circle to present that they have the same distance but with the different directions to the real FOE. Using these FOE positions, the "c-velocity" images are constructed and shown on the circle. When comparing the different representations of the plane in the "c-velocity" images, we can conclude that for the horizontal plane :

— The influence of Δx_{FOE} is bigger than that of Δy_{FOE}
— The influence of Δx_{FOE} on the plane representation in the "c-velocity" image is not only a severe dispersion but also a deformation (the fork in the end of the "line" presentation)

In brief, FOE accuracy can be reflected by the "line" dispersion. And the influence of the FOE shift is not isotropic. For a road plane, the shift in the x direction has a larger impact than that in the y direction.

Since FOE is used to compute the "c-values", the "line" dispersion is also the "c-value" dispersion. The "c-value" dispersion can be written by Equation 4.13. Where c_m is the "c-value" of point m computed using FOE information, and $c_{mean}(w(m))$ is the average of the *c-values* for all the points located on the same iso-velocity curve of m (see Equation 4.14). The whole dispersion equation is to compute the sum of the "c-value" dispersion for all the points existing on the plane Π.

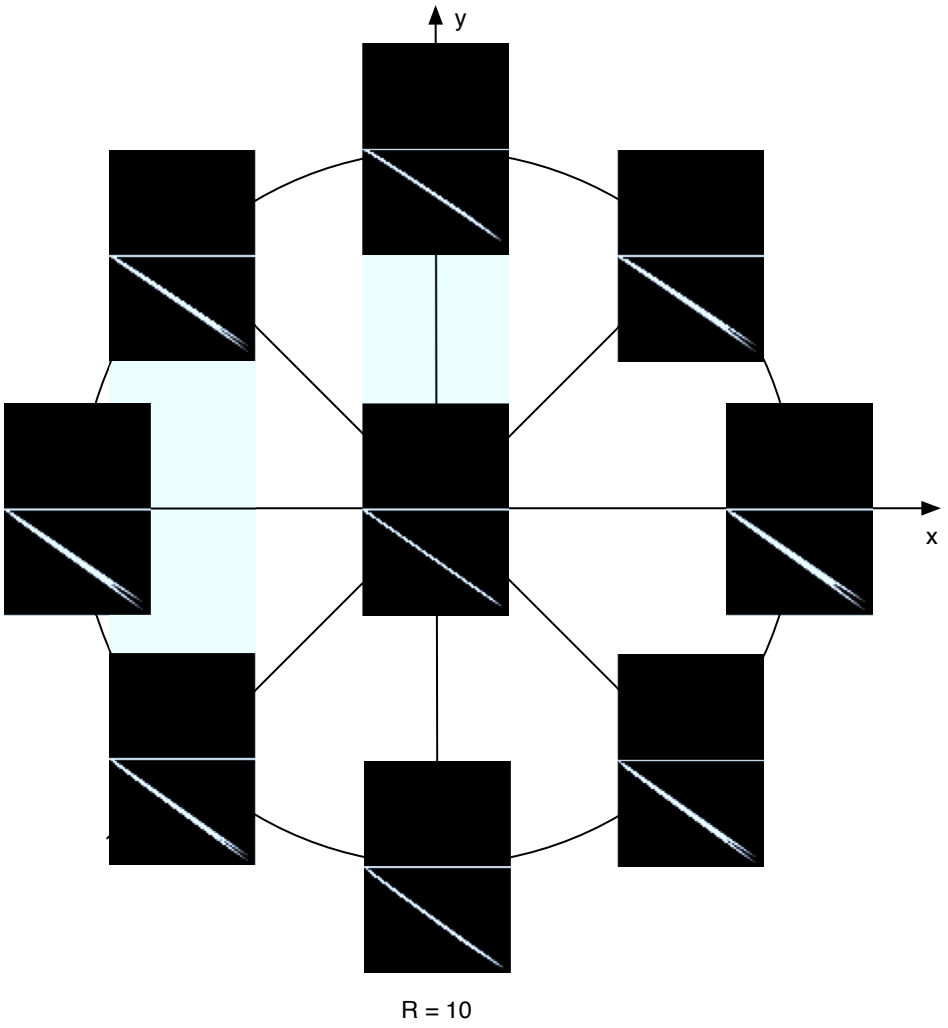


FIGURE 4.26: "C-velocity" images for a road plane in the road space when the FOE discrepancy is constant (R = 10 pixels) but in different direction

$$D_{FOE}(\Pi) = \sum_{m \in \Pi} (c_m - c_{\text{mean}}(w(m)))^2 \tag{4.13}$$

$$c_{\text{mean}}(w(m)) = \frac{\sum_{\substack{i \in \Pi \\ w(i)=w(m)}} c_i}{n} \tag{4.14}$$

With an accurate FOE position, the "c-values" of all the pixels on the same iso-velocity curve are equal to each other, therefore the dispersion is theoretically equal to zero if FOE is correct (as we consider the velocity computation is inerrant). The more imprecise the FOE is, the more dispersed the iso-velocity points will be. So the equation of the dispersion is also considered as the square of standard deviation of all the pixels in the plane Π.

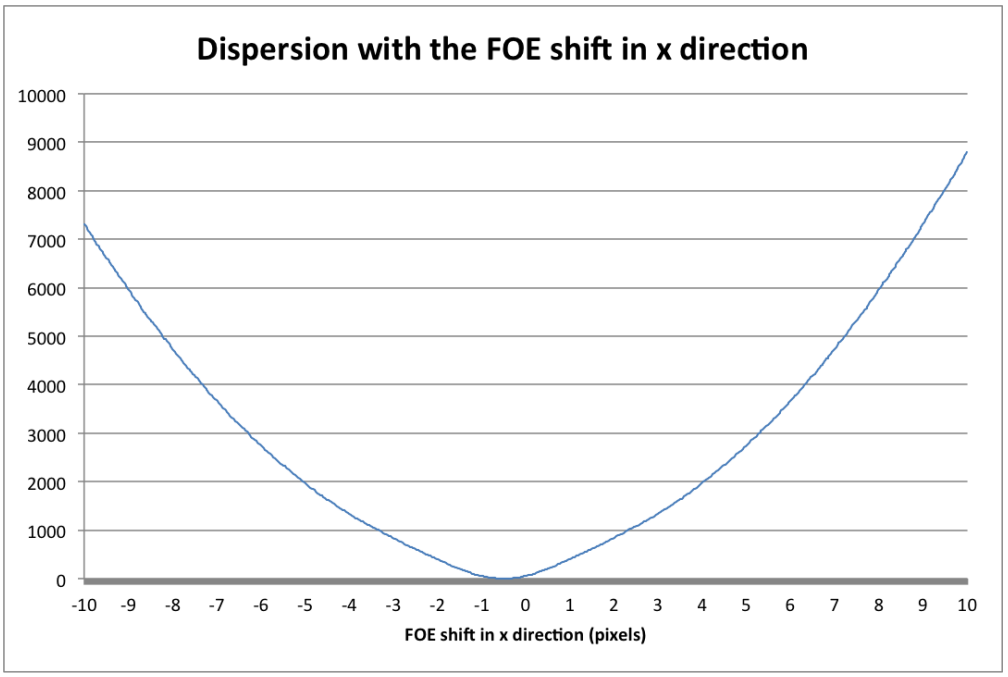


FIGURE 4.27: Dispersion of the presentation of a road plane in the road space, according to the shift of FOE in the x direction

This dispersion, as it is monotonous with the distance between the supposed FOE and the real FOE, can then be used as metric to compute the real FOE position. Here we use the above synthetic example (Figure 4.24) to prove the feasibility. For different FOE positions, the representation (ex. "line") of this horizontal plane in the road "c-velocity" image is also different (the dispersion is different). We can compute the dispersion (Equation 4.13) for each FOE position and then plot a curve as a function of the distance between this supposed FOE and the real FOE. Figure 4.27 shows us the changes of dispersion along with the FOE shift in the direction x. Apparently this curve is monotonous as a function of the FOE shift. Figure 4.28 shows dispersion changes as a function of FOE shift in the direction y. Although there are many local minima, the dispersion is still converging to the global minimum (The location of the real FOE). From these curves, it seems that the classic optimization techniques can be used to find the real FOE.

To further show the possibility of dispersion as a metric to compute FOE position, a mathematical deduction will be presented below. Considering a set of points from the same road plane, all these points have the same velocity norm w. In other words, these points are located on the same iso-velocity curve. Then the *c-values* of a general point m along this iso-velocity curve can be expressed by

$$c_r(m) = K^2 + \Delta x^2 \mp 2\Delta x \sqrt{\frac{K^2}{y^2} - (y - y_{FOE})^2} + \Delta y^2 - 2\Delta y(y - y_{FOE}) \qquad (4.15)$$

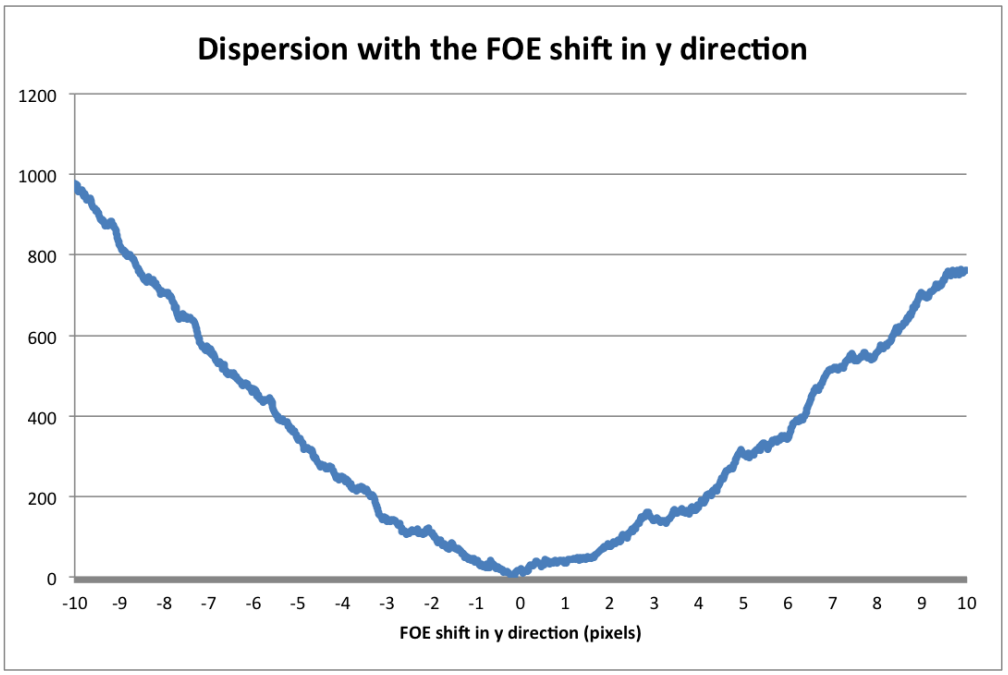


FIGURE 4.28: Dispersion of the presentation of a road plane in the road space, according to the shift of FOE in the y direction

is constant for all the points in the subset $\mathcal{C}$.

The computation of this equation can be found in Appendix C. Where$K = \frac{f d_r w}{T_Z}$ is constant for all the points with the same w. The shifts from the supposed FOE to the real FOE are expressed by Δx and Δy, and y_{FOE} correspond to the real FOE. When the supposed FOE is the real FOE, $\Delta x = \Delta y = 0$ and the equation becomes $c_r(m) = K^2 = constant$. All the points have the same "c-value". The dispersion of this iso-velocity curve is zero.

From this equation, we can also find that the variation (or the dispersion) of the *c-values* along an iso-velocity curve increases in accordance with FOE shifts (Δx, Δy). And there is no interaction between the two components of the FOE shifts. All this information show us that the dispersion presented in Equation 4.13 can be used as a metric to compute FOE. FOE estimation is then turned into a minimization problem on the dispersion $D_{FOE}(\Pi)$:

$$\begin{aligned}(x_{FOE}, y_{FOE}) &= \operatorname{argmin} D_{FOE}(\Pi) \\ &= \sum_{m \in \Pi} (c_m(x_{FOE}, y_{FOE}) - c_{\text{mean}}(w(m)))^2\end{aligned} \tag{4.16}$$

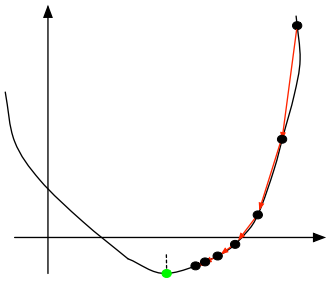


FIGURE 4.29: Gradient descent algorithm

Finally, the FOE estimation problem turns into the minimization of the dispersion function. To solve the minimization problem, classic gradient algorithms like steepest descent [153], Newton's method [154], Quasi-newton method, Gauss-Newton method, and Levenberg-Marguardt algorithm [155] are proposed to find the local minimum. The steepest descent is the simplest method, but we know that the optimization requires numerous iterations. While all other algorithms require the computation of the second order gradients. So for the first step, we still choose the steepest descent algorithm.

From Figure 4.27, the dispersion function in the direction x is monotonous, and x_{FOE} therefore can be estimated directly from a local minimum optimization algorithm. From Figure 4.28, it seems that there are many local minima for the dispersion function in the direction y. So the global minimization algorithm such as simulated annealing should be used. Or we still use local optimizer but with different starting points.

In the rest of this section, we will present the steepest descent algorithm and show how it can be integrate into the FOE estimation.

The steepest descent algorithm starts with an initial guess about the FOE position, and this FOE postilion will approach the real FOE by taking the steps proportional to the negative gradient of the dispersion at the current point (see Figure 4.29). The mathematical representation is like

$$\begin{aligned} x^i_{FOE} &= x^{i-1}_{FOE} - \alpha_x \frac{\partial D}{\partial x_{FOE}} \\ y^i_{FOE} &= y^{i-1}_{FOE} - \alpha_y \frac{\partial D}{\partial y_{FOE}} \end{aligned} \tag{4.17}$$

The partial derivations of the dispersion function can be written by

$$\frac{\partial D_{FOE}(\Pi)}{\partial x_{FOE}} = 2\sum_{m\in\Pi}(c_m - c_{\text{mean}}(w(m)))(\frac{\partial c_m}{\partial x_{FOE}} - \frac{\sum\limits_{i\in\Pi,\, w(i)=w(m)} \frac{\partial c_i}{\partial x_{FOE}}}{n}$$
$$)$$
$$\frac{\partial D_{FOE}(\Pi)}{\partial x_{FOE}} = 2\sum_{m\in\Pi}(c_m - c_{\text{mean}}(w(m)))(\frac{\partial c_m}{\partial x_{FOE}} - \frac{\sum\limits_{i\in\Pi,\, w(i)=w(m)} \frac{\partial c_i}{\partial x_{FOE}}}{n} \tag{4.18}$$
$$)$$

Where the partial derivations of the "c-values" are written by

$$\begin{aligned}\frac{\partial c_i}{\partial x_{FOE}} &= \frac{\|y_i\|(x_{FOE}-x_i)}{\sqrt{(x_i-x_{FOE})^2+(y_i-y_{FOE})^2}} \\ \frac{\partial c_i}{\partial y_{FOE}} &= \frac{\|y_i\|(y_{FOE}-y_i)}{\sqrt{(x_i-x_{FOE})^2+(y_i-y_{FOE})^2}}\end{aligned} \tag{4.19}$$

Input: Initial guess of FOE (x^0_{FOE}, y^0_{FOE}), a set of pixels Π corresponding to a road plane, optical flow norm $\mathbf{w}$ for Π, ϵ_0
Output: FOE(x_{FOE}, y_{FOE})

```
begin
    ε = ∞ ;
    (x_FOE, y_FOE) = (x^0_FOE, y^0_FOE) ;
    D^prev_FOE(Π) = 0 ;
    while ε > ε_0 do
        Compute ∂D/∂x_FOE from Equation 4.18 ;
        Update x_FOE by Equation 4.17 ;
        Compute ∂D/∂y_FOE from Equation 4.18 ;
        Update y_FOE by Equation 4.17 ;
        Compute D_FOE(Π) by Equation 4.13 ;
        ε = |D_FOE(Π) − D^prev_FOE(Π)| ;
    end
end
```

Algorithm 4: Gradient descent algorithm for FOE estimation

An example of the FOE estimation method using steepest descent is shown in Algorithm 4. In this method, the updates of x_{FOE} and y_{FOE} are alternate, which means that the iterative number is the same for these two parameters. In fact, since there is no coupling between these two parameters, we can estimate them separately with different iteration number and iteration condition. This seems more efficient since the optimization on x_{FOE} is much easier than that on y_{FOE} for the horizontal planes, we can spend more time on the optimization of y_{FOE}.

The choices of the parameters (α_x, α_y) are very important in the steepest descent algorithm for estimating (x_{FOE}, y_{FOE}). They are usually very small in order to ensure the convergence. But if they are too small, it will take many iterative number to reach the

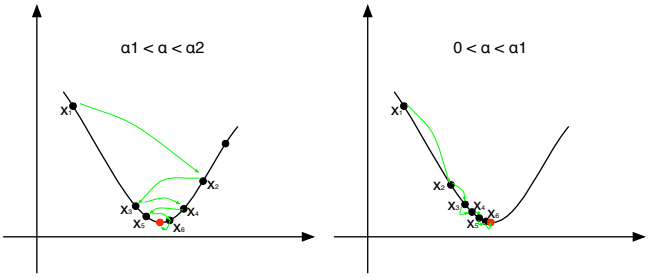
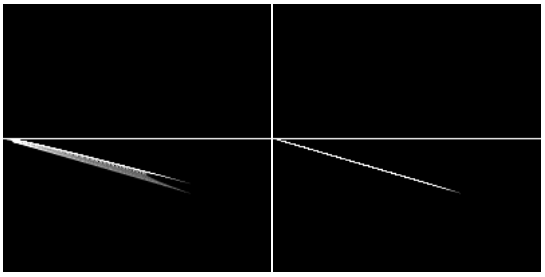


FIGURE 4.30: Two types of convergence for the gradient descent algorithm



(a) (b)

FIGURE 4.31: Road "c-velocity" images before (a) and after (b) the optimization of FOE

minimum point. We present here two different parameter limits ($\alpha 1$, $\alpha 2$). When $\alpha > \alpha 2$, the convergence is no more guaranteed. $\alpha 1$ is the limit of the parameter to distinguish two types of convergences. When $\alpha < \alpha 1$, the iteration will monotonously converge to the minimum point (see the right image in Figure 4.30). When $\alpha > \alpha 1$, the iteration will converge to the minimum point as a zigzag line (see the left image in Figure 4.30). In our case, the selection of α_x is not very strict as the dispersion curve with the FOE shift in x direction is strictly monotonous. But the selection of α_y should not be so small that the optimization can easily fall into the local minimum.

An optimization result is shown in Figure 4.31, where we can compare the plane representations in the "c-velocity" image before and after the optimization. Apparently after the optimization, the deformation of the road plane representation is much compensated.

4.5.3 Results on synthetic flow images

The above synthetic example proved the validation of this inverse "c-velocity" method, this synthetic example will also be used to study the sensibility of this approach on several factors :

— the noise of optical flow
— the influence of rotations that are not considered in our model

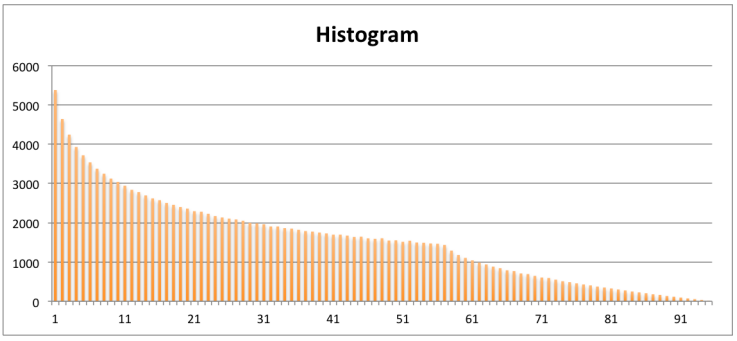


FIGURE 4.32: Histogram of the synthetic flow norms

— the extracted plane numbers

The noise from optical flow A common impact for the inverse "c-velocity" approach is the noise from optical flow field since the angular and endpoint optical flow errors from actual estimation techniques are still very important [114]. To study the influence of optical flow noise in our FOE estimation method, an additive uniformly distributed noise in the interval $[-\delta, \delta]$ is added on both components of the flow vector for each pixel :

$$\begin{aligned} u &= u_0 + U(-\delta, \delta) \\ v &= v_0 + U(-\delta, \delta) \end{aligned} \tag{4.20}$$

Then we augment the noise amplitudes δ to see their influences on the FOE estimation. The results are shown in Figure 4.33. The maximum noise amplitude δ is 10 pixels. As the noises are added on both flow vector component, the maximum noise for the flow norm is then about 14 pixels. To compare it with the histogram of the synthetic flow norm shown in Figure 4.32, the noise is however very important.

From Figure 4.33, it seems that our FOE estimation approach is very robust to flow noise, especially for x_{FOE}, the FOE location error in x direction is never beyond 1 pixel. Maybe it is due to the horizontal plane that we are using. The FOE location error in y direction increases according to the augmentation of noise. However, the precision of FOE location impact is very small ($< 5 pixels$) when the additive noise is smaller than 3 pixels. Therefore, we can conclude that the proposed method is very robust to the flow norm noise (since our method uses only the flow norm, not the flow direction). And a good solution to deal with the noise is that using the road plane (horizontal plane) for x_{FOE} estimation and the building plane (vertical plane) for y_{FOE} estimation.

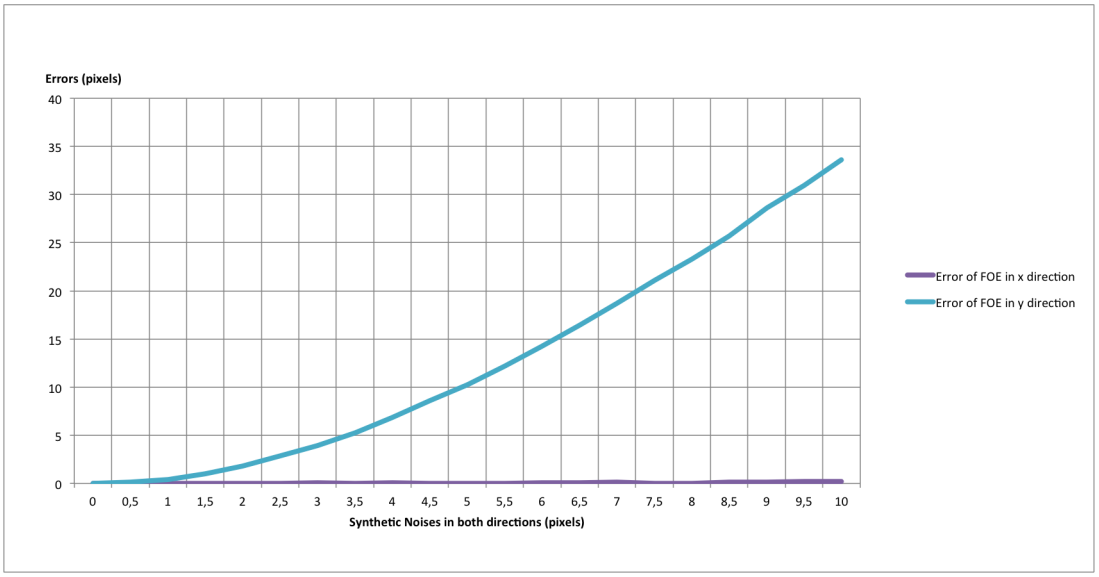


FIGURE 4.33: Impact of the optical flow noise du bruit on the location of FOE

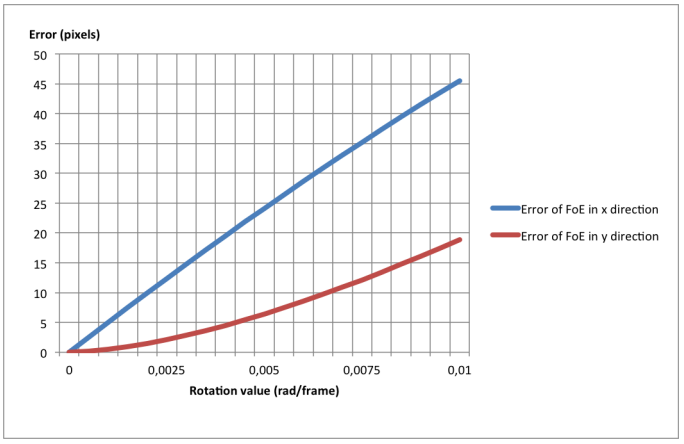


FIGURE 4.34: Influence of the rotations (yaw in this case) on the FOE precision

Influence of the rotations In the proposed approach, more generally, in the formalism of the "c-velocity", the rotational egomotion is not considered. This assumption may look like limited. However, there are plenty of methods to overcome this rotation. These include the approaches based on point tracking and the tri-linear tensor estimation [156], or that rely on the properties of optical flow [157]. When the rotation is large, we may have to remove it using the above methods. But we should still study the impact of the small rotation on our FOE estimation method. Although the rotation is three dimensional, in the case of vehicles on an urban road, we only add the yaw to flow vectors. Then the results are shown in Figure 4.34, it seems that the FOE error linearly increases with the augmentation of the yaw speed. And this influence is larger for the precision of x_{FOE} than that of y_{FOE}.

Influence the extracted plane numbers In all the above tests, only a horizontal plane is used for the FOE extraction. But if we use several planes for the estimation, the dispersion is then rewritten by

$$D_{FOE} = \sum_{\Pi_i \in \mathcal{P}} \sum_{m \in \Pi_i} (c_m - c_{\text{mean}}(w(m)))^2 \tag{4.21}$$

We should mention that these planes can belong to the different types ("Building" or "Road"). But moving "obstacles" planes are never used for FOE localization since these obstacles usually have their own motions and the extracted FOE from this type of plane is not the right FOE location purely due to the camera egomotion.

When using divers planes for FOE extraction, we did not note a significant influence of used plane numbers. On the other hand, the computation time largely increased with the augmentation of used plane number. So we choose only a plane for each estimation. If there are several planes, we select the one that has the best representation in the Hough transform space for the detection. Or from the above conclusions, the "best" building plane is used for x_{FOE} estimation and the "best" road plane is used for y_{FOE} estimation.

4.5.4 Results on real images

This approach is also tested on the real images from KITTI database, where scene structures are extracted by 3D data from a calibrated LIDAR sensor. And optical flow field is estimated by the method proposed in [148].

The problem of the test on real images is that we can never get a ground truth as the comparison base. Although there are some databases used for the comparison, such as the data from the University of Karlsruhe, we can expect that the precision of the FOE position is about 13 pixels. This precision is not enough to construct a solid base for the comparison.

We can therefore only give a qualitative evaluation for the extracted FOE. Figure 4.35 presents the FOE (the blue point) extracted from our method, one can compare it with the image center (the green point). We can see that the optical flow field of the road plane is towards to this FOE point, which proved the accuracy of the FOE location.

In fact, besides of using flow field lines as visual indicators for the FOE evaluation, another way to see the quality of the extracted FOE is the comparison between the initial "c-velocity" image (constructed with a supposed FOE located in the image center) and

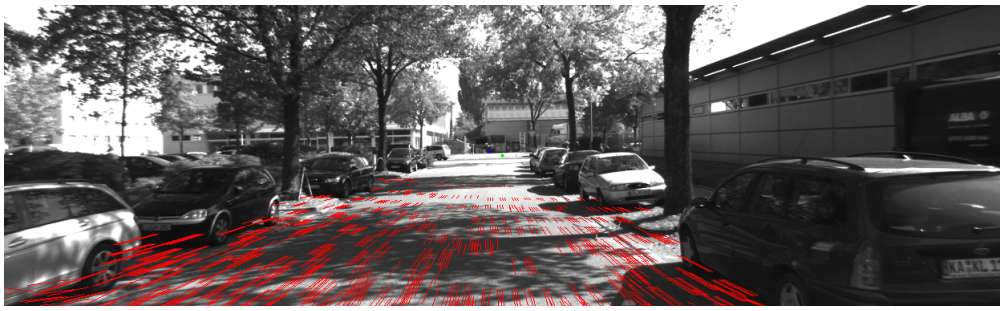
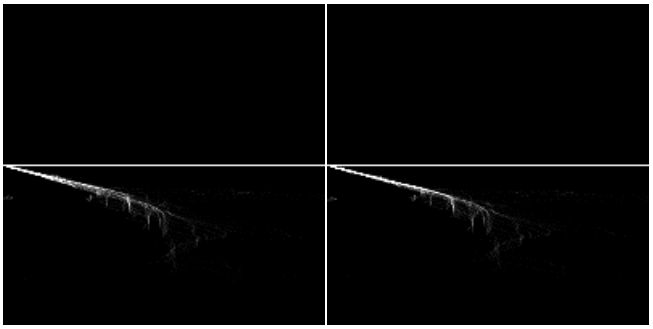


FIGURE 4.35: Extracted FOE (the blue point) by the proposed method



(a) (b)

FIGURE 4.36: "C-velocity" images before (a) and after (b) the optimization method

the final "c-velocity" image (constructed with the extracted FOE). Such comparison can be found in Figure 4.36. When comparing these two images, we can easily find that the "line" representation of road plane is much finer after the optimization.

4.6 Conclusion

In this chapter, we have presented three different methods for the Focus of Expansion localization from monocular image sequences. The first method is based on a voting process of optical flow field. The pixel that wins the most votes from optical flow vectors is considered as the FOE location. Although the cumulative process is very robust to the imprecision of optical flow quality, The time cost of this method is very high since each flow votes for all the pixels locating on the flow vector line. However, the second method computes the FOE position from the linear equation $Ax = b$ that is resolved by the help of Singular Value Decomposition. Since this least square method is very sensible to the flow quality, RANSAC algorithm is then used for the pre-selection of optical flow. The last method, but also the most important one, is called the inverse "c-velocity" method. Unlike most of existing methods including the above methods, this method uses only part of the flow information that is the flow norm. Therefore, this method is very robust to the noise of flow vector orientations, which is the most common error for the actual

techniques. However, to make use of this method, at least a rough scene structure should be estimated since the inverse "c-velocity" is based on these extracted planes. It seems very complex since the scene structure extraction needs either a sensor or a stereo-vision, but the ideal is to combine it with the direct "c-velocity" approach since the latter one uses the FOE information to execute the rough extraction of the scene structure. So an iterative method between the direct "c-velocity" and the inverse "c-velocity" allows to obtain simultaneous estimation of the structure of the scene and the ego-translational motion.

When evaluating this inverse "c-velocity" method, a purely synthetic example is firstly given to prove the exactness of the method and also discuss the influence of different factors such as flow noise, vehicle rotations and used scene plane numbers. The results showed that our method is very robust to these factors. Despite the use of the advanced measuring equipment, we were unable to obtain sufficiently precise measurements in real situation. However, the quantitative evaluation of our method on simulation sequences is sufficient to open a window for the real images.

5

Improving decisions using c-velocity

Contents

5.1 Introduction

Object recognition is very important in the framework of intelligent vehicle system. For example, road region should be detected as free space for vehicle navigation, obstacles such as moving vehicles or crossing pedestrians should be detected to avoid a collision. The surrounding objects such as buildings in the urban situation can be also detected to help the scene understanding. However, it is difficult to detect and recognize these objects at the same time. Many efficient approaches are designed specifically for only one type of object detection and recognition. These objects have indeed different properties and it is difficult to find a same formalism to detect them and then classify them into different categories. While the "c-velocity" approach seeks for a solution for this problem. Despite of their different properties, all these objects can be approximated by the planes with different orientations. Then we project these planes into a new space where their representations become into lines. Apparently line extraction is much simpler than parametric surface detection.

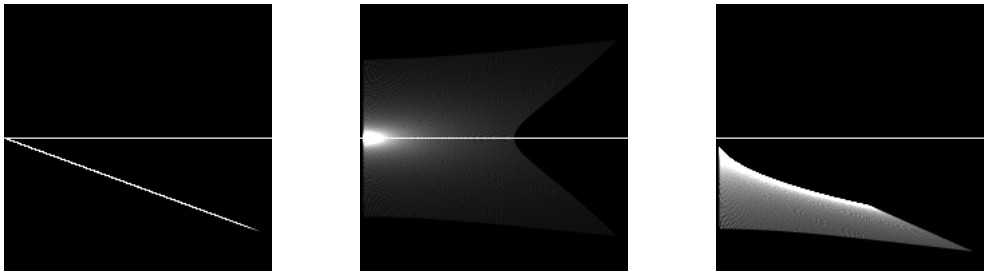
To be brief, the pursuit of the "c-velocity" is to approximate objects by planes and then detect them from other spaces where the planes are projected as easily detectable simple forms (eg. lines). However, in practice, due to the external perturbation like optical flow imprecision, wrong Focus of Expansion position and the internal perturbation (inter-space perturbation), object detection is difficult to perform. In this chapter, we will deal with the internal perturbation and make the "c-velocity" approach more robust.

5.2 Voting spaces cooperation for 3D plane detection from monocular image sequences

In the "c-velocity" framework, the most important objects in an urban road situation can be assimilated into three orientations of planes, and the construction of the "c-velocity" image for each type of plane is different since the computation of the "c-value" is unique for each type (see Table 5.1). This means that the projection of the plane is meaningful and representative only in its own "c-velocity" image. And the representation of this projection in other "c-velocity" images is considered as background noise. Figure 5.1 is a synthetic example to show the representations of a road plane in different "c-velocity" images. In this synthetic example, the optical flow field is directly computed from Equation 2.12 and the FOE position is obtained directly from Equation 3.34. So representations in the "c-velocity" images are constructed without any external errors. The road plane is represented as a straight line in the road "c-velocity" space (see Figure 5.1(a)). While in the others "c-velocity" spaces (the building space in Figure 5.1(b) and

Plane type	representative object	"c-value" presentation
Horizontal plane	road	$\lvert y\rvert\sqrt{(x-x_{FOE})^2+(y-y_{FOE})^2}$
Vertical plane	building	$\lvert x\rvert\sqrt{(x-x_{FOE})^2+(y-y_{FOE})^2}$
Lateral plane	obstacle	$\sqrt{(x-x_{FOE})^2+(y-y_{FOE})^2}$

TABLE 5.1: "C-value" presentations for different types of planes



(a) (b) (c)

FIGURE 5.1: Representation of a road plane in different "c-velocity" images : (a) The road space ; (b) The building space ; (c) The obstacle space

the obstacle space in Figure 5.1(c)), the votes from this road plane are scattered without any form. This background noise will more or less impact the detection of other models. From the synthetic example (see Figure 2.10), it seems that the noise from other models can not impact the plane detection in different spaces. But in practice, the imprecision of the optical flow field and FOE position make the plane detection more complex from these background noises. To get a robust plane detection method, we adapt an iterative histogram splitting approach [3]. The basic idea is that we first extract the easiest-detectable plane (less noising plane) and then we update all the "c-velocity" spaces by removing the contributions of this plane to reduce the background noise, and then do the second iteration until no plane can be detected. To better understand this method, we firstly recall the classic "c-velocity" process.

5.2.1 Classic "c-velocity" process

From Chapter 2, the idea of the "c-velocity" approach for object detection in an urban road situation is to project object planes into three different "c-velocity" spaces, and each space is corresponding to a plane type (see Table 5.1). Only in the correct "c-velocity" space, the object plane is represented as a straight line. Depending on the situations, the use of the "c-velocity" can be very different. Here we define three parameters to distinguish the different situations (see Table 5.2). The research area define that in which region the plane detection is executed. Generally, this area is the whole image. But with some a priori information, we may reduce the research area into a small region. In the latter case, the background noise in the "c-velocity" images is much smaller since

Research area	Plane number	Plane type
Reduced region	Known	Known
Whole image	Unknown	Unknown

TABLE 5.2: Different situations defined by three parameters

reduction of the research area is equal to the outliers elimination. The second parameter is the plane number. If we know how many planes will be detected, the plane detection process will be an iterative method with known iteration time. But if we do not know the plane number, we should also define a way to determine if there is a plane or not. The last parameter defines the plane type condition. If we know the plane type at first, we can only search the "lines" in the corresponding "c-velocity" image. But when the plane type is unknown, we should search the "lines" in all the "c-velocity" spaces. In this case, it becomes space cooperation problem.

In our case of intelligent vehicle frameworks, we should extract all the potential objects from the image sequence captured by a vehicle camera. The plane detection is of course on the whole image, the plane number and the plane type are not known before. So our situation can be explained by

- Research Region
  - ☐ Reduced region
  - ☒ Whole image [1]
- Plane Number
  - ☒ Unknown
  - ☐ known
- Plane Type
  - ☒ Unknown
  - ☐ known

An algorithm is shown in Figure 5.2 in order to deal with this problem. The whole process is divided into two steps : The first one - "c-velocity" process - is actually a cumulative process (see Figure 5.3). In order to deal with different plane orientations we have to consider several "c-velocity" spaces at the same time. This means that each pixel will vote in all the "c-velocity" spaces whatever its plane model. For example in Figure 5.3, for a pixel with the flow norm w_i (the red pixel in the Flow Map), the "c-values" in three spaces (c_b, c_r, c_o) are computed by Equation 2.16 with the known FOE position. Then this pixel will vote for the point (w_i, c_b) in the building "c-velocity" space, for the point (w_i, c_r) in the road "c-velocity" space and for the point (w_i, c_o) in the obstacle "c-velocity" space. When the voting is finished for all the pixels, the Hough Transform

1. In the next section, we will show that the sky region are firstly removed

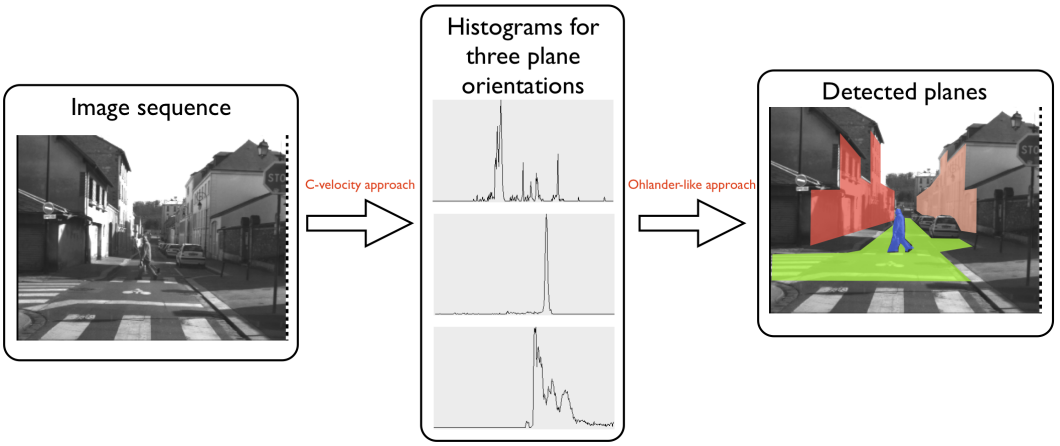


FIGURE 5.2: Voting spaces cooperation algorithm overview

is applied in each space for the "lines" extraction [2]. The votes from a "line" in the "c-velocity" image (or a plane in the scene image) contribute to make a peak emerge in the corresponding 1D histogram. However, in practice, due to optical flow imprecision and relative plane size variations, what we call "inter-model perturbation", the *maxima* selection is more difficult to perform if we do the peak extraction independently in each histogram. So the cooperation between three spaces are done using an iterative histogram splitting algorithm. This approach, that was first proposed for color image histograms, has the advantage to help *maxima* extraction in one histogram ("c-velocity" for a given plane model) by reducing the contribution of voters belonging to other plane models. Next section will detail the complete iterative process.

5.2.2 An Ohlander-like histogram splitting from *c-velocity* spaces

The histogram splitting method was first proposed for segmenting color images [3]. This iterative approach considers as an input a given region [3] in the image with the corresponding set of color histograms (Red, Green, Blue, Intensity, Hue, Saturation, Y, I, Q), the most prominent peak is selected among all the histograms. Then this peak is extracted by determining the upper and lower thresholds, and the corresponding pixels (a subset region) are removed from the current region. The segmentation is iterated until no peak can be found. The flow chart of the procedure for this iterative segmentation algorithm is given by Figure 5.4.

One disadvantage of the Ohlander algorithm is that the RGB features are highly redundant. In our case, except on plane boundaries, the voting spaces are not correlated. Then, we can expect to avoid the well-known drawbacks of the Ohlander techniques.

2. *The Hough Transform is one-dimensional with the line slope as parameter*
3. *Initially, this selected region is the entire image*

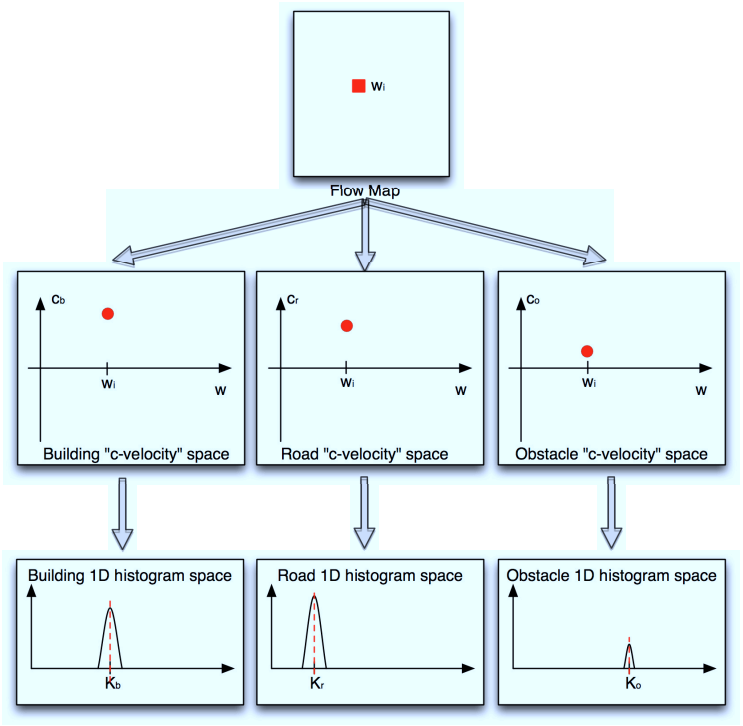


FIGURE 5.3: The "c-velocity" process to deal with the situation of intelligent vehicles

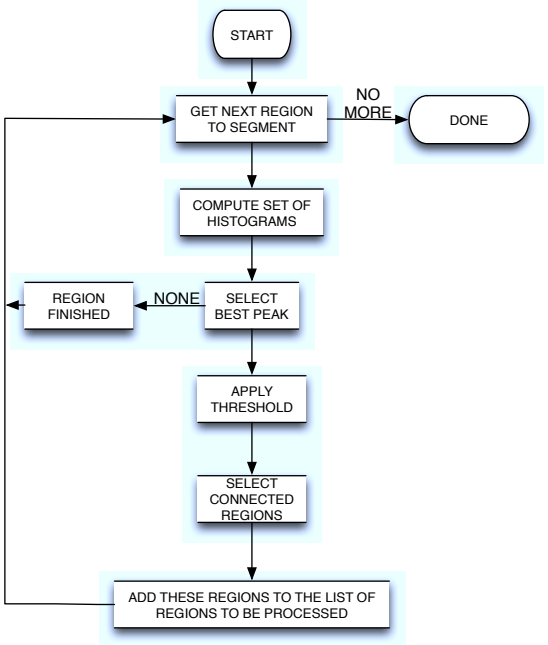


FIGURE 5.4: Region splitting based segmentation procedure [3]

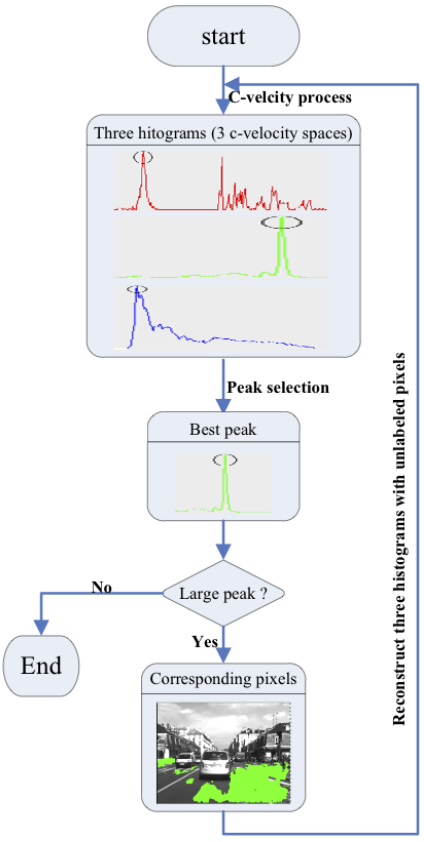


FIGURE 5.5: Schema of Ohlander-like algorithm

In order to reduce computation time and to simplify the "c-velocity" plane detection process, we build the voting spaces directly after computing the flow field and determining the "c_values" since the "c-velocity" spaces construction is just an intermediary step. This modification provides an acceleration in computational time as we deal with one histogram (histogram of k) instead of two ("c-velocity" + Hough transform). As a consequence, we need only to deal with three histograms of line slope (k) : one histogram for each plane model. A peak in a histogram corresponds to a line in the "c-velocity" space and to a 3D plane in the scene. Peak significance is function to plane size. The detailed algorithm detecting different categories of planes is given in Algorithm 5 and Figure 5.5.

selection The "best" peak selection can be divided into two steps : the peak selection on each histogram (intra-peak selection) and the peak selection over different histograms (inter-peak selection). Concerning the intra-peak selection, all possible candidates are weighted according to their significance and are associated to a maximum histogram value and two minimum values that are located on each side of the chosen maximum. The significance is computed using the sum of four slopes values (see Equation 5.2 and Figure 5.6) : left-hand slope, right-hand slope, left semi-slope and right semi-slope.

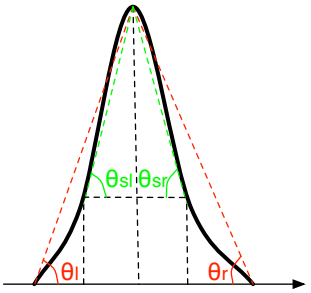


FIGURE 5.6: Schema of four slopes of a peak : left-hand slope $\tan\theta_l$, left semi-slope $\tan\theta_{sl}$, right-hand slope θ_r and right semi-slope θ_{sr}

smoothing Since most of the optical flow estimation approaches do not consider temporal smoothing, flow vectors may vary from one frame to another. These unstable flow vectors will affect directly the peak selection performance. For instance, a building can form a significant peak in one frame if the flow quality is very high. However, this same building may generate a less prominent peak in the next frame. As a consequence, the building is correctly detected in the first frame but missed in the following frame. In order to solve this problem, we apply a first order recursive filter when calculating the peak significance. It allows to implement a sort of "peak confirmation" from one frame to another before the final decision. The significance S of a peak is updated according to Equation 5.1, where α allows to adjust the importance of the history and $P(k)$ the probability density function associated to the slope k.

$$S(k) = \alpha P(k) + (1 - \alpha)S(k - 1) \tag{5.1}$$

Introducing priority for inter-peak selection In the case of intra-peak selection, the peak significance S depends on the corresponding plane size. When comparing peaks from different voting spaces, we have to consider the plane "representation" in the vote. For instance, a building or a road have usually more associated velocity vectors – thus more voters – than an obstacle (a moving car or a pedestrian). Therefore, their contribution in the obstacle voting space can have a serious impact on the obstacle peak detection. Moreover, a small peak in the obstacle histogram should be detected even if its significance is lower than the ones associated to the peaks in the building or road histograms. As a consequence, we propose to define a priority to the significance of a peak : highest priorities are given to "building" peaks, then to "road" peaks and finally "obstacles" peaks will be considered with a lowest priority. In practice, priority is implemented using a weighted value on the significance S associated to each peak (see Equation 5.3).

Corresponding pixels labeling After selecting most significant peaks from the set of considered histogram, corresponding pixels in the image are labeled. Then the resulting labeled image is post-processed. Processing include : expansion/shrink and small blobs suppression. The next iteration of peak selection starts again without labeled pixels. This process is repeated until no significant peak could be found.

Require: optical flow w_x with x $\in \mathcal{R}_0$, $\mathcal{R}_0$ = whole image

1.C-velocity process

Output : three histograms h_r, h_b, h_o

2.Intra-peak selection

Output : three peaks P_r, P_b, P_o

Definition of peak : A peak P_k in a histogram $h_x \in h_r, h_b, h_o$ is represented by a maxima m and an interval $[n_1, n_2]$ with the restriction that $m \subset [n_1, n_2]$, $\forall i \subset [n_1, m[, h_X(i) \leq h_X(i+1)$ and $\forall i \subset]m, n_2], h_X(i-1) \geq h_X(i)$.

Definition of peak significance S1 :

$$S1 = \frac{h_X(m) - h_X(n_1)}{m - n_1} + \frac{h_X(\frac{m+n_1}{2}) - h_X(n_1)}{\frac{m-n_1}{2}} + \frac{h_X(n_2) - h_X(m)}{n_2 - m} + \frac{h_X(n_2) - h_X(\frac{m+n_2}{2})}{\frac{n_2-m}{2}} \quad (5.2)$$

3.Inter-peak selection

Output : $P_x \in \{P_o, P_b, P_r\}$

Definition of peak significance S2 :

$$S2 = S1 \times \text{factor}_{space} \quad (5.3)$$

$\text{factor}_{building} > \text{factor}_{road} > \text{factor}_{obstacle}$

4.Projection in the image with P_x

Output : Labeled Region $\mathcal{L}_i$

5.Post-processing of $\mathcal{L}_i$

Output : Novel labeled Region $\mathcal{L}_i$

6.New region calculus for next iteration

Output : $\mathcal{R}_{i+1} = \mathcal{R}_i - \mathcal{L}'_i$

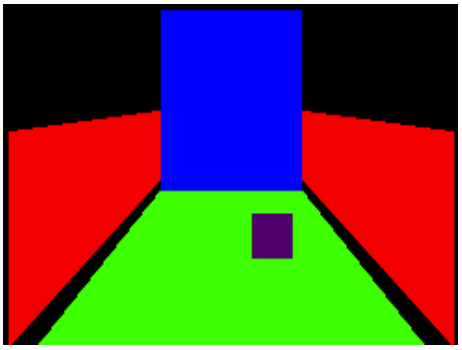
7.Redo step 1 - 6 with optical flow w_x with x $\in \mathcal{R}_{i+1}$

Algorithm 5. Ohlander-like histograms splitting

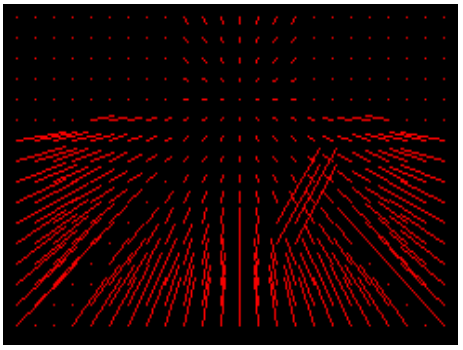
5.2.3 Experimental results

5.2.3.1 Synthetic 3D scene flow

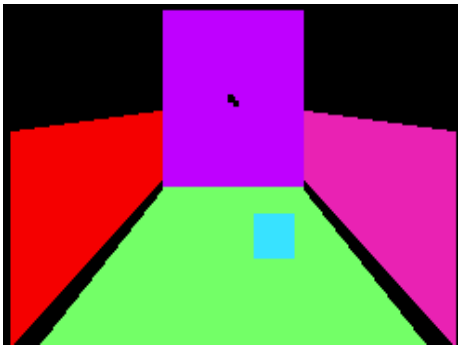
In the following toy example (see Figure 5.7), a synthetic velocity vectors field of a moving 3D scene is generated using Equation 1 (see Figure 5.7(b)). The scene is made up of 5 main planes : a frontal plane that corresponds to a fleeing car (blue plane in Figure 5.7(a)), a frontal plane corresponding to a crossing pedestrian (violet plane in Figure 5.7(a)), two lateral planes that correspond to two buildings (red plane in Figure 5.7(a)) and one horizontal plane corresponding to a road (green plane in Figure 5.7(a)). The labeled optical flow result is given in Figure 5.7(c) and shows that the 5 planes are successfully detected after five iterations. A small defect in the center of the violet plane is caused by the fact that 2D velocity values in the center of image (near the Focus of Expansion) are filtered when calculating the *c-velocity* values because they are negligible. The corresponding labeled optical flow vectors are given in Figure 5.7(d).



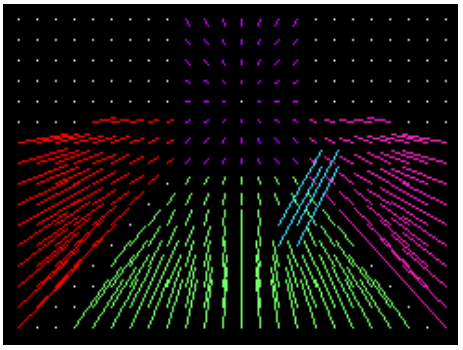
(a) Synthetic scene



(b) Synthetic velocity field



(c) Detected planes labeled with different color



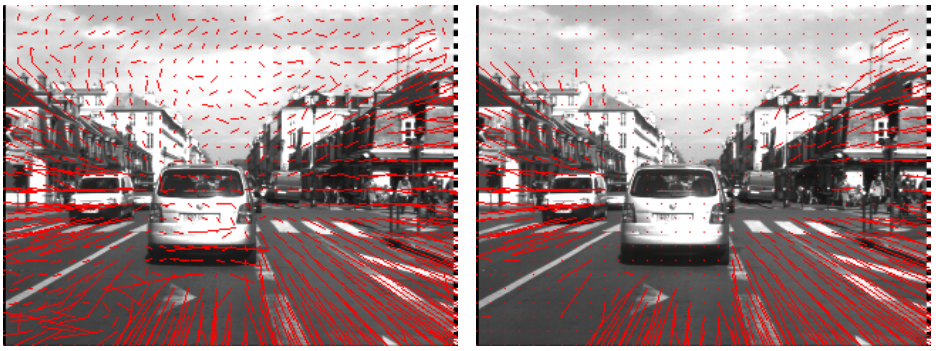
(d) Labeled velocity field

FIGURE 5.7: Results on synthetic image sequences

5.2.3.2 Real Image sequences

All image sequences considered for the experiments stem from the French research institute on transportation (IFSTTAR). The database includes various sequences of real car driving in urban scenes (in the city of Versailles) with different kinds of vehicle motions.

Pre-filter for the optical flow field The process needs an optical flow estimation, and the classical FOLKI [158] method was chosen with a 9×9 window size. But the flow quality is not very good and stable. A filter is applied to the flow field before using it for the detection process. In order to filter erroneous vectors, we exploit the spatio-temporal smoothing motion property : vectors are eliminated if no spatial or temporal confirmation is found. The spatial and temporal window is a fixed parameter that could be set up using depth information if needed. Figure 5.8 shows the resulting difference between the original flow computed by FOLKI (Figure 5.8(a)) and the remaining flow after filtering (Figure 5.8(b)).



(a) Original optical flow estimated by FOLKI (b) Results after optical flow filtering

FIGURE 5.8: Optical flow filtering

Another purpose is to filter the flow field from the sky pixels, so we should firstly detect the sky region. The homogeneity of the gray level is used to detect the sky region. To obtain the homogeneity of each pixel, the gray-level gradient is computed in both directions (x and y). And pixel is considered as sky if the compute gradient is small and the surrounding pixels also have small gradient. To get a better detection accuracy, the gradients are also computed in different resolutions. An example of such sky detection is shown in Figure 5.9

Real image results Even if the assumption of pure camera translation is not exactly valid in this database, we still detect successfully the three categories of planes : road (green parts in Figure 5.10(b), Figure 5.10(d) and Figure 5.10(f)), obstacles (blue parts

FIGURE 5.9: Detected sky region labeled with red color

in Figure 5.10(b), Figure 5.10(d) and Figure 5.10(f)) and buildings (red parts and yellow parts in Figure 5.10(b), Figure 5.10(d) and Figure 5.10(f)).

While computing the *c-value* for three spaces (c_r, c_b and c_o in Equation 3.35), the Focus of Expansion (FOE), which is the projection of the translation motion of the camera on the image plane, should be estimated before. Several methods have been suggested for computing the FOE in Chapter 4. In this study, we chose to estimate the FOE using the least square estimation with SVD solution. A RANSAC algorithm is done to select the "good" flow vectors for the SVD solution.

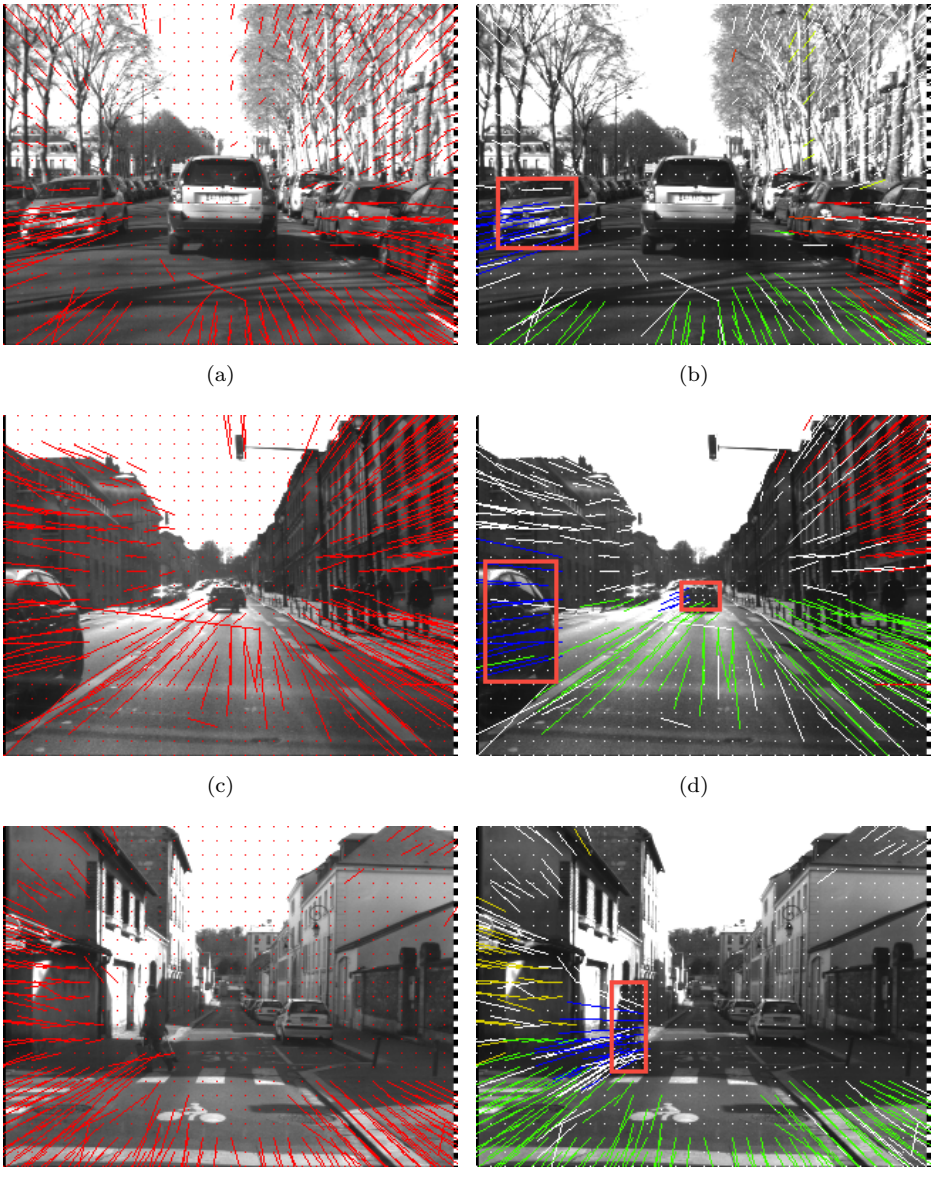
Figure 5.10 shows detection results from 3 urban scenes. Figure 5.10(a), Figure 5.10(c) and Figure 5.10(e) show optical flow field with red color. Figure 5.10(b), Figure 5.10(d) and Figure 5.10(f) show labeled optical flow field. The white flows mean that the corresponding pixels are not detected. The blue part represents the detected obstacles, the green part represents the road location and other colors are associated to the "buildings". In the first scene (see Figure 5.10(a) and Figure 5.10(b)), the stopped cars and the street lamp are detected as two lateral planes. The vehicle in front of the camera is not detected because it has almost the same velocity as the camera, and the associated optical flow is then close to zero. In the second scene (see Figure 5.10(c) and Figure 5.10(d)), the approaching car as well as the fleeing car are both detected. In the last scene (see Figure 5.10(e) and Figure 5.10(f)), we also detect the crossing pedestrian.

In Figure 5.11, we can see the detailed histograms of different object spaces. In the "building" histogram (Figure 5.11(b)), the left building forms a significant peak (in the

red rectangle), which enables to detect it easily. However, the approaching car in the left lane (peaks in green rectangles) also forms an important peak in the "building" histogram (see Figure 5.11(b)). Even worse, this same car is hidden by the largest peak corresponding to other objects like buildings, road, etc. in the "obstacle" histogram (see Figure 5.11(c)). The extraction of this peak directly from this histogram seems to be difficult not to say impossible. A histogram splitting method like the one we propose to use, adapted from the Ohlander approach, is well adapted for this kind of situations. Indeed, when large objects are first detected and their contribution removed from non-corresponding space models, small peak could appear and could become easily detectable (see Figure 5.11(d)). In this scene, the left building (yellow part in Figure 5.11(f)), the road (green part in Figure 5.11(f)) and the approaching car (blue part in Figure 5.11(f)) are successfully detected.

5.3 Conclusion

This chapter presents a new 3D plane detection approach from a moving camera. In the context of urban 3D scene, the main objects like buildings, road, moving cars and pedestrians are assimilated to specific plane orientations (vertical, horizontal and frontal planes). This approximation can on one hand eliminate the depth variable and on the other hand exhibit a proportionality relation between the perceived velocity w and the so-called *c-velocity* value. This proportional relationship can be exploited using a Hough-like transform. An iterative method derived from Ohlander multi-band image histogram segmentation approach is integrated for a complete plane detection algorithm. The whole process is proved to be feasible using both synthetic and reel image sequences. Results confirm the robustness of the approach that was implemented in real-time conditions. Our next step is to deal with more general parametrized surfaces than planes and to consider more complex camera motion.



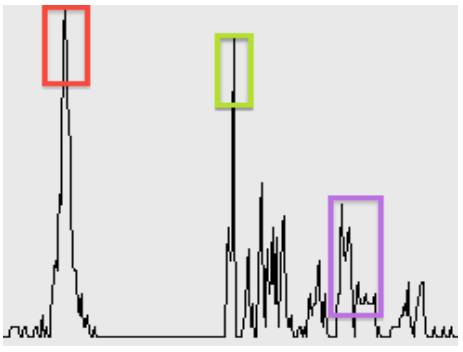
(a) (b)

(c) (d)

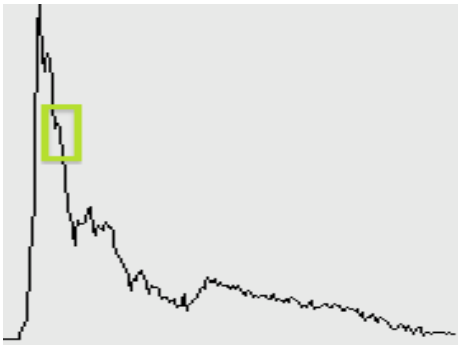
(e) (f)

FIGURE 5.10: Results on real image sequences

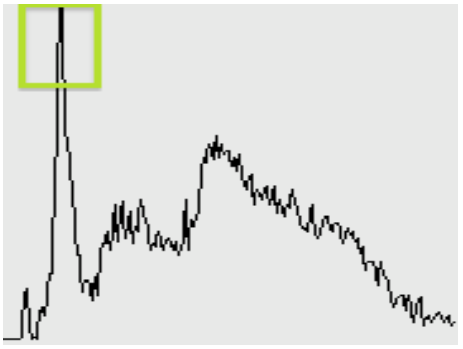
(a) Image scene



(b) "Building" histogram at first iteration

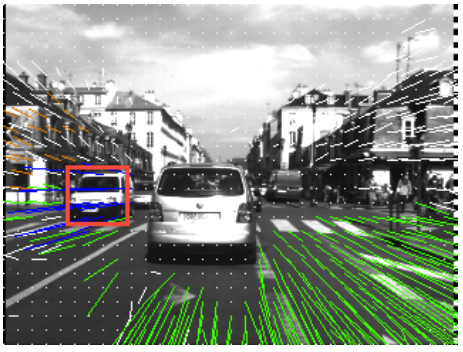


(c) "Obstacle" histogram at first iteration



(d) "Obstacle" histogram after detecting road object

(e) Filtered optical flow



(f) Labeled Optical flow

FIGURE 5.11: Histogram results on real image sequence

6 Conclusions

6.1 Conclusions

This thesis research is centered around a monocular vision based object detection method for the ADAS application. The approach exploits the motion information from the image sequences captured by a vehicle mounted camera to detect objects ahead of this host vehicle. In the context of driving on an urban road situation, there exist almost three different types of objects : buildings, road and obstacles. Road can be approximated as horizontal plane. Buildings locating on both sides of the road can be assumed as lateral planes. Moving obstacles like fleeing/approaching vehicles and crossing pedestrians are then considered as frontal vertical planes. Then the proposed "c-velocity" approach can project these planes onto the corresponding "c-velocity" frames and detect them in these new spaces as the representation of these planes in the new spaces are simple forms (ex. lines). Such idea of transforming detecting space comes from a stereo-vision based road detection method - "v-disparity" approach. However, the latter is proved to be very robust for object detection and widely used in many researches. Our new monocular method has suffered from many influences. So in our research, many efforts are done to improve the robustness of this method.

At first, the modification is about the algorithm. As one key for the vision-based ADAS system is the real-time constraint, we optimize the algorithm and also implement it with multi-thread programming. These improvements allow to at least reduce the computational cost by half. Then in order to make this algorithm possible to combine with the "v-disparity" approach, a variety called the "v-velocity" frame is designed. We can with this new frame on the one hand model the camera angular, and on the other hand easily integrate the "v-disparity" image.

In the "c-velocity" approach, the construction of the "c-velocity" frames needs optical flow and FOE information. So optical flow estimation and FOE estimation are two preliminary steps. The results from these steps will influence the object detection using "c-velocity" approach. So our second study is to find a suitable optical flow estimation method for our detection method. For this proper, we surveyed the state-off-the-art optical flow estimation method and also the evaluation benchmarks. Then we found that the existing estimation method is not efficient when there are really large motions (nearly several tens of pixels) and the regions are lack of textures. Unfortunately, road is in this case as it is lack of texture and its optical flow is usually very large due to the vehicle egomotion. Therefore, we proposed a model-based optical flow estimation method to cope with this situation. In this approach, optical flow is divided into two parts : we compensate the dominant flow using the available a priori knowledge from other sensors or from previous frames. Then the remaining part flow is easily detectable by a classical optical flow method or a simple block matching method. This method was tested both on synthetic and real images and proved better resulting than other existing methods.

At the same time, three different FOE estimation methods are proposed to supply the FOE information for the "c-velocity" approach. The first one think that all optical flow vectors due to the translational motions direct at the FOE point. Then a voting scheme is employed to deal with optical flow noises. The second method computes FOE point by solving a large linear equation system. The SVD decomposition is then used to rapidly resolve the equation system. This method also relies on optical flow information. So RANSAC algorithm is integrated to deal with large proportion optical flow outliers. The third estimation uses flow norm and scene structure as input to compute FOE. It takes advantages of the inverse "c-velocity" concept. From the original "c-velocity" approach, we construct "c-velocity" images on which an object (eg. buildings, road, or obstacles) is projected as a straight line when the flow norm and FOE information is correct. We further proved that this straight line will disperse with wrong FOE position. The worst the FOE position is, the larger this dispersion will be. Using this property, the proposed FOE estimation method computes FOE by minimizing the dispersion of projection representation of an object on the "c-velocity" image. This method is also tested both on the synthetic toy examples and real images. An advantage of this method is that it is possible to integrate the original "c-velocity" approach for an iterative FOE estimation and scene structure extraction.

As the "c-velocity" approach deals with three different types of objects (buildings, road, obstacles) and these objects are approximated by the planes with three different orientations (lateral, horizontal and frontal planes). The computation of "c-values" is very different for each type of plane, there is for each type of plane a "c-velocity" image.

And an object is projected as a straight line only on the corresponding "c-velocity" image. Their projections on other two "c-velocity" images are considered as background noise. In object detection process, we should know, for each pixel, if it is from an object and which type of object it belongs to. So each pixel will vote in all three "c-velocity" images. In this case, the potential building lines in the corresponding "c-velocity" image are perturbed by the ground noises from the votes of other types of objects (eg. road, obstacles). The line detection will become very difficult in the case of very large noises. To deal with this intra-perturbation, we adapt a histogram splitting method to iteratively deal with object detection in different "c-velocity" images. As a line in the "c-velocity" image is transformed into a peak in the Hough histogram, the intra-perturbation can easily make the peak submerged when the peak significance is small. So we should at first extract the most easily detectable line from three "c-velocity" images by finding the most significant peak from three Hough histograms. Then the corresponding object is extracted according to this peak and their contributions in the three histograms are also removed. Since this is the most significant peak, the vote numbers from this object on its own "c-velocity" image is of course very large, and the corresponding noises in other two "c-velocity" images are also huge. So removing their contributions will facilitate the detection of remaining objects. We do this extraction iteratively until no significant peak is detected.

6.2 Future works

Despite of these encouraging results, our works still leave many tracks to be continued.

First, the combination of the "v-velocity" and "v-disparity" approaches will be very interesting. As the road pixels in the same row of the image have the same disparity value and also the same velocity value v. The "v-velocity" method can on one hand confirm the detection from the "v-disparity" image and on the other hand introduce motion information for the mobile object detection.

For the proposed optical flow estimation method, only the road model (horizontal plane) is tested for the estimation. We believe that any other models (buildings or obstacles) can be also applied with this method.

For the inverse "c-velocity" FOE estimation approach, if we combine it with the original "c-velocity" approach, we can iteratively detect the FOE estimation and the scene structure extraction. For example, with an initial FOE position (from a guess or from the estimation of the previous image), we can give a rough extraction of the scene structure, then using the detected scene planes, we can recompute FOE, this estimated FOE

is proved closer to the real FOE, so using this FOE, the scene structure will be better extracted. We do this iteratively until the FOE position is fixed or the extracted scene planes are the same with the previous extraction results.

Finally, a hardware implementation is envisaged to mounted on an intelligent vehicle for the real test. As the construction and the vote accumulation of the "c-velocity" images are independents for the different object types, we believe that a hardware level parallelism is not difficult to realize.

A

Image Formation Geometry and Radiometry

When the lights are emitted towards a surface from the light source, the reflection (specular or diffuse) in the surface makes the lights bounce off the surface, then thest lights enter the camera lense and converged to the photographic film or the Charge Coupled Device (CCD) and finally turn into a photo (image).

So we can discuss the image formation from two aspects : We can exploit the geometric relationship between the camera and the object surface by supposing a camera model. We can also consider tha radiometric apsect by discussing the relation among the amounts of light energy emitted from light sources, reflected from surfaces and registered by sensors.

A.1 Relative Camera and Object Geometry

When Considering a perspective camera model, it means that the geometric distortions or burring of unfocused objects caused by lenses and finite sized apertures should be ignored. In fact these effects are so small that they can be neglected when a high quality camera is used. So we use this model to depict the 3D scene form a camera by taking advantage of its simplicity and also to keep its reality.

From Equation A.1, the model consists of a image plane π and the projection center **O**. The distance between π and **O** is the focal length *f*. The line through **O** and perpendicular to π is the optical axis, which coincides with the axis OZ. For a 3D point **P**(X, Y, Z), its projection in the image plane **p** can be expressed by

$$p = \begin{pmatrix} x \\ y \end{pmatrix} = f \begin{pmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{pmatrix} \tag{A.1}$$

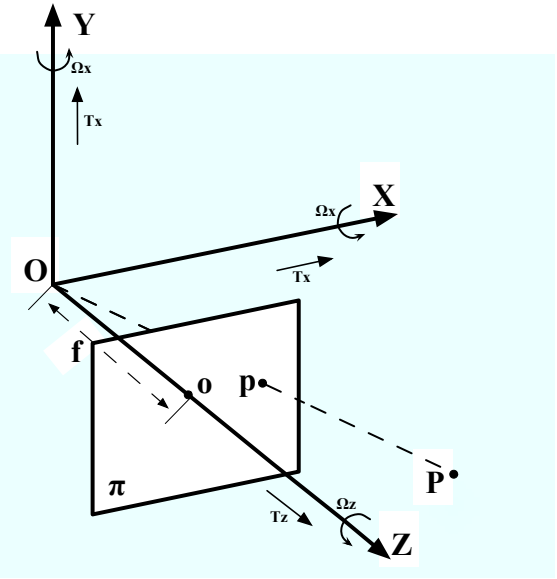


FIGURE A.1: The coordinate system using a perspective camera model

In the case of rigid motion, supposing a camera moves with a translational instantaneous velocity $\mathbf{T}$=(T_x, T_y, T_z) and a rotational instantaneous velocity $\Omega = (\Omega_x, \Omega_y, \Omega_z)$, then the static point $\mathbf{P}$ moves relative to the camera with a velocity V given by

$$\mathbf{V} = \frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} = \begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = -\mathbf{T} - \boldsymbol{\Omega} \times \mathbf{P} \tag{A.2}$$

$$\Longleftrightarrow \begin{cases} \dot{X} = -T_X - \Omega_Y Z + \Omega_Z Y \\ \dot{Y} = -T_Y - \Omega_Z X + \Omega_X Z \\ \dot{Z} = -T_Z - \Omega_X Y + \Omega_T X \end{cases} \tag{A.3}$$

We can compute the 2D velocity $\mathbf{v}$=($\dot{x}$, $\dot{y}$) of the image point $\mathbf{p}$=(x, y) by the derivation of Equation A.1.

$$\begin{cases} \dot{x} = f\frac{\dot{X}Z - \dot{Z}X}{Z^2} = \dot{X}\frac{f}{Z} - \dot{Z}\frac{x}{Z} \\ \dot{y} = f\frac{\dot{Y}Z - \dot{Z}Y}{Z^2} = \dot{Y}\frac{f}{Z} - \dot{Z}\frac{y}{Z} \end{cases} \tag{A.4}$$

Then the 3D motion ($\dot{X}$, $\dot{Y}$, $\dot{Z}$) can be substituted by adding the Equation A.3 to Equation A.4 :

$$\begin{cases} \dot{x} = \frac{f}{Z}(-T_X - \Omega_Y Z + \Omega_Z Y) - \frac{x}{Z}(-T_Z - \Omega_X Y + \Omega_Y X) \\ \dot{y} = \frac{f}{Z}(-T_Y - \Omega_Z X + \Omega_X Z) - \frac{y}{Z}(-T_Z - \Omega_X Y + \Omega_Y X) \end{cases} \tag{A.5}$$

Reuse the Equation A.1 into Equation A.5, the 2D motion can be expressed by the focal length f, the image point $\mathbf{p}$(x, y), the 3D motion parameters $\mathbf{T}$, Ω and the depth information Z (see Equation B.1).

$$\begin{cases} \dot{x} = \frac{xy}{f}\Omega_X - (\frac{x^2}{f} + f)\Omega_Y + y\Omega_Z + \frac{xT_Z - fT_X}{Z} \\ \dot{y} = (\frac{y^2}{f} + f)\Omega_X - \frac{xy}{f}\Omega_Y - x\Omega_Z + \frac{yT_Z - fT_Y}{Z} \end{cases} \tag{A.6}$$

From the equation, the motion field is the sum of two component, one of which depends only on the translation (see Equation A.7), the other only on the rotation (see A.8).

$$\begin{cases} u^T = \frac{xT_Z - fT_X}{Z} \\ v^T = \frac{yT_Z - fT_Y}{Z} \end{cases} \tag{A.7}$$

$$\begin{cases} u^\Omega = \frac{xy}{f}\Omega_X - (\frac{x^2}{f} + f)\Omega_Y + y\Omega_Z \\ v^\Omega = (\frac{y^2}{f} + f)\Omega_X - \frac{xy}{f}\Omega_Y - x\Omega_Z \end{cases} \tag{A.8}$$

A.2 Illumination and Surface Photometrics

Consdering a surface reflectance model (see Fig. A.2), the incident light enters the surface from an infinite light source, where the vector $\mathbf{I}$ defines the source intensity and the incident light direction. The scene radiance $\mathbf{L_r}$ describes the amount of the light radiated from the surface ($Wm^{-2}sr^{-1}$). It depends on the surface material and also the irradiance - the amount of the light falling on the surface (Wm^{-2}). The mathmatical representation is writen by

$$f(\theta_i, \theta_r) = \frac{L_r(\theta_r)}{L_i(\theta_i)} \tag{A.9}$$

$f(\theta_i, \theta_r)$ is called the bidirectional reflectance distribution function (BRDF). θ_i and θ_r are the incident angle and the emergent angle respectively. $\mathbf{n}$ is the surface normal vector.

For a Lambertian surface s, it reflects light with equal intensity in all directions. Its BRDF is a constant value as $\frac{\rho}{\pi}$, where ρ is the surface's albedo. Also considering an infinite light source, the scene radiance of the surface can be writen by Equation A.10.

$$L_r = \frac{\rho}{\pi}L_i(\theta_i) = \frac{\rho}{\pi}\|I\|cos(\theta_i) = \frac{\rho}{\pi}\mathbf{I}^T\mathbf{n} \tag{A.10}$$

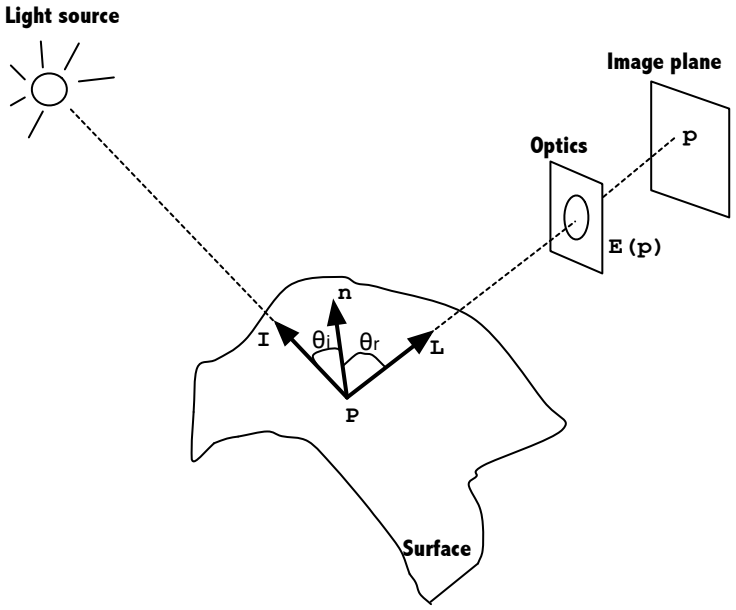


FIGURE A.2: Surface reflectance schema

After obtainning the equation of the scene radiance L_r, the image irradiance can be computed from the scene radiance. As the image irradiance is the power of the light per unit area at each point (x, y) in the image plane, we can consider it also as the image brightness pattern $E(x, y)$ at the point (x, y). In the case of the pinhole camera approximation, the camera is thought to have an inifinite small aperture, the image radiance E(p) is equal to the corresponding scene radiance $L_r(p)$.

$$E(p) = L_r(p) = \frac{\rho}{\pi}\mathbf{I}^T\mathbf{n} \tag{A.11}$$

B

Two-dimensional velocity (u, v) presentation of a plan

B.1 Two-dimensional velocity expression of a general plane

From Appendix A, we obtained the 2D velocity expression of a general 3D point **P**=(X, Y, Z) with a relative translational motion $\mathbf{T} = (T_X, T_Y, T_Z)$ and a rotational motion $\mathbf{\Omega} = (\Omega_X, \Omega_Y, \Omega_Z)$.

$$\begin{cases} \dot{x} = \frac{xy}{f}\Omega_X - (\frac{x^2}{f} + f)\Omega_Y + y\Omega_Z + \frac{xT_Z - fT_X}{Z} \\ \dot{y} = (\frac{y^2}{f} + f)\Omega_X - \frac{xy}{f}\Omega_Y - x\Omega_Z + \frac{yT_Z - fT_Y}{Z} \end{cases} \tag{B.1}$$

On the assumption of the plane-assimilated objects in an urban road situation, we look forward to the 2D velocity expression of a general plane. Suppose now that the camera is observing a planar surface of equation $\mathbf{n}^T\mathbf{P} = d$, where $\mathbf{n} = (n_X, n_Y, n_Z)$ is the unit vector normal to the plane, d is the distance "plane to origin" and **P**(X, Y, Z) represents a general point in the plane.

Since the plane equation is represented by

$$n_X X + n_Y Y + n_Z Z = d \tag{B.2}$$

The depth information Z of a point in this plane is then written by

$$\begin{aligned} \frac{1}{Z} &= \frac{1}{d}(n_X\frac{X}{Z} + n_Y\frac{Y}{Z} + n_Z) \\ &= \frac{1}{fd}(n_X x + n_Y y + n_Z) \end{aligned} \tag{B.3}$$

We replace $\frac{1}{Z}$ in Equation B.1 by Equation B.3, then we can get the 2D velocity expression of a plane :

$$\begin{cases} \dot{x} = \frac{xy}{f}\Omega_X - (\frac{x^2}{f} + f)\Omega_Y + y\Omega_Z + \frac{1}{fd}(n_X x + n_Y y + n_Z)(xT_Z - fT_X) \\ \dot{y} = (\frac{y^2}{f} + f)\Omega_X - \frac{xy}{f}\Omega_Y - x\Omega_Z + \frac{1}{fd}(n_X x + n_Y y + n_Z)(yT_Z - fT_Y) \end{cases} \tag{B.4}$$

We rearrange Equation B.4 as an expression of polynomial with variables x, y and 8 coefficients a_1 to a_8 ($u = \dot{x}$, $v = \dot{y}$) :

$$\begin{cases} u = \frac{1}{fd}(a_1 x^2 + a_2 xy + a_3 fx + a_4 fy + a_5 f^2) \\ v = \frac{1}{fd}(a_1 xy + a_2 y^2 + a_6 fy + a_7 fx + a_8 f^2) \end{cases} \tag{B.5}$$

$$\begin{aligned} a_1 &= -d\Omega_Y + T_Z n_X \quad & a_2 &= d\Omega_X + T_Z n_Y \\ a_3 &= T_Z n_Z - T_X n_X \quad & a_4 &= d\Omega_Z - T_X n_Y \\ a_5 &= -d\Omega_Y - T_X n_Z \quad & a_6 &= T_Z n_Z - T_Y n_Y \\ a_7 &= -d\Omega_Z - T_Y n_X \quad & a_8 &= d\Omega_X - T_Y n_Z \end{aligned}$$

B.2 Motion field expressions for three specifical planes

In our study, we select three most significant orientations of plane as they can assimilate almost all the important objects in an urban road situation :

- Horizontal plane(road)
- Lateral planes(buildings)
- Frontal planes(obstacles)
  - Fleeing/approaching obstacles (vehicles)
  - Crossing obstacles (pedestrians)

We will compute in the rest part of this section the 2D velocity expression corresponding to different types of plane. Here we consider only the translational motions of the camera, which means that $\Omega_X = \Omega_Y = \Omega_Z = 0$.

B.2.1 The case of a horizontal plane

The road can be usually simplified as horizontal plane, in this case we have

$$\mathbf{n} = (0, 1, 0) \quad \mathbf{\Omega} = (0, 0, 0) \tag{B.6}$$

then the 8 coefficients can be computed as

$$\begin{aligned} a_1 &= 0 \quad a_2 = T_Z \\ a_3 &= 0 \quad a_4 = -T_X \\ a_5 &= 0 \quad a_6 = -T_Y \\ a_7 &= 0 \quad a_8 = 0 \end{aligned} \tag{B.7}$$

Finally, the 2D velocity equation of a static horizontal plane is expressed as

$$\begin{cases} u = \frac{T_Z y}{fd}(x - f\frac{T_X}{T_Z}) \\ v = \frac{T_Z y}{fd}(y - f\frac{T_Y}{T_Z}) \end{cases} \tag{B.8}$$

B.2.2 The case of a lateral plane

The lateral planes are assimilated to the lateral buildings in each side of the road :

$$\mathbf{n} = (1, 0, 0) \quad \mathbf{\Omega} = (0, 0, 0) \tag{B.9}$$

then the 8 coefficients can be computed as

$$\begin{aligned} a_1 &= T_Z \quad & a_2 &= 0 \\ a_3 &= -T_X \quad & a_4 &= 0 \\ a_5 &= 0 \quad & a_6 &= 0 \\ a_7 &= -T_Y \quad & a_8 &= 0 \end{aligned} \tag{B.10}$$

Finally, the 2D velocity equation of a static lateral plane is expressed as

$$\begin{cases} u = \frac{T_Z x}{fd}(x - f\frac{T_X}{T_Z}) \\ v = \frac{T_Z x}{fd}(y - f\frac{T_Y}{T_Z}) \end{cases} \tag{B.11}$$

B.2.3 The case of a frontal plane

The difference of frontal planes from other planes is that they are considered as moving obstacles, not static objects. In this way, we should count their own velocities and rewrite the 2D velocity equations.

$\mathbf{T}$, Ω in Equation B.5 are directly camera motion information since we suppose that the observing objects are static, which means they should be rather considered as relative motion compare to the objects :

$$(\mathbf{T_r}, \mathbf{\Omega_r}) = \begin{cases} (\mathbf{T} - \mathbf{T_{obj}}, \mathbf{\Omega} - \mathbf{\Omega_{obj}}) & \text{if the observing object is moving} \\ (\mathbf{T}, \mathbf{\Omega}) & \text{if the observing object is static} \end{cases} \tag{B.12}$$

Here we should distinguish two different frontal planes : the fleeing/approaching planes and the crossing planes. The formers are usually modeled as the moving vehicles on the road, and are supposed to have only longitudinal motion ($T_{obj} = (0,\ 0,\ T_{Z1})$. While the crossing planes, which are considered as crossing pedestrians, are supposed to have only translational motion in X direction ($T_{obj} = (T_{X1},\ 0,\ 0)$.

As a consequence, for the fleeing/approaching planes we have :

$$\begin{cases} \mathbf{n} = (0, 0, 1) \\ \mathbf{\Omega} = (0, 0, 0) \\ T_Z \implies T_Z + T_{Z1} \end{cases} \tag{B.13}$$

After substituting these into the 8 coefficients, we get the final velocity equation for the the fleeing/approaching planes :

$$\begin{cases} u = \frac{T_Z + T_{Z1}}{d}(x - f\frac{T_X}{T_Z + T_{Z1}}) \\ v = \frac{T_Z + T_{Z1}}{d}(y - f\frac{T_Y}{T_Z + T_{Z1}}) \end{cases} \tag{B.14}$$

and the 2D velocity equations for the crossing planes are :

$$\begin{cases} u = \frac{T_Z}{d}(x - f\frac{T_X + T_{X1}}{T_Z}) \\ v = \frac{T_Z}{d}(y - f\frac{T_Y}{T_Z}) \end{cases} \tag{B.15}$$

C

FOE shift effection for the iso-velocity curves

In order to quantificationally discuss the FOE shift effection on the representation of a plane in the "c-velocity" image, we first define a set of points $\mathcal{S}$ from a same plane (ex. a road plane), then we define "iso-velocity" curve from the subset of these points. This subset is denoted as $\mathcal{C}$, then using the equations in Table 2.4 we can express it as

$$\mathcal{C} = \{m(x,y)|\frac{T_Z}{fd_r} * |y|\sqrt{(x-x_{FOE})^2 + (y-y_{FOE})^2} = w\} \tag{C.1}$$

Since w is constant for the points in the subset $\mathcal{C}$, and $\frac{T_Z}{fd_r}$ is also constant for all the points in the set $\mathcal{C}$, we can rewrite Equation C.1 by

$$\mathcal{C} = \{m(x,y)||y|\sqrt{(x-x_{FOE})^2 + (y-y_{FOE})^2} = K\} \tag{C.2}$$

where $K = \frac{fd_r w}{T_Z}$ is constant for the all the points in the subset $\mathcal{C}$. If we replace the x by a function of y, we get

$$\mathcal{C} = \{m(x_{FOE} \pm \sqrt{\frac{K^2}{y^2} - (y-y_{FOE})^2}, y)\} \tag{C.3}$$

Consider now there is a shift of the FOE position, if the real FOE is defined as (x_{FOE}, y_{FOE}), then a shift in both direction will be added in (FOE = ($x_{FOE}+\Delta x, y_{FOE}+\Delta y$)). The *c-value* of a point m in the subset $\mathcal{C}$ is rewriten as

$$c_r(m) = \|y\|\sqrt{(x-x_{FOE}-\Delta x)^2 + (y-y_{FOE}-\Delta y)^2} \tag{C.4}$$

Since m is in the subset $\mathcal{C}$, Equation C.3 is valid for m. So we can replace the x in Equation C.4 by the expression of y in Equation C.3 :

$$c_r(m) = \|y\| \sqrt{(\pm\sqrt{\frac{K^2}{y^2} - (y - y_{FOE})^2} - \Delta x)^2 + (y - y_{FOE} - \Delta y)^2} \tag{C.5}$$

When $\Delta x = \Delta y = 0$, Equation C.5 becomes $c_r(m) = K$. This means that all the points located in the same iso-velocity curve have the same *c-value*. Here we return to the base of the direct "c-velocity".

To discuss the influence of the Δx and Δy, we compute the $c_r(m)^2$:

$$c_r(m) = K^2 + \Delta x^2 \mp 2\Delta x \sqrt{\frac{K^2}{y^2} - (y - y_{FOE})^2} + \Delta y^2 - 2\Delta y(y - y_{FOE}) \tag{C.6}$$

From Equation C.6, the dispersion of the *c-values* for all the points in an iso-velocity curve is due to the two terms.

$$\Delta x^2 \mp 2\Delta x \sqrt{\frac{K^2}{y^2} - (y - y_{FOE})^2} \tag{C.7}$$

And

$$\Delta y^2 - 2\Delta y(y - y_{FOE}) \tag{C.8}$$

The first one (Equation C.7) depends solely on Δx and the second one (Equation C.7) depends solely on Δy. Because of the uniqueness of the Laurent expansion, there is no interaction between Δx and Δy, which means that the computation of Δx and Δy can be executed seprately.

Bibliographie

[1] URL http://www.ti.com/lsds/ti/automotive/processors/adas/overview.page.

[2] R. Labayrade, D. Aubert, and J. P Tarel. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 646–651 vol.2, 2002.

[3] Ron Ohlander, Keith Price, and D. Raj Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8 (3) :313 – 333, 1978.

[4] Michael Darms, Paul Rybski, Christopher R. Baker, and Christopher Urmson. Obstacle detection and tracking for the urban challenge. *IEEE Transactions on Intelligent Transportation Systems*, 10(3) :475–485, September 2009.

[5] J. Laneurit, C. Blanc, R. Chapuis, and L. Trassoudaine. Multisensorial data fusion for global vehicle and obstacles absolute positioning. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 138–143, June 2003.

[6] A.F. Garcia-Fernandez, L. Hammarstrand, M. Fatemi, and L. Svensson. Bayesian road estimation using onboard sensors. *Intelligent Transportation Systems, IEEE Transactions on*, 15(4) :1676–1689, Aug 2014.

[7] M. Perrollaz, J.-D. Yoder, A. Ngre, A. Spalanzani, and C. Laugier. A visibility-based approach for occupancy grid computation in disparity space. *Intelligent Transportation Systems, IEEE Transactions on*, 13(3) :1383–1393, Sept 2012.

[8] Thien-Nghia Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M. Meinecke. Stereo-camera-based urban environment perception using occupancy grid and object tracking. *Intelligent Transportation Systems, IEEE Transactions on*, 13(1) : 154–165, March 2012.

[9] C.G. Keller, M. Enzweiler, M. Rohrbach, D. Fernandez Llorca, C. Schnorr, and D.M. Gavrila. The benefits of dense stereo for pedestrian detection. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4) :1096–1106, Dec 2011.

[10] J. Fritsch, T. Kuhnl, and F. Kummert. Monocular road terrain detection by combining visual and spatial information. *Intelligent Transportation Systems, IEEE Transactions on*, 15(4) :1586–1596, Aug 2014.

[11] R. Gopalan, Tsai Hong, M. Shneier, and R. Chellappa. A learning approach towards detection and tracking of lane markings. *Intelligent Transportation Systems, IEEE Transactions on*, 13(3) :1088–1098, Sept 2012.

[12] J.M.A. Alvarez and A.M. opez. Road detection based on illuminant invariance. *Intelligent Transportation Systems, IEEE Transactions on*, 12(1) :184–193, March 2011.

[13] N. Onkarappa and A. Domingo Sappa. Speed and texture : An empirical study on optical-flow accuracy in adas scenarios. *Intelligent Transportation Systems, IEEE Transactions on*, 15(1) :136–147, Feb 2014.

[14] F. Garcia, P. Cerri, A. Broggi, A. de la Escalera, and J.M. Armingol. Data fusion for overtaking vehicle detection based on radar and optical flow. In *IEEE Intelligent Vehicle Symposium (IV)*, 2012.

[15] U. Franke and S. Heinrich. Fast obstacle detection for urban traffic situations. *IEEE Transactions on Intelligent Transportation Systems*, 3(3) :173–181, 2002.

[16] A.I. Comport, E. Malis, and P. Rives. Real-time quadrifocal visual odometry. *Int. J. Rob. Res.*, 29(2-3) :245–266, February 2010.

[17] A. Bensrhair, M. Bertozzi, A. Broggi, P. Miche, S. Mousset, and G. Toulminet. A cooperative approach to vision-based vehicle detection. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 207–212, 2001.

[18] Thomas Kalinke, Christos Tzomakas, and Werner V. Seelen. A texture-based object detection and an adaptive model-based classification. In *in Procs. IEEE Intelligent Vehicles Symposium98*, pages 341–346, 1998.

[19] Shashi D. Buluswar and Bruce A. Draper. Color machine vision for autonomous vehicles. *Engineering Applications of Artificial Intelligence*, 11(2) :245 – 256, 1998.

[20] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

[21] Chang Huang and R. Nevatia. High performance object detection by collaborative learning of joint ranking of granules features. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 41–48, June 2010.

[22] P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 734–741 vol.2, Oct 2003.

[23] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591, Jun 1991.

[24] V.N. Vapnik. An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5) :988–999, Sep 1999.

[25] S. Munder and D.M. Gavrila. An experimental study on pedestrian classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(11) :1863–1868, Nov 2006.

[26] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning : data mining, inference and prediction.* Springer, 2 edition, 2008.

[27] Joo Gama and Pavel Brazdil. Cascade generalization. *Machine Learning*, 41(3) : 315–343, 2000.

[28] Trung-Dung Vu, J. Burlet, and O. Aycard. Grid-based localization and online mapping with moving objects detection and tracking : new results. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 684–689, June 2008.

[29] S. Nedevschi, S. Bota, and C. Tomiuc. Stereo-based pedestrian detection for collision-avoidance applications. *Intelligent Transportation Systems, IEEE Transactions on*, 10(3) :380–391, Sept 2009.

[30] S.A. Rodrguez F, V. Fremont, P. Bonnifait, and V. Cherfaoui. Visual confirmation of mobile objects tracked by a multi-layer lidar. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 849–854, Sept 2010.

[31] Nicolas Soquet, Mathias Perrollaz, Raphael Labayrade, and Didier Aubert. D. ; free space estimation for autonomous navigation. In *In Proceedings of the 5th International Conference on Computer Vision System*, pages 1–6, 2007.

[32] John Y Aloimonos. Perspective approximations. *Image and Vision Computing*, 8 (3) :179 – 192, 1990.

[33] Conrad J. Poelman and Takeo Kanade. A paraperspective factorization method for shape and motion recovery. 19 :97–108, 1997.

[34] Tim Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam) : part ii. *Robotics Automation Magazine, IEEE*, 13(3) :108–117, Sept 2006.

[35] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4293–4299, May 2009.

[36] Sanbao Xu and Per-Erik Danielsson. Robust estimation of focus of expansion and depth from high confidence optical flow. In *MVA'92*, pages 105–108, 1992.

[37] Didi Sazbon, Hctor Rotstein, and Ehud Rivlin. Finding the focus of expansion and estimating range using optical flow images and a matched filter. *Machine Vision and Applications*, 15(4) :229–236, 2004.

[38] F.C. Wu, L. Wang, and Z.Y. Hu. {FOE} estimation : Can image measurement errors be totally corrected by the geometric method ? *Pattern Recognition*, 40(7) : 1971 – 1980, 2007.

[39] Q.-T. Luong and O. D. Faugeras. Camera calibration, scene motion and structure recovery from point correspondences and fundamental matrices. *IJCV*, 22 :261–289, 1997.

[40] R.I. Hartley. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6) :580–593, Jun 1997.

[41] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 208(1173) :385–397, 1980.

[42] Gideon P. Stein, Ofer Mano, and Amnon Shashua. A robust method for computing vehicle ego-motion. In *In IEEE Intelligent Vehicles Symposium (IV2000*, 2000.

[43] Michal Irani, Benny Rousso, and Shmuel Peleg. Recovery of ego-motion using region alignment. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 19 :268–272, 1997.

[44] S. Heinrich. Fast obstacle detection using flow/depth constraint. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 658–665 vol.2, June 2002.

[45] Uwe Franke, Clemens Rabe, Hernn Badino, and Stefan Gehrig. 6d-vision : Fusion of stereo and motion for robust environment perception. In WalterG. Kropatsch, Robert Sablatnig, and Allan Hanbury, editors, *Pattern Recognition*, volume 3663 of *Lecture Notes in Computer Science*, pages 216–223. Springer Berlin Heidelberg, 2005.

[46] F. Dornaika and R. Chung. Cooperative stereomotion : Matching and reconstruction. *Computer Vision and Image Understanding*, 79(3) :408 – 427, 2000.

[47] Christophe Braillon, C. Pradalier, K. Usher, JamesL. Crowley, and Christian Laugier. Occupancy grids from stereo and optical flow data. In Oussama Khatib, Vijay Kumar, and Daniela Rus, editors, *Experimental Robotics*, volume 39 of *Springer Tracts in Advanced Robotics*, pages 367–376. Springer Berlin Heidelberg, 2008.

[48] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[49] P. Lenz, J. Ziegler, A. Geiger, and M. Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 926–932, June 2011.

[50] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *Int. J. Comput. Vision*, 72(2) :179–193, April 2007.

[51] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *Proceedings of the 10th European Conference on Computer Vision : Part I*, ECCV '08, pages 739–751, Berlin, Heidelberg, 2008. Springer-Verlag.

[52] Manolis I. A. Lourakis and Stelios C. Orphanoudakis. Visual detection of obstacles assuming a locally planar ground. pages 527–534, 1997.

[53] Todd Williamson. *A High-Performance Stereo Vision System for Obstacle Detection.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 1998.

[54] Zhongfei Zhang, R. Weiss, and A.R. Hanson. Qualitative obstacle detection. pages 554–559, 1994.

[55] Dieter Koller, Quang Luong, and Jitendra Malik. Binocular stereopsis and lane marker flow for vehicle navigation :. Technical report, Berkeley, CA, USA, 1994.

[56] A. Wedel, H. Badino, C. Rabe, H. Loose, U. Franke, and D. Cremers. B-spline modeling of road surfaces with an application to free-space estimation. *Intelligent Transportation Systems, IEEE Transactions on*, 10(4) :572–583, Dec 2009.

[57] S. Bouchafa and B. Zavidovique. C-velocity : A cumulative frame to segment objects from egomotion. *Pattern Recognition and Image Analysis*, 19(4) :583–590, 2009.

[58] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814 vol. 2, June 2005.

[59] G. Bradski. Opencv computer vision library. *Dr. Dobb's Journal of Software Tools*, 2000.

[60] J.Y. Bouguet. Matlab camera calibration toolbox. 2000.

[61] Gruyer D., Royre C., du Lac N., Michel G., and Blosseville J.M. Sivic and rt-maps interconnected platforms for the conception and the evaluation of driving assistance systems. In *Proceedings of the ITS World Congress*, 2006.

[62] Zhencheng Hu, F. Lamosa, and K. Uchimura. A complete u-v-disparity study for stereovision based 3d driving environment analysis. In *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*, pages 204–211, June 2005.

[63] M. Gouiffs, A. Patri, and M. Vasiliu. Robust obstacles detection and tracking using disparity for car driving assistance. In *27th Conference on Intelligent Robots and Computer Vision : Algorithms and Techniques*, volume 7539, 2010.

[64] Samia Bouchafa and Bertrand Zavidovique. c-velocity : A flow-cumulating uncalibrated approach for 3d plane detection. *International Journal of Computer Vision*, 97(2) :148–166, 2012. ISSN 0920-5691.

[65] C. Fermuller and Y. Aloimonos. Global rigidity constraints in image displacement fields. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 245–250, 1995.

[66] Trucco, Emanuele, Verri, and Alessandro. *Introductory Techniques for 3D Computer Vision.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[67] Michael J. Brooks, Wojciech Chojnacki, and Luis Baumela. Determining the egomotion of an uncalibrated camera from instantaneous optical flow. *Journal of the Optical Society of America A*, 14 :2670–2677.

[68] Tina Y. Tian, Carlo Tomasi, and David J. Heeger. Comparison of approaches to egomotion computation. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, Washington, DC, USA, 1996. IEEE Computer Society.

[69] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, 1981.

[70] Qiong Nie, S. Bouchafa, and A. Merigot. Voting spaces cooperation for 3d plane detection from monocular image sequences. In *Image Processing Theory, Tools and Applications (IPTA), 2012 3rd International Conference on*, pages 135–140, Oct 2012.

[71] S Ullman. The interpretation of structure from motion. *Proc R Soc Lond B Biol Sci*, 203(1153) :405–426, 1979 Jan 15. ISSN 0080-4649 (Print).

[72] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 722–729, 1999.

[73] F. Huguet and F. Devernay. A variational method for scene flow estimation from stereo sequences. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7, Oct 2007.

[74] T. Basha, Y. Moses, and N. Kiryati. Multi-view scene flow estimation : A view centered variational approach. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1506–1513, June 2010.

[75] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. *Computer Vision, IEEE International Conference on*, 0 :1377–1384, 2013.

[76] *The perception of the visual world.* Cambridge, Massachusetts : Houghton Mifflin, 1950.

[77] M. Kharbat, N. Aouf, A. Tsourdos, and B. White. Robust brightness description for computing optical flow. In *Proc. BMVC*, pages 46.1–46.10, 2008.

[78] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, 1981.

[79] Daniel Cremers and Stefano Soatto. Motion competition : A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3) :249–265, 2005.

[80] M. Unger, M. Werlberger, T. Pock, and H. Bischof. Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1878–1885, June 2012.

[81] Andrs Bruhn, Joachim Weickert, and Christoph Schnrr. Lucas/kanade meets horn/schunck : Combining local and global optic flow methods. *International Journal of Computer Vision*, 61 :211–231, 2005.

[82] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(13) :185 – 203, 1981.

[83] Li Xu, Jiaya Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[84] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. An improved algorithm for tv-l1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, volume 5604 of *Lecture Notes in Computer Science*, pages 23–45. Springer Berlin Heidelberg, 2009.

[85] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1663–1668, Sept 2009.

[86] Yana Mileva, Andrs Bruhn, and Joachim Weickert. Illumination-robust variational optical flow with photometric invariants. In FredA. Hamprecht, Christoph Schnrr, and Bernd Jhne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 152–162. Springer Berlin Heidelberg, 2007.

[87] J. van de Weijer and T. Gevers. Robust optical flow from photometric invariants. In *Image Processing, 2004. ICIP '04. 2004 International Conference on*, volume 3, pages 1835–1838 Vol. 3, Oct 2004.

[88] Thomas Brox, Andrs Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In Toms Pajdla and Ji Matas, editors, *Computer Vision - ECCV 2004*, volume 3024 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin Heidelberg, 2004.

[89] D.W. Murray and Bernard F. Buxton. Scene segmentation from visual motion using global optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(2) :220–228, March 1987.

[90] M.J. Black and P. Anandan. Robust dynamic motion estimation over time. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 296–302, Jun 1991.

[91] Hans-Hellmut Nagel and Wilfried Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(5) : 565–593, Sept 1986.

[92] Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic huber-l1 optical flow. In *Proceedings of the British Machine Vision Conference (BMVC)*, London, UK, September 2009. to appear.

[93] Levi Valgaerts, Andrs Bruhn, and Joachim Weickert. A variational model for the joint recovery of the fundamental matrix and the optical flow. In Gerhard Rigoll, editor, *Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 314–324. Springer Berlin Heidelberg, 2008.

[94] A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers. Duality tv-l1 flow with fundamental matrix prior. In *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*, pages 1–6, Nov 2008.

[95] Antoni Buades, Triet Le, Jean-Michel Morel, and Luminita Vese. Cartoon+Texture Image Decomposition. *Image Processing On Line*, 1, 2011.

[96] Michael J. Black and P. Anandan. The robust estimation of multiple motions : Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1) :75 – 104, 1996.

[97] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th DAGM Conference on Pattern Recognition*, pages 214–223, Berlin, Heidelberg, 2007. Springer-Verlag.

[98] Deqing Sun, S. Roth, and M.J. Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439, June 2010.

[99] Qiong Nie, Samia Bouchafa, and Alain Merigot. Model-based optical flow for large displacements and homogeneous regions. In *IEEE International Conference on Image Processing, ICIP 2013, Melbourne, Australia, September 15-18, 2013*, pages 3865–3869, 2013.

[100] M. Gleicher. Projective registration with difference decomposition. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 331–337, Jun 1997.

[101] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(6) :681–685, Jun 2001.

[102] Heung-Yeung Shum and Richard Szeliski. Systems and experiment paper : Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2) :101–130, 2000.

[103] Simon Baker and Iain Matthews. Lucas-kanade 20 years on : A unifying framework. *Int. J. Comput. Vision*, 56(3) :221–255, February 2004.

[104] G. Le Besnerais and F. Champagnat. Dense optical flow by iterative local window registration. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 1, pages I–137–40, Sept 2005.

[105] Jordi Ribas-corbera and David L. Neuhoff. On the optimal block size for block-based, motion-compensated video coders. In *in Proceedings of the SPIE Conference on Visual Communications and Image Processing*, pages 1132–1143, 1997.

[106] M.H. Chan, Y. B. Yu, and A.G. Constantinides. Variable size block matching motion compensation with applications to video coding. *Communications, Speech and Vision, IEE Proceedings I*, 137(4) :205–212, Aug 1990.

[107] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. 1981.

[108] J. Jain and A. Jain. Displacement measurement and its application in interframe image coding. *Communications, IEEE Transactions on*, 29(12) :1799–1808, Dec 1981.

[109] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block-matching motion estimation. *Image Processing, IEEE Transactions on*, 9(2) : 287–290, Feb 2000.

[110] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1) :51–59, 1996.

[111] David Young. Affin optical flow. URL http://www.mathworks.com/matlabcentral/fileexchange/27093-affine-optic-flow.

[112] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1) :43–77, 1994. ISSN 0920-5691.

[113] B McCane, K Novins, D Crannitch, and B Galvin. On benchmarking optical flow. *Computer Vision and Image Understanding*, 84(1) :126 – 143, 2001.

[114] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, MichaelJ. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1) :1–31, 2011. ISSN 0920-5691.

[115] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[116] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving ? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[117] Stephan Meister, Bernd Jahne, and Daniel Kondermann. Outdoor stereo camera system for the generation of real-world benchmark data sets. *Optical Engineering*, 51(2) :021107, 2012.

[118] URL http://www.sintel.org.

[119] URL http://www.mathworks.com/matlabcentral/fileexchange/27093-affine-optic-flow.

[120] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black. Lessons and insights from creating a synthetic optical flow benchmark. In A. Fusiello et al. (Eds.), editor, *ECCV Workshop on Unsolved Problems in Optical Flow and Stereo Estimation*, Part II, LNCS 7584, pages 168–177. Springer-Verlag, October 2012.

[121] Stefan Roth and MichaelJ. Black. On the spatial statistics of optical flow. *International Journal of Computer Vision*, 74(1) :33–50, 2007.

[122] David J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A*, 4(12) :2379–2394, Dec 1987.

[123] Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24(1) :1193–1216, 2001.

[124] URL www.xbox.com/en-US/kinect.

[125] A. Giachetti, M. Campani, and V. Torre. The use of optical flow for road navigation. *IEEE Transactions on Robotics and Automation*, 14(1) :34–48, Febrary 1998.

[126] A. Giachetti and V. Torre. Refinement of optical flow estimation and detection of motion edges. In *European Conference on Computer Vision (ECCV)*, 1996.

[127] E. Meinhardt-Llopis and J. Sanchez. Horn-schunck optical flow with multi-scale strategy. preprint, 2012.

[128] J.-Y. Bouguet. Pyramidal implementation of lucas kanade feature tracker. description of the algorithm. Technical report, 2001. URL http://sourceforge.net/projects/opencvlibrary/.

[129] S. Bouchafa, A. Patri, and B. Zavidovique. Efficient plane detection from a single moving camera. In *IEEE International Conference on Image Processing (ICIP)*, 2009.

[130] S. Bouchafa and B. Zavidovique. c-velocity : A flow-cumulating uncalibrated approach for 3d plane detection. *International Jounal of Computer Vision*, 97(2) : 148–166, April 2012.

[131] A. Branca, E. Stella, G. Attolico, and A. Distante. Focus of expansion estimation by an error backpropagation neural network. *Neural Computing & Applications*, 6(3) :142–147, 1997.

[132] Ramesh Jain. Direct computation of the focus of expansion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-5(1) :58–64, Jan 1983.

[133] Shahriar Negahdaripour and Berthold K.P Horn. A direct method for locating the focus of expansion. *Computer Vision, Graphics, and Image Processing*, 46(3) : 303 – 326, 1989.

[134] Shahriar Negahdaripour. Direct computation of the {FOE} with confidence measures. *Computer Vision and Image Understanding*, 64(3) :323 – 350, 1996.

[135] C. Fermuller. *Basic Visual Capabilities*. PhD thesis, University of Maryland, 1993.

[136] N. Ancona and T. Poggio. Optical flow from 1d correlation : Application to a simple time-to-crash detector. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 209–214, May 1993.

[137] C. McCarthy, N. Barnes, and R. Mahony. A robust docking strategy for a mobile robot using flow field divergence. *Robotics, IEEE Transactions on*, 24(4) :832–842, Aug 2008.

[138] Masaki Fukuchi, Naotsugu Tsuchiya, and Christof Koch. The focus of expansion in optical flow fields acts as a strong cue for visual attention. *Journal of Vision*, 9 (8) :137, 2009.

[139] James Jerome Gibson. The ecological approach to visual perception. 1979. Houghton Mifflin.

[140] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[141] Sanbao Xu and Per-Erik Danielsson. Robust estimation of focus of expansion and depth from high confidence optical flow. In *MVA*, pages 105–108, 1992.

[142] Martin A. Fischler and Robert 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.

[143] Zezhi Chen, Nick Pears, John McDermid, and Thomas Heseltine. Epipole estimation under pure camera translation. In *DICTA'03*, pages 849–858, 2003.

[144] Jae Kyu Suhr, Ho Gi Jung, Kwanghyuk Bae, and Jaihie Kim. Outlier rejection for cameras on intelligent vehicles. *Pattern Recognition Letters*, 29(6) :828 – 840, 2008.

[145] I.S. McQuirk, Hae-Seung Lee, and B.K.P. Horn. An analog vlsi chip for estimating the focus of expansion. In *Solid-State Circuits Conference, 1997. Digest of Technical Papers. 43rd ISSCC., 1997 IEEE International*, pages 40–41, Feb 1997.

[146] IgnacioS. McQuirk, BertholdK.P. Horn, Hae-Seung Lee, and Jr. Wyatt, JohnL. Estimating the focus of expansion in analog vlsi. *International Journal of Computer Vision*, 28(3) :261–277, 1998.

[147] Tomoaki Teshima, Hideo Saito, and Shinji Ozawa. Ihara : estimation of foe without optical flow for vehicle lateral position detection. In *Proceedings of IAPR Conference on Machine Vision Applications*, pages 406–409, 2005.

[148] T. Brox and J. Malik. Large displacement optical flow : Descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3) :500–513, March 2011.

[149] Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(12) :87 – 119, 1995.

[150] Adrien Bak, Samia Bouchafa, and Didier Aubert. Focus of expansion localization through inverse c-velocity. In *Image Analysis and Processing ICIAP 2011*, volume 6978 of *Lecture Notes in Computer Science*, pages 484–493. Springer Berlin Heidelberg, 2011.

[151] S.A Rodriguez F, V. Fremont, and P. Bonnifait. Extrinsic calibration between a multi-layer lidar and a camera. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 214–219, Aug 2008.

[152] Luca Iocchi, Kurt Konolige, and Max Bajracharya. Visually realistic mapping of a planar environment with stereo. In *Experimental Robotics VII*, volume 271 of *Lecture Notes in Control and Information Sciences*, pages 521–532. Springer Berlin Heidelberg, 2001.

[153] P. Debye. Nherungsformeln fr die zylinderfunktionen fr groe werte des arguments und unbeschrnkt vernderliche werte des index. *Mathematische Annalen*, 67(4) : 535–558, 1909.

[154] Tjalling J. Ypma. Historical development of the newton-raphson method. *SIAM Rev.*, 37(4) :531–551, December 1995.

[155] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. J. Appl. Maths.*, II(2) :164–168, 1944.

[156] B. Rousso, S. Avidan, A Shashua, and S. Peleg. Robust recovery of camera rotation from three frames. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 796–802, Jun 1996.

[157] R. Hummel and V. Sundareswaran. Motion parameter estimation from global flow field data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15 (5) :459–476, May 1993.

[158] G. Le Besnerais and F. Champagnat. Dense optical flow by iterative local window registration. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 1, pages I–137–40, Sept 2005.