Hybrid Machine Learning and Geometric Approaches for Single RGB Camera Relocalization

Nam-Duong Duong

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

CentraleSupélec

THESE DE DOCTORAT DE

CENTRALESUPELEC
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *AST – Signal, Image, Vision*

Par
Nam Duong DUONG

Hybrid Machine Learning and Geometric Approaches for Single RGB Camera Relocalization

Thèse présentée et soutenue à Cesson-Sévigné, le 10 décembre 2019
Unité de recherche : Équipe FAST - IETR/CentraleSupélec
Thèse N° : 2019CSUP0008

Rapporteurs avant soutenance :

Tomas PAJDLA	Associate Professor - Université technique de Prague
Vincent LEPETIT	Professeur - Université de Bordeaux

Composition du Jury :

Président du jury :	Guillaume MOREAU	Professeur des Universités, École Centrale de Nantes
Rapporteur :	Tomas PAJDLA	Associate Professor - Université technique de Prague
Rapporteur :	Vincent LEPETIT	Professeur - Université de Bordeaux
Directeur de thèse :	Pierre-Yves RICHARD	Professeur, École CentraleSupélec
Co-encadrant de thèse 1 :	Catherine SOLADIÉ	Professeure assistante, École CentraleSupélec
Co-encadrant de thèse 2 :	Jérôme ROYAN	Docteur, IRT b<>com

Acknowledgements

Three years ago, I began a new journey of my life to become a doctor in computer vision. On my way, I approached and collected new knowledge of the world to broaden my eyes. More luckily, I also got to know many new friends, companions and mentors. Knowledge and friendship are the most important things I gained during my PhD in France.

First and foremost, I would like to express my sincere gratitude toward my supervisors Assoc. Prof. Catherine Soladié, Dr. Jérôme Royan, Prof. Pierre-Yves Richard for their guidance and support during the past three years. They gave me an opportunity to work on this interesting project as well as enthusiastically instructed me to efficiently walk on the path of scientific research.

I would like say thanks to my colleagues at IRT b<>com, especially the ARGO team at Immersive Iteration laboratory, together with my friends at FAST team in CentraleSupélec. I have been fortunate to work with all of you in research projects and to learn from you all the French culture and language. My special thanks to Amine Kacete, Jérémy Lacoche, Lucie Petta who have been dedicated to supporting me and creating beautiful demonstrations all together.

Last but not least, I would like to thank my family and my friends in Vietnam for their love and care. Conversations from thousands of kilometers away have always encouraged me to overcome difficulties. I am much obliged to my parents who not only gave birth to me and raised me but also inspired me to start this thesis. And I give special thanks to my wife, my darling for her patience and endless love, for having her accompanied on every road to overcome all hardships. I am blessed to have her by my side now when I write the last lines of my thesis. Finally, I would like to give my doctorate to my family.

Rennes, France, 23 August 2019
Nam Duong DUONG

Abstract

In the last few years, image-based camera relocalization becomes an important issue of computer vision applied to augmented reality, robotics as well as autonomous vehicles. Camera relocalization refers to the problematic of the camera pose estimation including both 3D translation and 3D rotation. In localization systems, camera relocalization component is necessary to retrieve camera pose after tracking lost, rather than restarting the localization from scratch. However, the classical existing camera relocalization methods store a large set of keypoints or keyframes to relocalize camera based geometric information. Consequently, memory usage as well as processing time rise with respect to the size of the models. Accordingly, machine learning approaches have been developed to tackle these constraints. Nevertheless, the limitations of machine learning approaches lie in their time-consuming training process, moderate accuracy and lack of confidence score in the estimation of each pose. Recently, hybrid methods increase considerably the accuracy. Yet, they still take more time to optimize camera pose from thousands of correspondences. Moreover, all these machine learning based methods still fail to challenge dynamic scenes with moving objects.

Given those pros and cons, this thesis aims at improving the performance of camera relocalization in terms of both runtime and accuracy as well as handling challenges of camera relocalization in dynamic environments. We present camera pose estimation based on combining multi-patch pose regression to overcome the uncertainty of end-to-end deep learning methods. To balance between accuracy and computational time of camera relocalization from a single RGB image, we propose a sparse feature hybrid methods. A better prediction in the machine learning part of our methods leads to a rapid inference of camera pose in the geometric part. To tackle the challenge of dynamic environments, we propose an adaptive regression forest algorithm that adapts itself in real time to predictive model. It evolves by part over time without requirement of re-training the whole model from scratch. When applying this algorithm to our real-time and accurate camera relocalization, we can cope with dynamic environments, especially moving objects. The experiments proves the efficiency of our proposed methods. Our method achieves results as accurate as the best state-of-the-art methods on the rigid scenes dataset. Moreover, we also obtain high accuracy even on the dynamic scenes dataset.

Résumé

Au cours des dernières années, la relocalisation de la caméra à base d'images est devenue un enjeu important de la vision par ordinateur appliquée à la réalité augmentée, à la robotique ainsi qu'aux véhicules autonomes. La relocalisation de la caméra fait référence à la problématique de l'estimation de la pose de la caméra incluant à la fois la translation 3D et la rotation 3D. Dans les systèmes de localisation, le composant de relocalisation de la caméra est nécessaire pour récupérer la pose de la caméra après le suivi perdu, plutôt que de redémarrer la localisation à partir de zéro. Cependant, les méthodes classiques de relocalisation de la caméra existantes stockent un grand nombre de points-clés ou d'images-clés pour relocaliser basées sur les informations géométriques. Par conséquent, l'utilisation de la mémoire ainsi que le temps de traitement augmentent par rapport à la taille des modèles. Donc, des approches d'apprentissage machine ont été développées pour tacler ces contraintes. Néanmoins, les limites de ces approches résident dans leur long processus d'apprentissage, leur précision modérée et leur manque de confiance dans l'estimation de chaque pose. Récemment, les méthodes hybrides augmentent la précision. Pourtant, ils prennent encore plus de temps pour optimiser la pose de la caméra à partir de milliers de correspondances. De plus, toutes ces méthodes basées sur l'apprentissage machine ne parviennent toujours pas à défier les scènes dynamiques avec des objets en mouvement.

Compte tenu de ces avantages et inconvénients, cette thèse vise à améliorer les performances de la relocalisation de la caméra en termes de temps d'exécution et de précision ainsi qu'à relever les défis de la relocalisation des caméras dans des environnements dynamiques. Nous présentons l'estimation de la pose de la caméra basée sur la combinaison de la régression de pose multi-patch pour surmonter l'incertitude des méthodes d'apprentissage profond de bout en bout. Afin d'équilibrer la précision et le temps de calcul de la relocalisation de la caméra à partir d'une seule image RVB, nous proposons une méthode hybride à caractéristiques éparses. Une meilleure prédiction dans la partie d'apprentissage automatique de nos méthodes conduit à une inférence rapide de la pose de la caméra dans la partie géométrique. Pour relever le défi des environnements dynamiques, nous proposons une forêt de régression adaptative qui s'adapte en temps réel au modèle prédictif. Il évolue en partie au fil du temps sans qu'il soit nécessaire de ré-entrainer le modèle entier à partir de zéro. En appliquant

cet algorithme à notre relocalisation de la caméra en temps réel et précise, nous pouvons faire face à des environnements dynamiques, en particulier des objets en mouvement. Les expériences prouvent l'efficacité des méthodes que nous proposons. Notre méthode permet d'obtenir des résultats aussi précis que les meilleures méthodes d'état de l'art. De plus, nous obtenons également une grande précision même sur des scènes dynamiques.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AR	Augmented Reality
ARF	Adaptive Regression Forest
BoW	Bag of Words
CNN	Convolutional Neural Network
DLT	Direct Linear Transformation
DoF	Degree of Freedom
F2P	Feature-to-Point
FC	Fully Connected
FCN	Fully Convolutional Network
GMMs	Gaussian Mixture Models
HMD	Head-Mounted Display
ICP	Iterative Closest Point
KF	Kalman Filter
LSTM	Long Short-Term Memory
MIMO	Multiple-Input and Multiple-Output
MLP	Multi-Layer Perceptron
MR	Mixed Reality

P2F Point-to-Feature

PCA Principal component Analysis

PnP Perspective-n-Point

RANSAC RANdom SAmpling Consensus

RFs Random Forests

RMSD Root Mean Squared Deviation

RNN Recurrent Neural Network

SfM Structure from Motion

SLAM Simultaneously Localization And Mapping

SVD Singular Value Decomposition

SVM Support Vector Machine

SVR Support Vector Regressor

VLAD Vector of Locally Aggregated Descriptors

VR Virtual Reality

Chapter 1

Introduction

Contents

1.1 Context

In recent years, Virtual/Augmented/Mixed Reality (VR/AR/MR), robotics, autonomous vehicles (self-driving) have become increasingly trendy in Industry 4.0.

Virtual reality (VR) is an interactive computer-generated experience taking place within a simulated environment. Unlike traditional user interfaces, VR places the user inside a fully virtual experience. Instead of viewing a screen in front of them, users are immersed and able to interact with 3D worlds. Differently from VR, **Augmented Reality (AR)** does not create a completely virtual environment. In AR, the real environment is annotated or extended with virtual objects. To be able to do this seamlessly, the virtual objects must be transformed according to the point-of-view of the person or objects pose. According to [9], an AR system must have the following three characteristics: Combines real and virtual; Interactive in real time; Registered in 3D. **Mixed reality (MR)** [121], sometimes referred to as hybrid reality, is the merging of real and virtual worlds to produce new environments and visualizations where physical and digital objects co-exist and interact in real time.

Fig. 1.1 Augmented reality for maintenance of a data center at IRT b-com.

In particular, AR is widely used in several sectors and contexts, from consumer applications to manufacturers. AR technology, including smart glasses, provides employees with a field access to the digital twin that perfectly matches the real environment. This improves productivity, quality and safety in the workplace for some tasks as follows. For assisting workers in complex tasks, the use of AR to overlay instructions onto the workspace has been proven to reduce error rates in manufacturing assembly and accurately guide the technician through each step. AR also allows co-workers to communicate easily using real-time data. AR improves warehousing and logistics for control and pick-and-place tasks by effectively guiding workers and avoiding mistakes. For training and supervision, AR enables vastly more effective learning outcomes for workers who need to understand complex equipment or high-risk environments. Supervisors are also able to mentor and assess a worker's capability, resulting in higher quality work with fewer mistakes. Finally, AR is able to monitor workplaces in real-time and significantly reduce the near-miss incidents. Figure 1.1 shows a use case of AR for repair and maintenance of a data center.

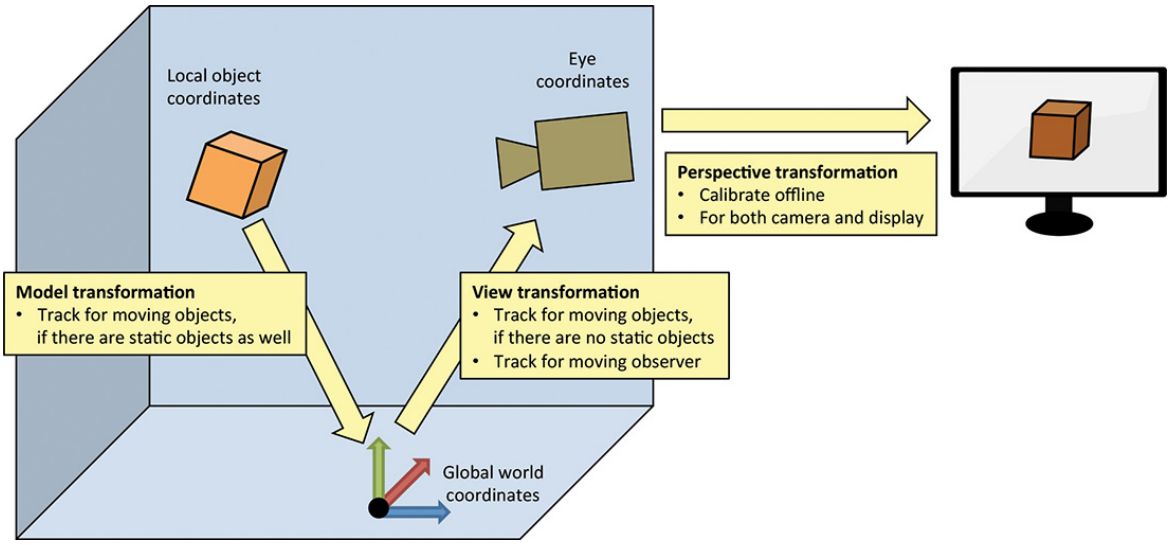


Fig. 1.2 Multiple transformation system of AR pipeline, extracted from [152]

All VR/AR/MR systems require a localization and tracking component. AR uses sensors and algorithms to determine the camera pose and the pose of objects in the environment, with high accuracy and in real-time. VR/MR systems need to colocalize the viewpoint on the virtual contents with the pose of the Head-Mounted Display (HMD). For robotics and autonomous systems, pose estimation is a key feature for automatically navigating in a 3D space. Given the significance of camera/object pose estimation in the cutting-edge fields mentioned above, improvement of their performance becomes more essential.

First of all, we need to distinguish between two concepts: camera pose estimation and object pose estimation. 6 Degree of Freedom (6-DoF) Camera pose estimation is the problem of determining the position and orientation of the camera relatively to a world coordinate system (also called global coordinate system). Object pose estimation, on the other hand, consists of determining the position and orientation of an object in the coordinate system of the camera.

Figure 1.2 shows the AR pipeline using camera/object pose estimation. AR needs to consider multiple transformation systems. The model transformation describes the pose of moving objects in a static environment. The view transformation describes the pose of a camera in an environment. The perspective transformation describes the mapping from eye/camera coordinates to screen coordinates. The moving objects in this pipeline can be real objects or virtual objects. For virtual objects, they are controlled by the application and do not require tracking. Thus, camera pose estimation is a very critical part of augmented reality.

Most of the existing solutions to camera pose estimation problem are related to employing multiple sensors because navigation, such as in outdoor environments, may involve complicated scenarios that are hardly handled by just one sensor modality. For multiple sensors navigational approach, previous works in pose estimation use data from popular sensors such as camera, GPS, LIDAR and IMU [191, 113, 188, 85, 16]. Not surprisingly, many scientific work and commercial products using them have obtained considerable positive results e.g. Tango, Hololens, ARKit, ARCore.

Concerning camera pose estimation from vision-based modality, with the recent rapid development of computer vision and machine learning coupled with the increasing computing capacity on mobile or embedded terminals, numerous camera pose estimation methods based on images have been developed and applied to augmented reality, robotics and autonomous vehicles. These methods have obtained many promising results. However, achieving high accuracy in any environment with fast computational time is still a challenge. Besides, camera pose estimation based on images still faces multiple difficulties such as dynamic scenes, large scale, fast camera motion, occlusion and variable lighting conditions.

1.2 Visual Camera relocalization

The two most common solutions of camera pose estimation for commercial systems are Structure from Motion (SfM) [161, 162, 186, 75, 122, 184] and Simultaneously Localization And Mapping (SLAM): direct SLAM [130, 39, 170], indirect SLAM [35, 86, 124].

SfM methods process offline an unordered set of images with no temporal constraint to estimate camera pose. They uses matching features amongst pairs of images. Thanks to these matching, they can both reconstruct a 3D scene model and estimate camera pose. However, these methods are not suitable for real-time systems.

Conversely, SLAM methods can work in real-time on an ordered sequence of images acquired from a set-up of one or more cameras, potentially associated with an inertial measurement unit. The majority of SLAM systems are based on camera motion tracking, using consecutive frames to robustly calculate camera pose. Unfortunately, in the case of fast camera motion or sudden change of viewpoint such as in a hand-held camera, tracking failure interrupted camera pose estimation. In addition, the tracking which relies on the reconstructed map fails every time the camera points outside the restricted maps. This can frequently happen in AR scenarios in which the user is in full control of a hand-held or head-mounted camera. For 3D reconstruction task, in order to acquire an accurate map of the scene, these reconstruction pipelines rely on successfully tracked frames. Tracking failure can have severe consequences. Integrating measurements with incorrect poses results in implausible, invalid geometry and might destroy already reconstructed areas in the map. Such a camera relocalization module allows seamless continuation of mapping. This avoids the frustrating task of restarting an entire scan from scratch due to tracking failure. Also, this makes AR applications more robust. Augmentations can be visualized with the correct transformation immediately after successful recovery of the camera pose.

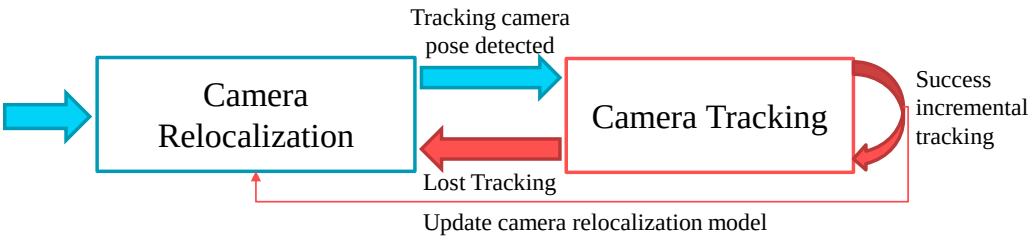


Fig. 1.3 A common localization system consists of two components: camera localization and camera relocalization.

Figure 1.3 presents a common localization system that combines camera relocalization and camera tracking to define camera pose. Both camera relocalization and camera tracking are problematics that consist of estimating camera pose. While camera tracking is an

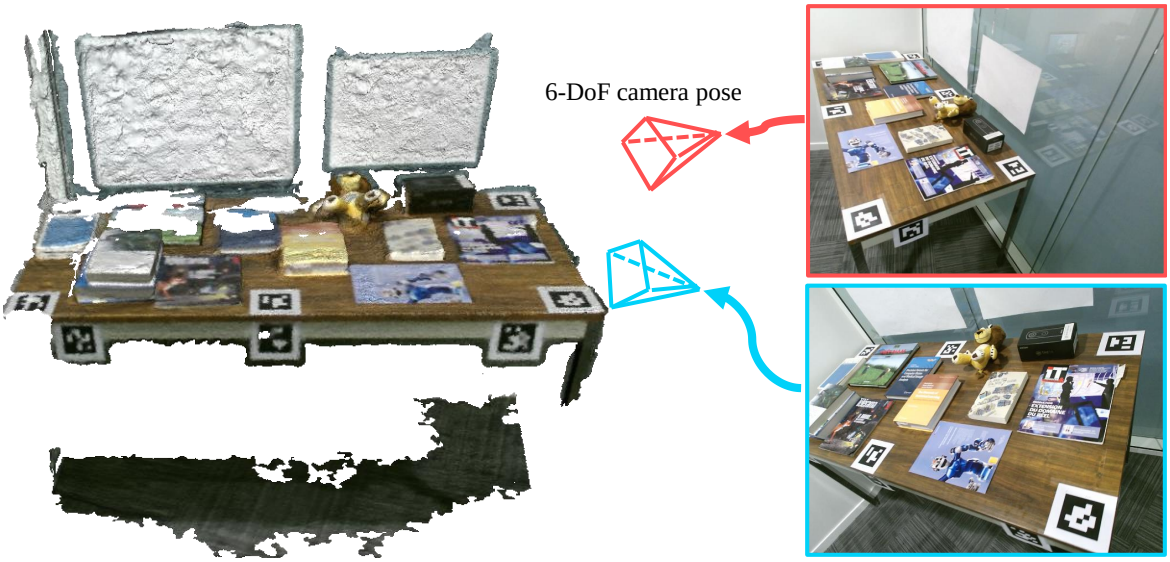
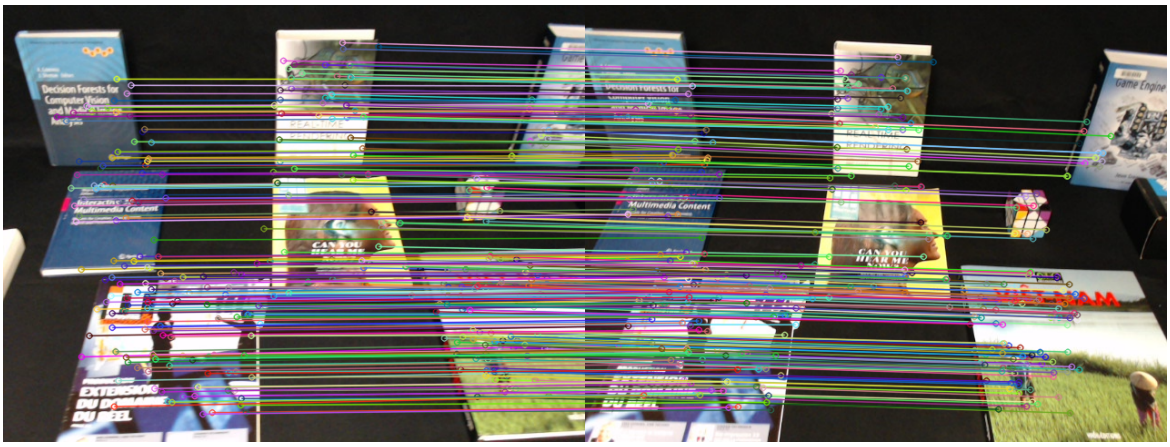


Fig. 1.4 Camera relocalization methods define individual frames independently based on prior knowledge of a scene.

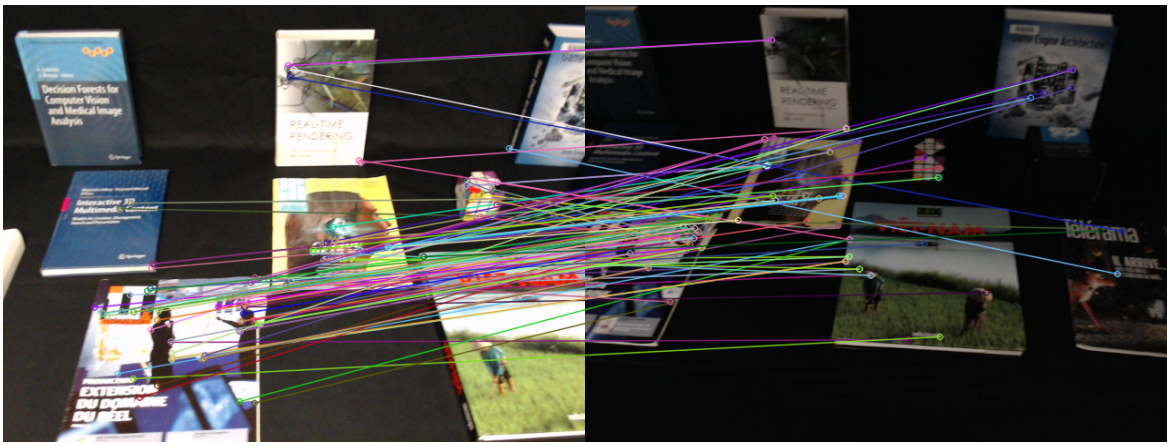
iterative process in which each current measurement is based on previous knowledge, camera relocalization is a wholy-new calculation of camera pose.

The camera relocalization is an important component in localization systems. It allows to define camera pose from individual frames independently based on models built from known information of a scene, as illustrated in Figure 1.4.

1.3 Challenges



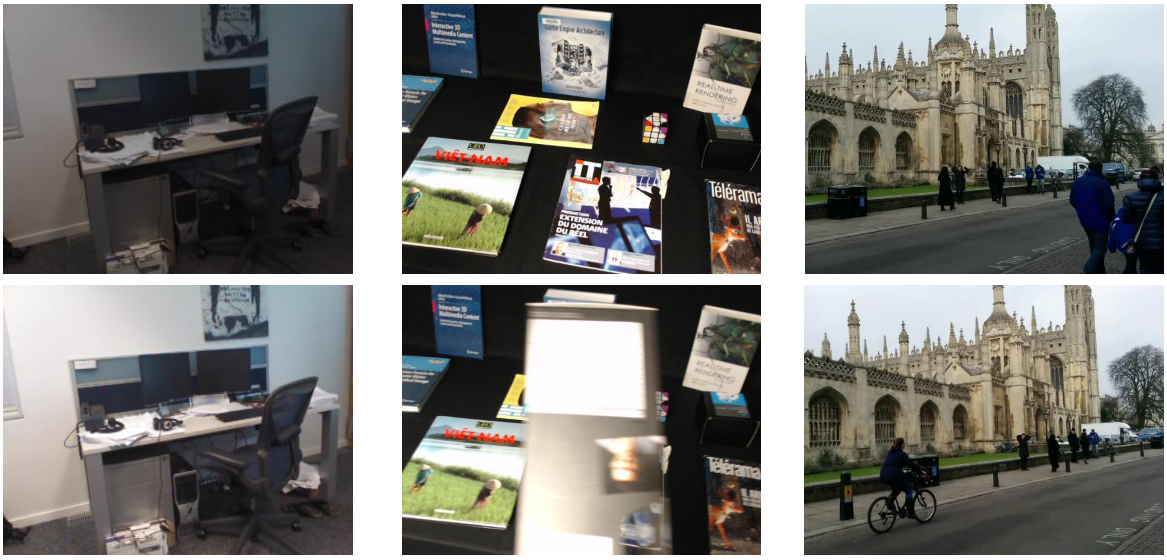
(a) Feature matching of consecutive images in a camera localization



(b) Feature matching of a current image and a nearest retrieval image in a camera relocalization

Fig. 1.5 Feature matching approaches in the camera pose estimation process.

We consider the difficulty of the camera relocalization versus the camera tracking. Figure 1.5 illustrates the feature matching in the camera pose estimation process for the camera tracking and the camera relocalization. The camera tracking uses corresponding features of consecutive frames. This allows the relative transformation of camera poses to be rapidly determined, but leads to error accumulation. Because there are few changes between successive frames, a lot of correspondences are detected. On the other hand, the camera relocalization is based on the prior knowledge of a scene that can be a set of observed images. However, these observations are achieved at different times. If there are major changes in viewpoints or environmental conditions, the feature matching between a current frame and a prior frame can fail, as shown in Figure 1.5-b). Thus, the camera relocalization task is extremely challenging



a) b) c)

Fig. 1.6 Dynamic environment challenges for the camera relocalization. a) illumination change. b) occlusion. c) moving objects.

in the computer vision community. A great camera relocalization system has to meet the following requirements:

- **Accuracy**
- **Computational time**
- **Robustness to dynamic environments**
- **Scalability in large-scale scenes**

Accuracy is the most important criterion for the camera relocalization methods. When camera tracking fails, the localization system requires a camera relocalization system. This system is able to define an accurate camera pose relatively to a 3D representation previously reconstructed of the real environment instead of restarting from scratch. Indeed, an accurate camera relocalization will allow 3D reconstruction to be extended in an homogeneous way without corrupting it. The accuracy is necessary to preserve the location of augmentations (Virtual objects) that have been setup by users in AR applications. Accuracy and stability of augmentations registration with real environment can be a key factor in the adoption of augmented reality technologies, especially when intended use case cannot accept any ambiguity of localization at the risk of severe consequences (e.g. medical surgery, autonomous vehicles, aeronautics or aerospace).

Computational time of the camera relocalization consists of the training time and processing time. Concerning processing time of camera relocalization, the real-time systems such as AR applications need immediate relocalization of the camera after tracking loss. Otherwise, this leads to an unpleasant seamless of the user's experience. In terms of training time, camera relocalization takes time to build a model from the prior knowledge e.g. a set of images captured from a scene. It can be implemented offline. However, this model is specific to each scene. Thus, the speed up model building makes the localization systems more scalable for different scenes.

Robustness to dynamic environments is a common challenge in the computer vision community, camera relocalization is not an exception. All the localization systems are still restricted by challenging dynamic scenes such as in factories and outdoor scenes. Figure 1.6 shows some camera relocalization challenges in dynamic environments. The environment can be considered as dynamic according to various conditions such as illumination changes, occlusions, moving objects. Because the camera relocalization model is built on a rigid scene, it cannot perform if the scene encounters a large change.

Scalability in large-scale scenes allows the localization systems to keep the accuracy when the scene is expanded. The camera relocalization methods need to store a large set of known information. Consequently, memory usage as well as processing time increase with respect to the size of the models.

1.4 Contributions

This thesis focuses on investigating solutions in order to tackle camera relocalization challenges mentioned above. The contributions of this thesis can be summarized as follows:

- **Accurate and real-time 2D-3D point correspondences regression**. The main contribution of this research is to address limitations of state-of-the-art methods in terms of both accuracy and computational time (including both processing time and training time) in order to achieve a real-time accurate camera relocalization method from a single RGB image. Our methods are based on the hybrid method pipeline that can be summarized in three principal steps: feature extraction; estimation of 2D-3D point correspondences from a regression learned model; camera pose estimation from these correspondences. We improve both computational time and accuracy of 2D-3D point correspondences by:
  - **Deep learning based methods**. We use the MIMO strategy (Multiple-Input and Multiple-Output) to overcome the uncertainty of deep learning models. We present a light regression convolutional neural network for camera relocalization from local patches based on key-point detection. Our network performs efficiently and robustly to predict correspondences between image pixels represented by RGB patches with fixed size and their 3D positions in world coordinates. We called this network **xyzNet**. To accelerate processing as well as improve accuracy, we only perform on patches defined thanks to a key-point detection in order to process discriminant information and generate a set of probabilistic predictions.
  - **Random forest based methods**. We propose a new split function at each node of a regression forest, which takes whole relevant information of feature vectors as inputs. This split function enables to increase the accuracy of camera relocalization. Another originality of our method concerns the feature extraction algorithm itself: we use a learned feature extraction from our convolutional neural network **xyzNet**. Our learned features are more discriminate compared to previous hybrid methods, that use the intensity of pixels or non-learned feature. Our regression forest allows us to make an online learning during capturing images. Furthermore, the regression forest predicts multiple 3D world coordinates output corresponding to 2D image feature. This allows to deal with ambiguous of scenes containing repetitive structure.
- **Adaptive regression forest for dynamic scenes**. Another important contribution of this thesis is to propose an **Adaptive Regression Forest - ARF** that has the capacity

to update rapidly when environment changes (even for dynamic environment including moving objects) in order to maintain the accuracy of the learned model. We apply our adaptive regression forest in a real-time camera relocalization method from only RGB images in dynamic scenes. Our ARF is performed by two main originalities based on detection and update of passive leaves.

  - Passive leaves detection. The first originality of our ARF is to detect the leaves which give non relevant information. There are two criteria for a leaf to become a passive leaf: having a high variance of predictive model; giving repeatedly a result rather different from the other leaves
  - Passive leaves update. The second originality of our ARF is to update in real time passive leaves of the regression forest model. This is performed by re-modeling their predictive model from new computed labels. These labels are computed based on the results given by the other leaves (active leaves). Note that the update is only performed on passive leaves and not on the whole regression forest model. We demonstrate the efficiency of this mechanism on some applicative examples in camera relocalization. The regression results remain constant as long as the changes do not involve too much data at a time (no accumulation of error).

- **Camera relocalization datasets**. A last contribution concerns the creation of datasets to evaluate camera relocalization methods. We make two datasets from indoor scenes. The first one consists of absolutely different camera trajectories to evaluate generalization capability. The second one is the first public database including indoor dynamic scenes, which is one of the most important challenges for (re)localization methods.

1.5 Thesis Outline

The following chapters of this thesis are organized as follows.

Chapter 2 firstly gives a survey of existing camera relocalization methods. We classify methods according to different approaches based on their mathematical models. We provide the principle algorithms of the approaches prior to analyze their advantages and their limitations. We then present the public datasets for the camera relocalization. We also propose two datasets of indoor scenes. Each one has a different challenge and aims at evaluating the strength of the camera relocalization methods in terms of accuracy, computational time, generalization and robustness to dynamic scenes.

Chapter 3 begins with some proposed improvements for end-to-end machine learning approaches. Then, we present our hybrid method that balances both accuracy and computational time of the camera relocalization. Our hybrid method is based on both machine learning approach and geometric approach and aims at benefiting from each.

Chapter 4 deals with the challenge of dynamic scenes. We propose a completely new machine learning algorithm based on regression forest process, that adapts itself in real time to predictive model. It evolves by part over time without having to re-train the whole model from scratch. We first describe mathematically our Adaptive Regression Forest - ARF with several concepts used in ARF model. We then apply it to our real-time camera relocalization approach from only RGB images in dynamic environments.

Chapter 5 concludes with a summary of this thesis and suggests directions for future research.

Chapter 2

Camera Relocalization - state-of-the-art

Contents

2.1 Introduction

Camera relocalization is an important module in localization system. It allows an instant recovery of the camera pose in case of initialization or tracking failures. Camera relocalization leverages models built from known information of a scene in order to infer camera pose from each image independently. In this chapter, based on the mechanism for modelling prior information, we first present the state-of-the-art of camera relocalization methods according to four different approaches as shown in Figure 2.1. We then introduce databases before concluding the chapter.

This chapter is organized as follows:

- **Section 2.2** starts with a theory of camera pose estimation based on geometric information. Then, it presents geometric approach for camera localization. This approach builds a 3D scene model, which is a 3D point cloud attached to 2D features.
- **Section 2.3** begins with a theory of machine learning, especially regression learning which is applied to pose estimation. We then present end-to-end regression learning methods which infer camera pose based on a learned model.
- **Section 2.4** presents image retrieval approach which constructs an image retrieval model. Given a query image, it first finds nearest images and provides a coarse pose. The final camera pose is achieved through either relative pose or absolute pose.
- **Section 2.5** introduces hybrid approach. It combines both the machine learning approach and the geometric approach to estimate camera pose. The machine learning approach is applied to learn a 3D world coordinate model which predicts 3D position of each pixel in the world coordinate system. Camera pose is then estimated by using geometric algorithms.
- **Section 2.6** introduces public camera relocalization datasets and metrics used to evaluate the accuracy of camera relocalization methods.
- **Section 2.7** concludes with the advantages as well as the limitations of the state-of-the-art methods.

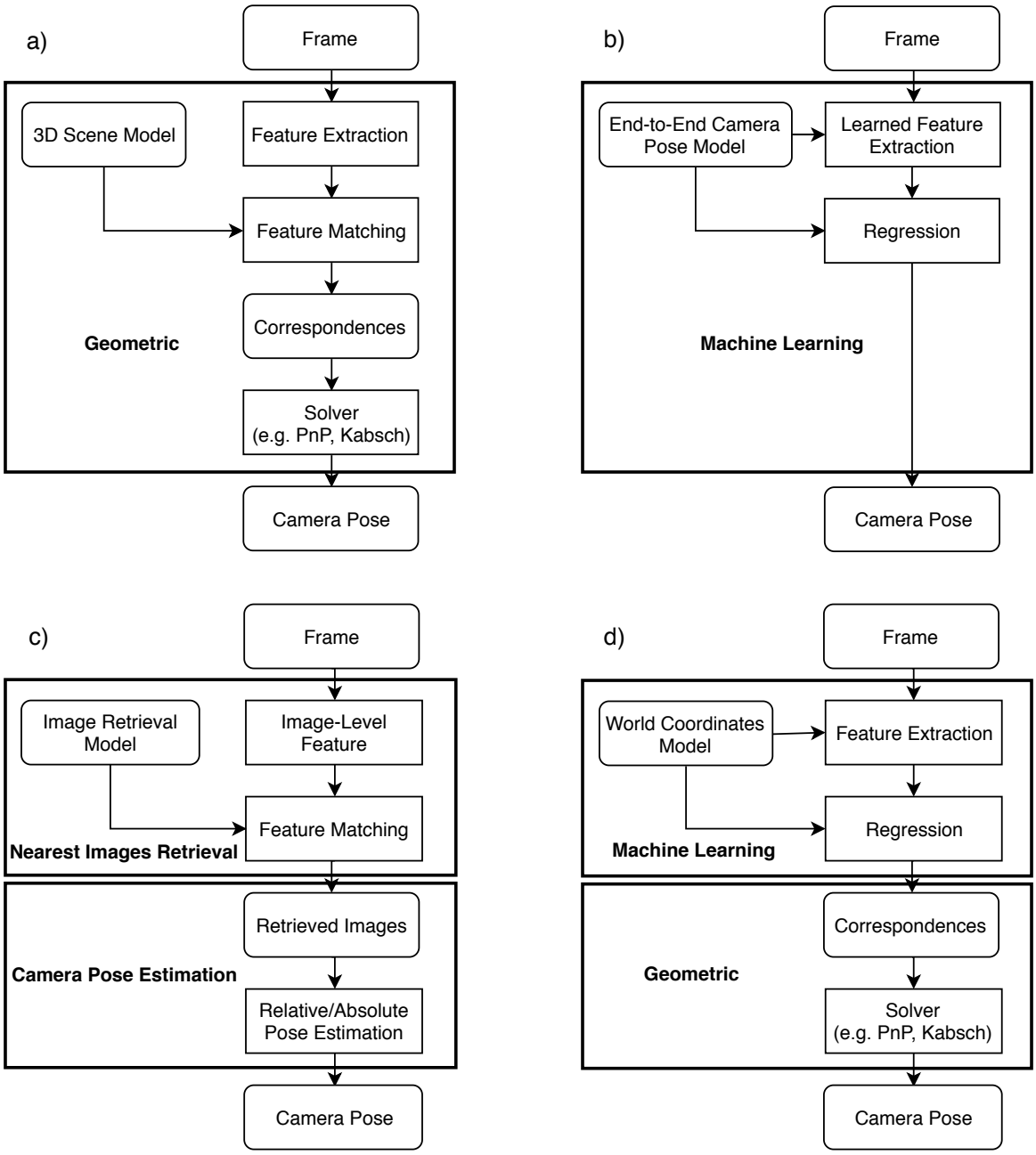


Fig. 2.1 The pipelines of the state-of-the-art camera relocalization methods: a) Geometric approach; b) Machine learning approach; c) Image retrieval approach; d) Hybrid approach.

2.2 Geometric approach

The geometric approaches for camera relocalization are methods based on point correspondences between a query image and a 3D point cloud. This point cloud is established from prior observations of the scene. The common pipeline for geometric based methods consists of three principal steps as shown in Figure 2.1-a). They begin with a keypoint detection and feature extraction. Then, the features are matched to 3D scene model, where each 3D point is attached to a feature vector. Finally, camera pose is estimated by running a solver based on point correspondences. In this section, we first present theoretical camera pose estimation based on geometric correspondences. Then, we introduce state-of-the-art geometric based methods.

2.2.1 Theory of camera pose estimation

Based on [61, 98], we present mathematical camera model and camera pose estimation methods from geometric correspondences such as 2D-3D point correspondences, 3D-3D point correspondences, 2D-3D line correspondences.

Camera representation

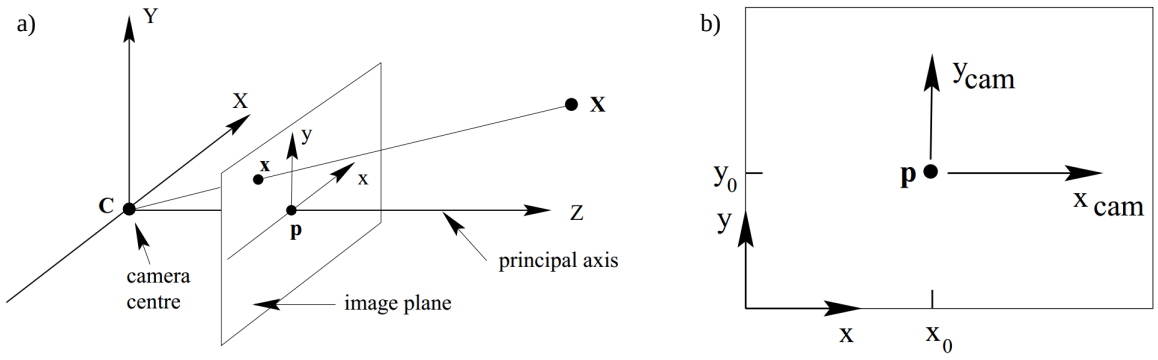


Fig. 2.2 Pinhole camera model. a) In the camera coordinate system, *C* is the camera centre and *p* the principal point which is the projection of the camera centre on the image plane. The camera centre is here placed at the coordinate origin. b) Principal point offset in image coordinate, extracted from [61]

A camera model allows to project the 3D world coordinate system onto a 2D image coordinate system. This paragraph develops several camera models based on matrices with particular properties that represent the camera projection. We focus on the standard pinhole camera model. We consider the central projection of points in 3D space onto the image

plane. Let the centre of the projection be the origin of an Euclidean coordinate system, and consider the plane $Z = f$, which is called the image plane or focal plane. Figure 2.2-a) shows the pinhole camera model. A 3D point in the world coordinate system $\mathbf{X} = (X,Y,Z)^T$ is projected onto the image plane where a line joining the point $\mathbf{X}$ to the centre of projection meets the image plane.

Camera calibration matrix. Based on the intercept theorem, we can quickly computes that the point $\mathbf{X}$ is projected onto the point $\mathbf{x} = (fX/Z, fY/Z)^T$ on the image plane with the assumption that the origin of coordinates in the image plane coordinate system is at the principal point. In practice, this may not be the case, as illustrated in 2.2-b), so that in general, the projection is defined as follows:

$$\mathbf{X} = (X,Y,Z)^T \mapsto \mathbf{x} = (fX/Z + p_x, fY/Z + p_y)^T \tag{2.1}$$

This equation may be expressed conveniently with the matrix equation as follows:

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{2.2}$$

Denote:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

The Equation 2.2 has the concise form:

$$\mathbf{x} = K\mathbf{X}_{cam} \tag{2.4}$$

Where $(X,Y,Z)^T$ is written as $\mathbf{X}_{cam}$ to emphasize that the camera is assumed to be located at the origin of an Euclidean coordinate system and the point $\mathbf{X}_{cam}$ is expressed in this coordinate system, called the **camera coordinate system**. The matrix K is called the **camera calibration matrix**. It contains the **intrinsic camera parameters**, also called internal parameters. In the case of CCD cameras, there is the additional possibility of having non-square pixels. If image coordinates are measured in pixels, then this has the extra effect of introducing unequal scale factors in each direction. Thus, the general form of the

calibration matrix is:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represent the focal length of the camera defined in pixels along the x and y axis respectively. m_x and m_y are the number of pixels along the x and y axis. $\mathbf{x}_0 = [x_0, y_0]^T$ represents the image coordinates of the intersection of the optical axis with the image plane, also called the principal point. s is referred as the skew. It is non-zero only if the x and y directions are not perpendicular, and is generally considered negligible on modern cameras. These parameters can be estimated during an offline camera calibration stage, from the images themselves. Some popular calibration methods [163, 192] rely on a simple planar pattern seen from several positions. The OpenCV library [24] also provides a camera calibration tool.

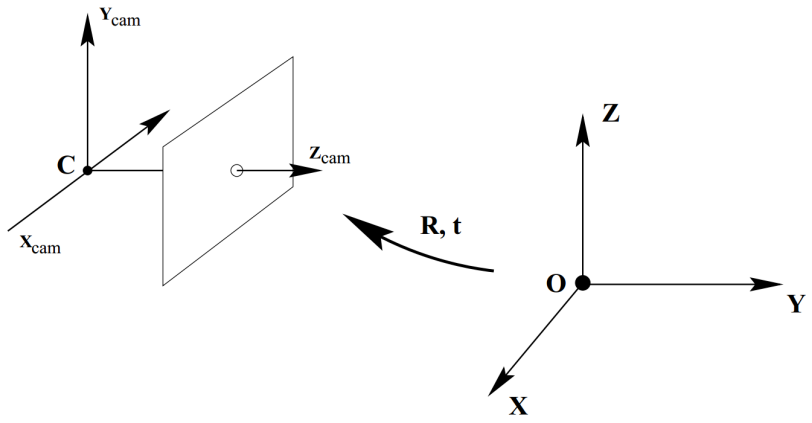


Fig. 2.3 The Euclidean transformation between the world and camera coordinate systems, extracted from [61]

Camera pose. In general, points in 3D space will be expressed in the **world coordinate system**. The camera coordinate system is defined in the world coordinate system by an affine 3D transformation consisting of a 3D translation and a 3D rotation, as shown in Figure 2.3. $\mathbf{X}$ is a 3D point in the world coordinate system, and $\mathbf{X}_{cam}$ represents the same point in the camera coordinate system. Thus, we have a relative equation $\mathbf{X}_{cam} = R(\mathbf{X} - C)$, where C represents the coordinates of the camera center in the world coordinate system. R is a 3×3 rotation matrix representing the orientation of the camera coordinate system relatively to the world coordinate system. This equation may be written in homogeneous coordinates as

follows:

$$\mathbf{X}_{cam} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.6}$$

Denote $t = -RC$, we have a matrix $H = [R \mid t]$ being the 3×4 **extrinsic parameters matrix** that defines the orientation and the position of the camera. It is formed of a R rotation matrix and a t translation vector. It corresponds to the Euclidean transformation from a world coordinate system to the camera coordinate system. Thus, we refer to the **camera pose** T as the inverse of external parameters matrix H:

$$T = H^{-1} = [R^{-1} \mid -R^{-1}t] = [R^T \mid -R^T t] \tag{2.7}$$

The camera localization systems usually assume that the calibration matrix K is known in advance and focus on estimating the camera pose T.

Camera pose estimation

By replacing the $\mathbf{X}_{cam}$ in Equation 2.4 by Equation 2.6, we can define the projection matrix from each 3D point $\mathbf{X}$ defined in the 3D world coordinate system to the point defined in the 2D image coordinate system:

$$\mathbf{x} = K[R \mid t]\mathbf{X} = P\mathbf{X} \tag{2.8}$$

Denote $P = K[R \mid t]$, it is the 3×4 **perspective projection matrix** that is defined up to a scale factor, and thus depends on 11 parameters. The 11-DoF of the projection matrix P can be decomposed into the 5-DoF for K, 6-DoF of camera pose including 3-DoF for R, and 3-DoF for t. This equation allows to estimate the external parameters when the internal parameters are known is often referred to as the **camera pose estimation**.

2D-3D point correspondences. We assume here that we have n correspondences between 3D points $\mathbf{X}_i$ in the world coordinate system, and their projections $\mathbf{x}_i$ in the image coordinate system. We are looking for the perspective projection matrix P of Equation 2.8 that projects the points $\mathbf{X}_i$ on $\mathbf{x}_i$. In the computer vision community [41, 61], the Direct Linear Transformation (DLT) is developed to estimate the whole matrix P by solving a linear system even when the internal parameters are not known. Each correspondence $\mathbf{X}_i$ - $\mathbf{x}_i$ gives rise to two

linearly independent equations in the entries P_{ij} of P:

$$\begin{aligned} \frac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} &= x_i \\ \frac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} &= y_i \end{aligned} \tag{2.9}$$

We can write these equations in the form $\mathbf{Ap} = 0$, where $\mathbf{p}$ is a vector made of the coefficients P_{ij}. Then the correct solution can be found from the Singular Value Decomposition (SVD) of $\mathbf{A}$. It needs at least 6 correspondences ($n \geq 6$) to define a unique excepted when the points are coplanar and there is no triplets of collinear points, the solution is unique for $n \geq 4$.

3D-3D point correspondences. For RGB-D cameras, from 2D-3D point correspondences with the depth image, we can define 3D-3D correspondences between 3D points $\mathbf{X}_i$ in the world coordinate system, and their projections $\mathbf{X}_{cam}$ in the camera coordinate system. The problem becomes to define the affine transformation between two paired sets of 3D points.

$$\begin{pmatrix} \mathbf{X}_{cam} \\ 1 \end{pmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \tag{2.10}$$

The translation t coincides with the translation of their centroids. The rotation is found by the Kabsch algorithm [77] or Horn algorithm [66] (it uses unit quaternions to represent rotations) that minimizes the Root Mean Squared Deviation (RMSD) between 3D-3D point correspondences.

Line correspondences. It is quite easy to extend DLT algorithm to consider line correspondences. A line in 3D may be represented by two points $\mathbf{X}_0$ and $\mathbf{X}_1$ through which the line passes. The plane formed by back-projecting a line l from the image plane is equal to $P^T l$. The condition that the point $\mathbf{X}_j$ lies on this plane is then

$$l^T P X_j = 0, j = 0, 1. \tag{2.11}$$

Each choice of j gives a single linear equation in the entries of the matrix P. So two equations are obtained for each 3D to 2D line correspondence. These equations, being linear in the entries of P, may be added to Equations 2.9 obtained from point correspondences and a solution to the composite equation set may be computed.

The Perspective-n-Point (PnP) Due to the low adoption of depth cameras in mobile devices (3D-3D point correspondences) and the difficulty in defining good line descriptors

(2D-3D line correspondences), the 2D-3D point correspondences methods are the most popular. The DLT method aims at estimating all 11 parameters of the projection matrix. But it relies on an over-parameterization if the internal parameters are known. This results in instability requiring more point correspondences than is really necessary. The problem of recovering the 6-DoF pose of a calibrated camera from n 2D–3D point correspondences is known as the Perspective-n-Point (PnP) problem. A minimum of three 2D–3D correspondences is necessary for six constraints matching the 6-DoF. With only three points, this P3P (PnP with $n = 3$) algorithm delivers up to four ambiguous solutions. A fourth point correspondence is required to determine the unique solution. Different approaches to the P3P problem have been proposed within the Computer Vision community [60, 138, 49]. They usually attempt to first estimate the distances $\mathbf{x}_i = \|C\mathbf{X}_i\|$ between the camera center C and the 3D points $\mathbf{X}_i$, from constraints given by the triangles $C\mathbf{X}_i\mathbf{X}_j$. Once the $\mathbf{x}_i$ are known, the $\mathbf{X}_i$ are expressed in the camera system as $\mathbf{X}_i^c$. Then, $[R \mid t]$ is chosen to be the displacement that aligns the points $\mathbf{X}_i$ on the $\mathbf{X}_i^c$ and can be found in closed-form solution using quaternions [66] or singular value decomposition (SVD) [67]. Solving the $\mathbf{x}_i$ requires finding the roots of a fourth degree polynomial. To remove the ambiguity, one additional correspondence is needed. A simple approach is to solve the P3P problem for subsets of three of the four points, and retain the common solution. POSIT [36] is another popular way to solve the pose estimation problem for $n \geq 4$. [99] proposes a non-iterative solution to the PnP whose computational complexity grows linearly with n.

Then, camera pose should be refined by minimizing the sum of the reprojection errors. When a first estimation of the camera pose is known, we minimize the squared distance between the projection of the 3D points $\mathbf{X}_i$ and their known 2D image coordinates $\mathbf{x}_i$:

$$T^* = \arg\min_T \sum_i \|KT^{-1}\mathbf{X}_i - \mathbf{x}_i\|_2^2 \tag{2.12}$$

This kind of quadratic minimization problem is usually solved iteratively [18].

RANSAC PnP problem assumes that correspondences are correct, what is rarely the case and can result in a wrong convergence on minimization. To filter the good correspondences of the bad ones and thus minimize the reprojection error in order to obtain an optimal installation, the random sampling consensus is generally used (RANSAC) [43]. The main idea of RANSAC is to estimate the model parameters from a randomly chosen subset of the data points. For camera poses, P3P is often used, since it requires only three 2D–3D point correspondences. For each of the remaining potentially point correspondences, we compute the residual error, assuming the camera pose is computed from the three chosen

points. A data point with a residual smaller than a threshold counts as an inlier. If the ratio of inliers to outliers is not sufficient, the procedure is repeated. RANSAC terminates either if enough inliers are found or if a maximum number of iterations is reached. Finally, the model parameters are re-estimated using all inliers from the iteration with the largest number of inliers.

2.2.2 Point correspondences matching based methods

The geometric approach for camera relocalization is based on a pre-computed 3D point cloud. Given a 3D model of the scene, the camera pose can be estimated by directly matching 2D image features from the query image to 3D points in the map to define 2D-3D point correspondences for RGB images or 3D-3D point correspondences for RGB-D images. Then, it solves a standard camera absolute pose problem via PnP (2D-3D) or Kabsch (3D-3D) algorithms. If the matches found are contaminated by some small portion of wrong matches (outliers), RANSAC is conventionally applied to clean up the matches as well as accelerate the camera pose computation. See 2.2.1 for further details of PnP, Kabsch and RANSAC algorithms.

In the following paragraphs, we present the two main steps of the geometric approach pipeline:

- Build a 3D model of the scene. This model contains a set of 3D points in the world coordinate system associated with feature vectors.
- Match 2D image features of a query image to the 3D model to define 2D-3D point correspondences.

3D model construction

A 3D point cloud model is built from a set of images captured by one or more cameras observing a scene. The 3D model can be a sparse or dense point cloud.

A dense point cloud is generally build from direct SLAM methods [131, 107, 130, 181, 129, 39, 38, 182, 183, 170]. By using a depth sensor, a dense point cloud is created from depth maps. Camera pose is estimated by tracking points cloud using Iterative Closest Point (ICP) algorithm [130] and Lucas-Kanade algorithm [134]. However, for the camera relocalization, due to the restriction of 3D feature matching, a query image cannot match its features to the dense model. Thus, we utilize a sparse 3D point cloud associated with 2D features from RGB images, as shown in Figure 2.4. This sparse 3D point cloud is constructed thanks to offline methods or online methods.

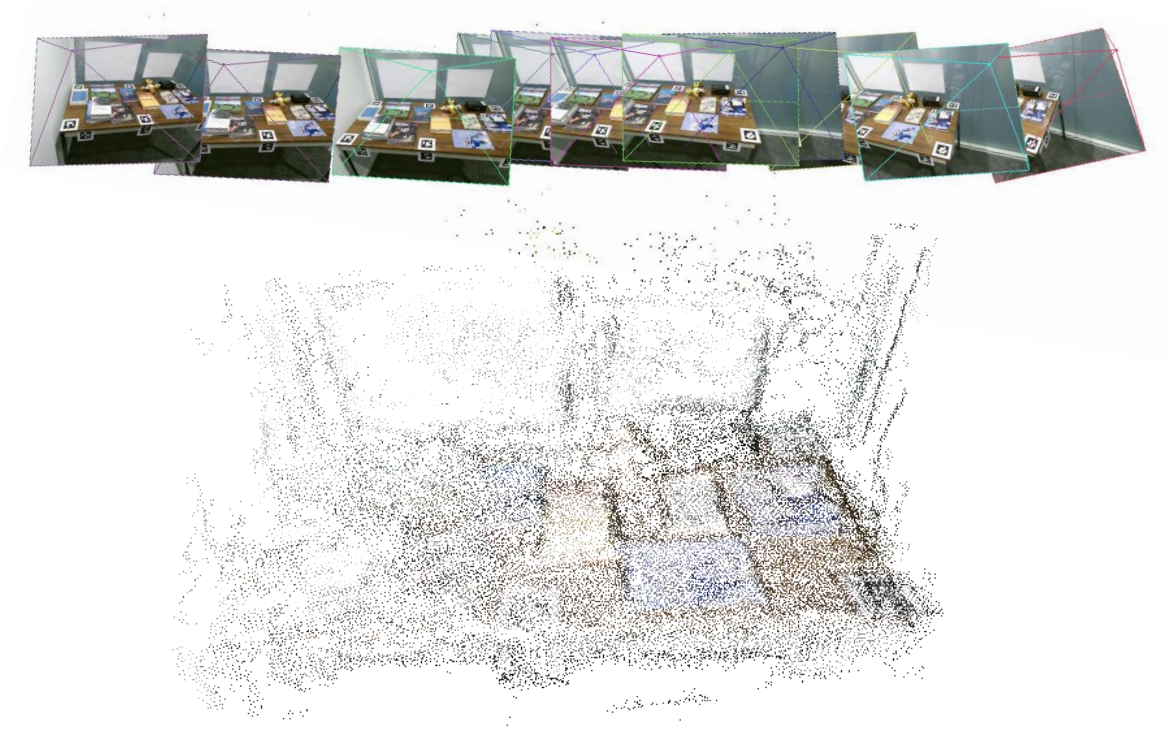


Fig. 2.4 A 3D points model attached to feature vectors is constructed by SfM from a set of images.

The offline approach is known as Structure from Motion (SfM) [161, 162, 186, 75, 122, 184, 1, 136, 61]. It allows a complete sequence of images to be analyzed in order to perform a 3D map reconstruction and a camera trajectory estimation. Firstly, feature extraction (e.g. SIFT [111], SURF [13], ORB [144], AKAZE [3]) is performed on images. Then, camera pose is estimated by using features matching amongst pairs of images in the image connectivity graph. Next, 3D points in the map are computed based on estimated camera pose and corresponding points in pairs of images by using triangulation algorithm [62]. As illustrated in Figure 2.5, from each pair of matching keypoints (p_1, p_2) of two RGB images for which the pose (T_1, T_2) is known, a 3D point P is defined by:

$$\begin{cases} p_1 \times (KT_1^{-1}P) = 0 \\ p_2 \times (KT_2^{-1}P) = 0 \end{cases} \tag{2.13}$$

And each 3D point is represented by a 2D descriptor. It can be the mean of descriptors obtained from different observations. Finally, an optimization called bundle adjustment [173] is performed to minimize the re-projection error of points.

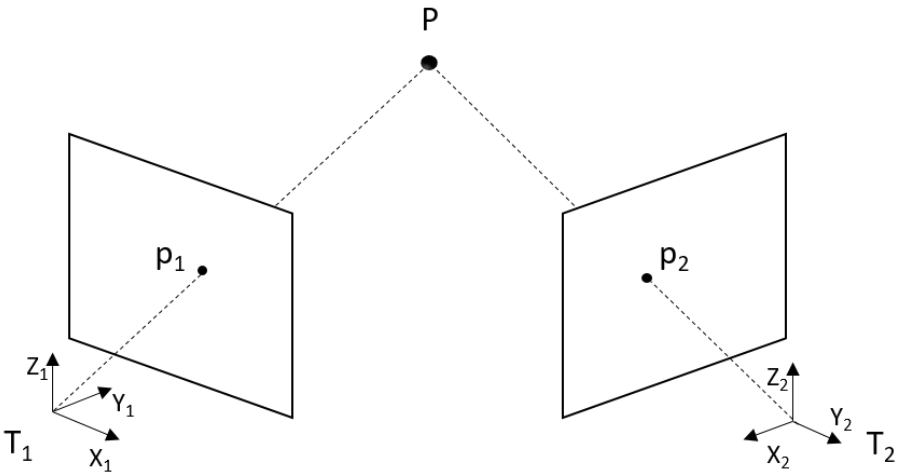


Fig. 2.5 3D point triangulation from a pair of matching points in two images with their estimated camera poses.

SfM includes two approaches namely incremental SfM and global SfM. Incremental SfM methods [161, 186, 2] are the standard approach that adds one image at a time to grow the reconstruction. While this method is robust, it is not scalable because it requires to repeat expensive operations such as bundle adjustment. On the contrary, global SfM methods [75, 122, 184] consider the entire view graph altogether, instead of incrementally adding more and more images to the reconstruction. For this reason, global SfM methods prove themself to be much faster, with the same or even better accuracy than the incremental SfM approach. It is also much more readily parallelized. The major limitation of SfM relies on the fact that it only works on an already collected sequence of images.

The most popular online approach is known as indirect SLAM [35, 86, 124] (Simultaneous Localization And Mapping). The majority of visual SLAM systems are based on camera motion tracking from frame-to-frame through consecutive frames. A tracking system passes through three steps: initialization, prediction, correction. First a few initial landmarks have to be given by using a known object [35] (as a A4 sheet of paper or fiducial marker) or by stereo algorithm [86, 124]. Then the camera pose is predicted and corrected by using Kalman Filter - KF [35, 86, 124] or a particle filter [137, 92]. Next, the 3D point cloud is updated based on a triangulation algorithm knowing the pose of the cameras.

Finally, SLAM methods use local bundle adjustment on selected keyframes [86] or fast global optimization to refine and avoid map duplication (e.g. pose graph) by loop closure as in [124]. However, in large scenes without loop closure, frame-to-frame tracking accumulates error causing drifts. This leads to a 3D point cloud that is not built correctly.

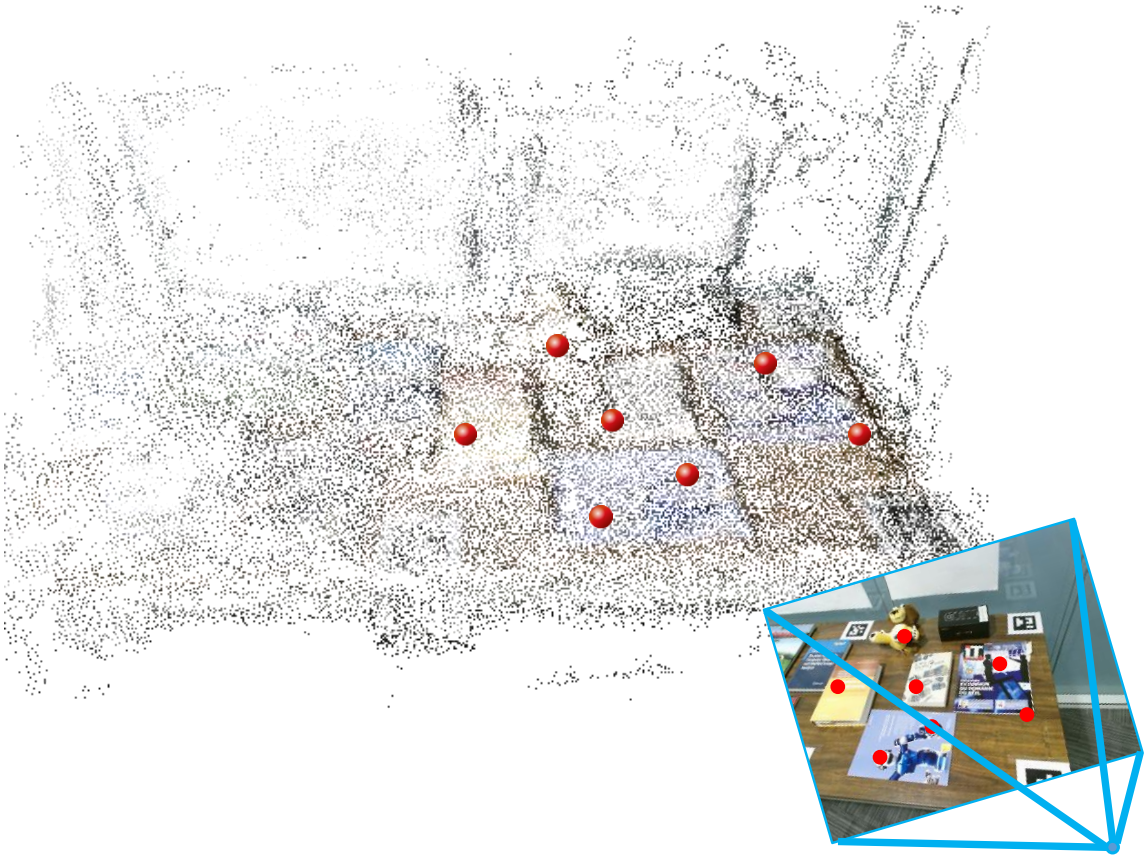


Fig. 2.6 Camera relocalization from directly defining 2D-3D point correspondences.

Direct 2D-3D point correspondences matching

Direct 2D-3D point correspondences matching consists in finding 3D points in the world coordinate system corresponding to each 2D feature of a set of keypoints detected in the image coordinate system. This is performed by searching for the nearest neighbors of the keypoint descriptors in the space containing all the descriptors of the 3D point cloud. Figure 2.6 shows 2D-3D point correspondences (red points) defined by this approach. When the 3D map is very large, for example, covering a wide geographical area of a city, there may have millions of 3D points, which raises two major challenges to this approach:

- How to quickly search within a massive database of a very large scene containing millions of 3D points
- How to accurately find correct 2D-3D matches without suffering from ambiguity.

Classical direct matching approaches use the approximate nearest neighbor search. The most widely used algorithm for the nearest neighbor search is the kd-tree [44] which works

well for exacting nearest neighbor in low dimensional data. [14, 8] modify the original kd-tree algorithm to use it for an approximate strategy with high dimension. [158] proposes the use of multiple randomized kd-trees as a means to speed up the approximate nearest neighbor search. On the other hand, [109, 132] use k-means algorithm to compute k-nearest neighbors. [123] proposes FLANN (Fast Library for Approximate Nearest Neighbors) library that contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. However, these searching methods become prohibitively expensive in very large and dense feature collections.

Between methods based on prior information, [7] presents a system that recovers the pose of a camera by confronting an image to a subset of 3D points that should be visible in the query. The subset of 3D points is retrieved by using a rough camera pose provided from an external sensor like GPS.

Without a prior camera pose, the 2D-3D matching becomes much more difficult because of having millions of 3D points. Given a set of 2D features $\mathscr{F}$ in a query image and a set of features $\mathscr{P}$ representing the 3D points in the 3D model, two basic matching strategies are considered:

- **Feature-to-point matching - F2P**, where one takes each feature in the query image, and finds the best matching point in the 3D model.
- **Point-to-feature matching - P2F**, where one conversely matches points in the 3D model to 2D features in the query image.

Using tree-based approximate search [123], the computational complexity for matching all the features against the 3D points is $\mathscr{O}(|\mathscr{F}|\log|\mathscr{P}|)$. And the one for matching all points against the query image requires time $\mathscr{O}(|\mathscr{P}|\log|\mathscr{F}|)$. For large-scale scenes, there are orders of magnitude more points than query features. Thus, P2F matching will only be more efficient than F2P search if only a small fraction of all points is considered. To accelerate 2D-3D matching for both F2P and P2F strategies, methods attempt to reduce feature search space by using the prioritized feature search algorithm [106, 149, 151].

[106] proposes a prioritized P2F matching strategy based on co-visibility information. Starting with a set of seed points selected from all parts of the model, they match points against the query image in order to descend priorities. Once a new match is found for a point, P2F increases the priorities of all other points that are visible together with this point in at least one database image. The search is stopped when a fixed number of points has been tried. [71] first proposed the F2P method for camera localization based on the SfM scene representation. To overcome limited viewpoints in the database images, they artificially synthesized novel view image to augment the database. [149] introduces a Vocabulary-based

Prioritized Search (VPS) inspired by Bag-of-Words (BoW) matching method. The number of features stored in a visual word therefore gives a good estimate of the matching cost for this particular query feature. To speed up feature matching, they process the features in ascending order of their matching costs, starting with features whose activated visual words contain only few features. The search stops once large enough correspondences have been found. [105] show that the class of methods introduced in [71, 106] can deal with large environment. They augment the P2F matching with hypothesis of co-occurrence of 3D points present in a close neighbourhood. Based on similar spatial observation, [148, 108] consider visibility graph to reject wrong matches. [42] proposed to use binary features to speed up the search. [37] uses the feature redundancy associated to 3D points to train random ferns on the top of each points. F2P matching time requirement is by the fact greatly reduced. [166] considers F2P matching as a combinatorial optimization problem and design a fast outliers rejection scheme. This promising work have been improved through contribution of [190].

Lowe's ratio test [112] is classically used to reject ambiguous matches. After finding the two nearest neighbors f_1, f_2 in the point feature space for a descriptor f of the query image, a 2D-3D correspondence between the 2D feature f and the 3D point corresponding to f_1 is established only if the ratio test is passed:

$$\|f - f_1\|_2 < \tau \|f - f_2\|_2 \tag{2.14}$$

where τ is typically from the range $[0.6; 0.8]$. The effectiveness of the ratio test strongly differs for the two search strategies. Since F2P search performs matching on a global level, the second nearest neighbor is also contained in this set, allowing the ratio test to discard the match. On the contrary, P2F matching obviously resolves this global ambiguity since it considers each 3D point independently of the others. Thus, it is likely that the ratio test accepts matches for all 3D points in the set if one of the points passes the test, leading to a significantly higher false positive matching rate. At the same time, F2P search is more likely to reject correct matches due to such global ambiguities. For larger models, F2P matching can therefore reject too many correct matches, leading to a reducing localization effectiveness.

[150, 151] propose an **Active Search** mechanism based on both F2P and P2F search. This allows them to exploit the distinct advantages of both strategies, while avoiding their weaknesses. This method first considers features more likely to yield F2P matches and to terminate the correspondence search as soon as enough matches have been found. Matches initially lost due to the quantization are efficiently recovered by integrating P2F search with a lower computational complexity.

2.3 Machine learning approach

The problematic of camera relocalization can be solved by an end-to-end learning approach. In machine learning, camera relocalization is usually considered as a supervised regression problem. Methods based on this approach follow the same pipeline as described in Figure 2.1-b): Features are extracted by classical feature descriptors or learned features using the known network, e.g. AlexNet [89], GoogLeNet [167], VGG [159], ResNet [64], pretrained on ImageNet dataset [147]. Then, the feature is used to regress the camera pose in the scene. A regression model is learned from the data of the known scene represented as a set of labeled images: the images of the scene are captured from different viewpoints and labelled with theirs camera poses (6-DoF). Machine learning based methods are only capable of performing on known scenes and each trained model is uniquely used for the corresponding scene. In this section, we first present machine learning from theory to application and especially regression learning which is applied to pose estimation. Then we introduce state-of-the-art camera pose regression based methods.

2.3.1 Machine learning theory for pose estimation

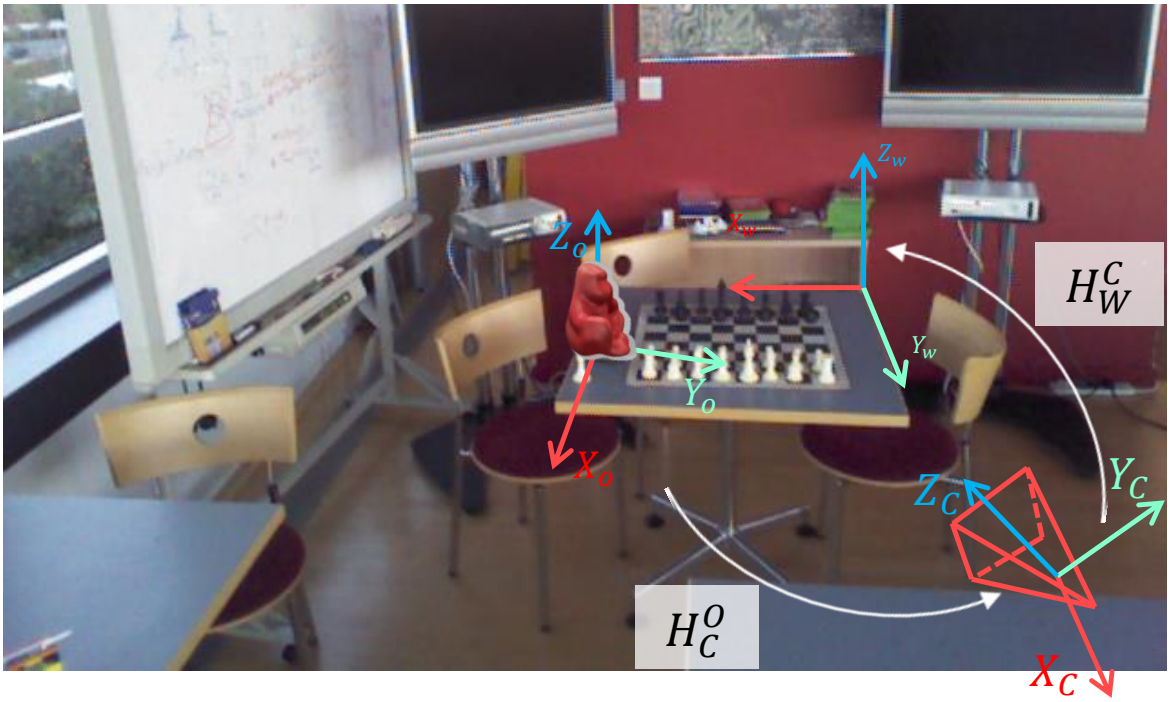


Fig. 2.7 Object pose estimation: 6-DoF object pose in the camera coordinate system, H_C^O. Camera relocalization: 6-DoF camera pose in the world coordinate system, H_W^C.

Here we present the theory of supervised random forest and deep learning that are widely used for camera relocalization. For each method, we introduce briefly its mechanism and its applications from classification to regression. Then, we summarize machine learning methods for object pose estimation and for camera relocalization, both addressing a 6-DoF problem, even if camera relocalization generally address a larger scale. Figure 2.7 gives an example to distinguish these two concepts.

Random Forest

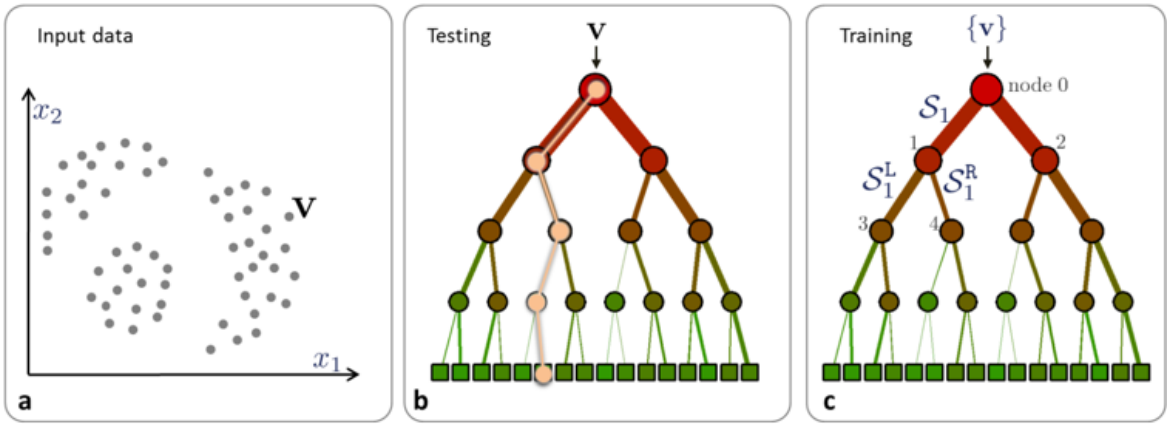


Fig. 2.8 Decision tree. a) Input data is represented as a collection of points in the d-dimensional space. b) Testing a decision tree with data v. c) Training a decision tree involves sending all training data v into the tree. Extracted from [33].

Random forests (RFs) are one of the most popular machine learning tools because of their speed, robustness and generalization [26, 33]. RFs are widely applied in computer vision tasks such as object detection [46], object pose estimation [21], eye tracking [78], human pose recognition [157], etc. A random forest is an ensemble of decision trees. A tree is a collection of nodes and edges organized in a hierarchical structure (Figure 2.8-b,c). Nodes are divided into internal (or split) nodes and terminal (or leaf) nodes. We focus only on binary trees where each internal node has exactly two outgoing edges. For each decision tree, a previously unseen data v is considered starting at the root node and descending to the leaf node by repeatedly evaluating weak learner at each split node (Figure 2.8-b).

Training a random forest. In a random forest, a fraction of data S is selected to train a decision tree (Figure 2.8-c). At each split node i, the parameters of a split function θ_i allow

to define how the set of feature S_i is split into left child node S_i^L and right child node S_i^R.

$$h(v,\theta_i) = \begin{cases} 0, & \text{go to left child node} \\ 1, & \text{go to right child node} \end{cases} \tag{2.15}$$

The training entails finding the parameters of split functions stored in the split nodes by optimizing a specified objective function I_i:

$$\theta_i^* = \arg\max_{\theta\in\Theta} I_i(S_i) \tag{2.16}$$

$$I_i = H(S_i) - \sum_{j\in\{L,R\}} \frac{|S_i^j|}{|S_i|} H(S_i^j) \tag{2.17}$$

Where Θ represents the space of all split parameters. $H(.)$ is the entropy function. The split procedure described above proceeds recursively to all the newly constructed nodes and the training phase continues until a stopping criterion is met. There are various stopping criteria such as stopping the tree when a maximum number of levels is reached or a node contains too few training features. In the leaf node, the statistics of labels are stored to predict the unknown features. The statistic model is a discrete distribution for classification tasks and is a continuous distribution for regression tasks.

Testing a random forest. In the testing phase, unseen data feature v passes through a regression forest to obtain a set of predictions: one prediction per a decision tree. Then, the forest output is usually the average of all predictions:

$$p(y|v) = \frac{1}{T}\sum_{t\in T} p_t(y|v) \tag{2.18}$$

Deep learning

Deep learning [95, 54] is now widespread in the field of computer vision. As a machine learning tool, deep neural networks are very effective at understanding high dimensional data, such as images. They learn representations by encoding the input through a number of non-linear layers and sub-sampling operations, resulting in powerful image-level understanding and recognition capabilities. Deep learning models were first used in computer vision for image recognition tasks [96, 89]. Recently, deep learning has been successfully applied to different tasks in computer vision since their major success in object detection [51, 141], semantic segmentation [110]. Figure 2.9 shows some applications of deep learning.

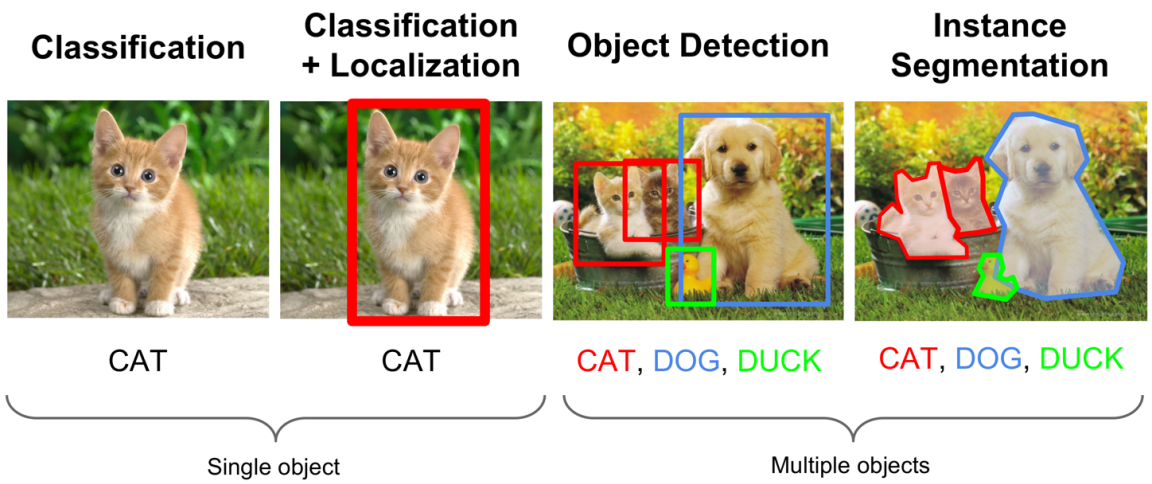


Fig. 2.9 Some deep learning applications.

Deep learning models, in simple words, are large and deep artificial neural networks. Compared to MLP (Multilayer Perceptron), deep neural network has many more layers and many more nodes in each layer, which results in exponentially many more parameters to tune. insufficient data will lead to sub-optimal learning. Without powerful computers, learning would be too slow and insufficient. Convolutional neural networks [96] are a subset of deep learning which are particularly useful for computer vision because they can share neurals to reduce processing time and they are spatially invariant.

The first CNNs have been applied to classification tasks. The architecture of a typical CNN is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank. The result of this local weighted sum is then passed through a non-linearity such as a ReLU. All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. The classification happens in the later fully-connected layers which consume the extracted features and softmax classifier layer which outputs a vector representing the probability distributions of a list of classes. These models are typically optimized by propagating a training signal from the output all the way to the input. This is known as end-to-end learning. Algorithms like stochastic gradient descent [17] and back-propagation [145] can train networks containing millions of parameters. These networks can be optimized from a loss function formulated from supervised labelled training data.

For regression tasks in computer vision, deep learning based methods span a large ensemble of applicative scenarios such as: human pose estimation [172], object pose estimation

[80], camera pose estimation [83], depth estimation [170]. Besides classification, CNN methods are also used to solve regression problems. In this case, the softmax layer is commonly replaced with a fully connected regression layer with linear or sigmoid activations. However, limitations of these methods lie in the lack of probabilistic distribution for output. Recently, [45] proposes a practical framework for understanding uncertainty with deep learning models.

Object pose regression

Object pose estimation includes two steps: object detection (known as object localization and recognition) and object pose estimation. In this paragraph, we present object pose estimation solutions based on machine learning, whether dedicated to classification or regression, from shallow to deep learning, as well as methods solving both object detection and pose estimation simultaneously.

From classification to regression Classification strategy is widely used for object pose estimation. The first application is for estimating head pose in [69, 93, 101]. Each image in the dataset is labeled with a discrete information (Euler angles: roll, pitch, yaw). Then, a classification algorithm learns the dataset discriminatively according to these classifiers, which are discretized poses, by using SVM classifier with various strategies one-vs-one [93], one-vs-all [69]; adaboost with image RGB-D in [189]; random forest classification in [68]; and a fine-grained viewpoint classification using CNNs in [164]. This approach is similar to templates-based methods. But instead of comparing an image to a large set of individual templates, the image is evaluated by a trained model. This approach modifies the classification goal to take into account the uncertainty of the annotations and encode implicitly the topology of the pose space. An advantage of these methods is that each model is also capable of making the distinction between objects and non-objects. Another improvement is that, unlike templates-based and geometry-based methods, machine learning methods can solve appearance variation as well as for high and low-resolution objects. However, head pose in particular and more generally pose estimation are measured in a continuous 6-DoF space. The classification methods have binary outputs as coarse pose estimation. Therefore, these methods cannot exactly calculate object pose. Furthermore, they face difficulties to train many discrete poses. This approach seems to be suitable only for tasks that do not require high accuracy such as determining head pose of driver (straight, left, right, top, bottom) to warn distraction.

With the development of machine learning, the regression approach has been proposed. Specially, nonlinear regression methods that estimate pose by learning a nonlinear fitting. The regression methods include support vector regressors (SVRs) [104, 127], multilayer

perceptron (MLP) [154], hough forest for regression [40], convolutional neural networks (CNNs) [185]. The first methods for object pose estimation are proposed in [104, 127, 154]. Each DoF of object pose is trained separately. The trained model of an object are a set of models. Recently, [19, 21, 90, 120, 88] propose to link conjointly object detection and pose estimation in unique model based on hough forests [47, 46, 156]. Moreover, [91] uses an extended particle filter framework to track object pose. It can also handle strong occlusion. However, these methods can only be applied on RGB-D images [156]. [88] proposes a method that only requires positive datasets for training.

From shallow to deep In this paragraph, we present machine learning approaches from shallow to deep related to learned feature (deep learning) or non-learned feature (traditional machine learning).

The classical pipeline of machine learning consists of two steps: feature extraction and learning model. In deep learning, feature extraction is learned. The classic machine learning methods for object pose estimation use the known feature: KPCA [101]; HOG [189]. However, feature extraction aims at changing image space to more discriminate feature space. In particular, pose estimation is a highly non-linear problem with large result space (6-DoF), so learned features are more suitable. But deep learning requires a large dataset to efficiently learn features.

Deep learning methods for object pose estimation can be divided into two approaches that are whole image based and patch image based. The whole image based methods require an object localization step [51, 50, 141, 58]. After that, the objects are recognized and their poses are estimated. [57, 164] separate object recognition and object pose estimation in two different models. Otherwise, [185, 153, 114] do not require a object recognition step. They introduce a powerful learning feature on different objects and different view points. In feature space, images from different objects and different views distinguished into clusters. That is an important factor to perform both object recognition and object pose estimation in a common model. These holistic methods perform rapidly. But, the precision drops quickly when object are occluded. The patch image based methods are effective ways to solve these problems. [80] demonstrates that neural networks coupled with a local voting-based approach can be used to perform reliable 3D object detection and pose estimation even with clutter and occlusion.

2.3.2 Camera pose regression

For machine learning approaches, handling camera relocalization is as a regression problem solved by a supervised learning. It is performed based on the informations known in advance

of each scene. The training phase uses labeled images (images and their corresponding camera poses). The testing phase uses the trained model to relocalize the camera from each unseen image independently.

[83] first propose to address camera pose estimation with the end-to-end deep learning strategy. A CNN is learned from whole images labeled with the camera poses. Then, the trained model is used to directly predict camera pose from every RGB images. [83] presents the way to adapt the GoogLeNet [167] model from classification to regression by properly modifying their final layers to regress camera pose: adding a fully connected (FC) layer before the two affine regressors. They leverage transfer learning from a pre-trained model of scene recognition by fine-tuning to regress camera pose. The features from the final convolutional layer are a high-level representation of the whole image, providing robustness to various lighting conditions, weather and other dynamic changes in the mainly unchanged scene. Training phase is performed with an objective loss function which is the sum of the translation error and the rotation error:

$$\mathscr{L}(I) = \|t - \hat{t}\|_2 + \beta \left\| q - \frac{\hat{q}}{\|\hat{q}\|} \right\|_2 \tag{2.19}$$

Where (t,q) and $(\hat{t},\hat{q})$ are ground truth and estimated translation-orientation pairs respectively (orientation being represented by a quaternion). β is a scale factor used to keep both error values to be approximately equal. However, the value of the scale factor is specific to each scene, which makes it remarkably hard to determine for a new scene.

[81] generates a probabilistic pose estimation by using dropout after every convolutional layer as a means of sampling the model weights of PoseNet. The dropout layers in PoseNet do not only play an important role to prevent over-fitting, but also provide an alternative interpretation for CNNs with dropout as a Bayesian model approximation. Instead of adding dropout layers, [28] uses Stochastic Variational Inference [65] (SVI) and Gaussian Process Regression (GPR) [140] as another way to provide the probability distribution for the 6DoF camera pose with one-time inference.

[177] is an improvement of PoseNet's architecture with spatial Long Short-Term Memory (LSTM) added after CNN layers. These features from convolutional layers are considered as an input sequence to a block of four LSTM units operating along four directions (up, down, left, and right) independently. On top of that, there is a regression part which encompasses fully connected layers for predicting the camera pose. [32] also applies LSTMs to predict camera translation only, but using short videos as an input aims at exploiting the temporal information to enhance camera pose estimation. Their method is a bidirectional recurrent neural network (RNN), which captures dependencies between adjacent frames refining

accuracy of the global pose. Both of the two architectures lead to an improvement in the accuracy of 6-DoF camera pose, outperforming PoseNet.

[116] trains an hourglass network, using skip connections between their encoder and decoder, to directly regress the camera pose. [79] proposes a different method based on a regression forest with hough voting approach to directly regresses the camera pose but it uses the same objective function combining translation error and rotational quaternion error. [128, 187] extend the set of training images with synthetic data.

[82] solves the ambiguity of the scale factor between location error and orientation error in the loss function of [83] by a novel loss function based on the re-projection error.

$$\mathscr{L}_g(I) = \sum_{X \in \mathscr{P}'} \|KT^{-1}X - K\hat{T}^{-1}X\|_2 \tag{2.20}$$

Where $\mathscr{P}'$ is a subset of all 3D points X in the scene are visible in the image I. K is the intrinsic calibration matrix of the camera. T and $\hat{T}$ are ground truth and estimated camera pose respectively. However, this loss function requires more time to compute and converges more difficultly.

Rather than using a single image, [175, 139, 100, 25] propose visual odometry methods based on localizing sequences of images. [175] trains a multi-task network to predict both 6D global pose and the relative 6D poses between consecutive frames, and report improvements over earlier neural network-based approaches, although their best results rely on using the estimated pose from the previous frame. In very recent work, [139] have added semantics to this approach. [100, 25] are also able to estimate camera pose of a monocular camera and the depth of its view while preserving the scale thanks to the training phase using stereo image pairs.

2.4 Image retrieval approach

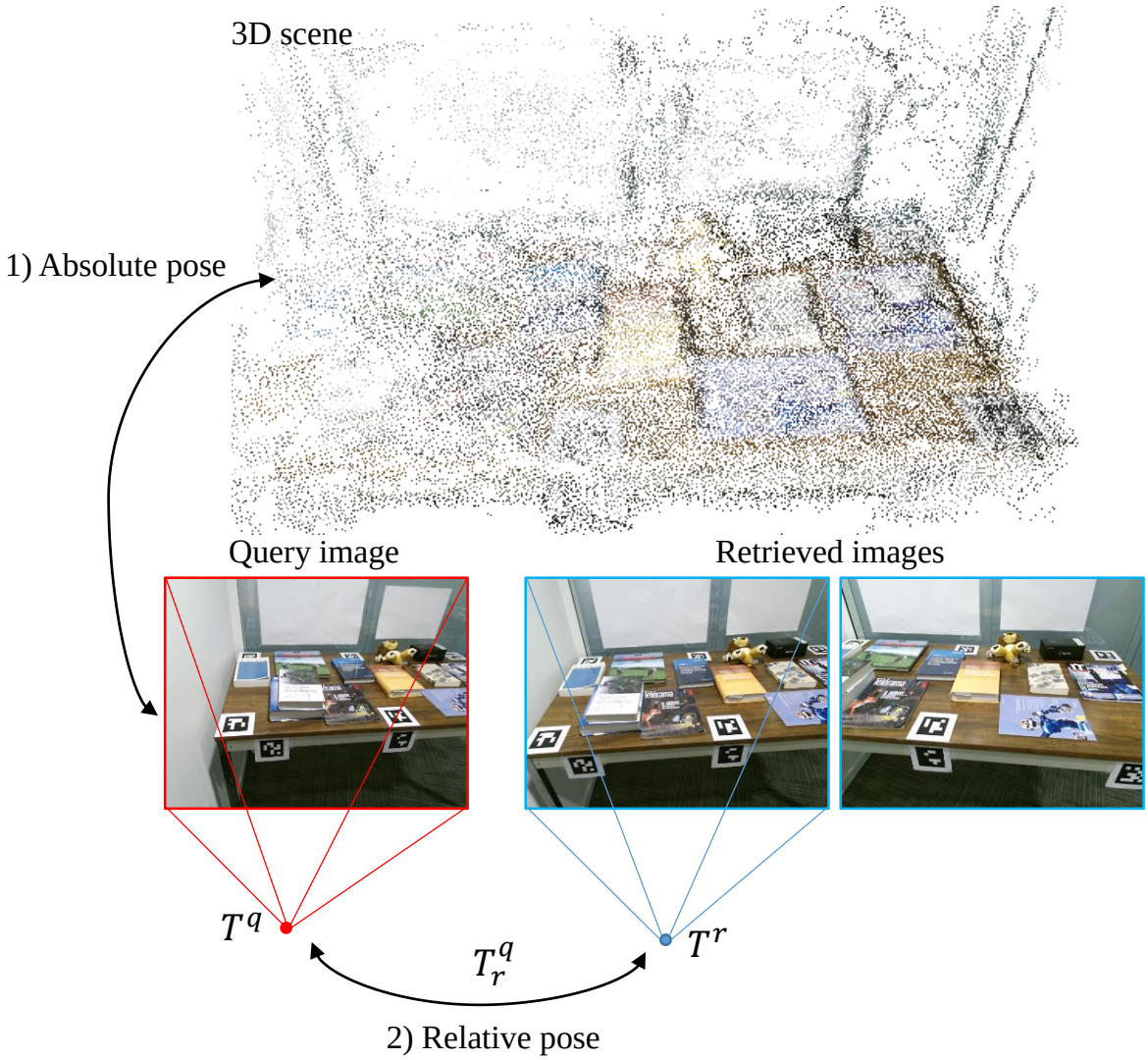


Fig. 2.10 Image retrieval approach. Camera relocalization is handled based on nearest image retrieval. The camera pose of the query image can be estimated by two ways: 1) Calculating absolute pose using the geometric approach. 2) Through defining relative pose between the query image and retrieved images.

The two above approaches directly compute camera pose of a query image based on geometric information correspondences or regression learning. On the contrary, image retrieval approach consists of indirect methods, that cast the camera relocalization as an image retrieval problem and provides a coarse pose about the query image, as illustrated in Figure 2.10. The final camera pose is obtained either through estimating the relative pose between the query image and retrieved images, or the absolute pose using a geometric

approach. In this section, we introduce the pipeline of the image retrieval approach, as shown in Figure 2.1-c), according to two steps: nearest images retrieval and camera pose estimation.

2.4.1 Nearest images retrieval

The aim of the nearest images retrieval methods is to retrieve a set of images that are in the database and that are similar to an input query image. If the database stores not only a set of images but also the poses of the cameras that have captured them, the pose of the retrieved image provides an information on the possible location of the query image. This image retrieval problem includes two steps: extracting image-level feature for both the query image and the database; searching nearest images in the database based on a feature similarity.

Image-level feature

Extracting image-level feature aims at producing a compact feature representing each image. This can be performed by global feature extraction and local feature aggregation.

For global feature extraction, the raw image can serve as a feature, with systematic resizing [63, 87]. Some methods produce this feature on the whole image by using GIST descriptors [133] or several random pixels [52]. With the recent emergence of deep learning, a new class of very efficient global feature have been created [5, 55, 84, 11, 94, 168] based on deep learning models. Deep learning based methods for camera pose regression need to be trained for a specific scene. In contrast, they have been used for the image retrieval task without special training, exploiting inherent domain transfer capability of neural network. It is also interesting to notice that the most discriminative features are extracted from mid-level convolutional layers instead of fully connected layers. Many approaches leverage classification networks [53, 155], and fine tune them with place recognition datasets.

For local feature aggregation methods, from each single image, a large number of local features are extracted. The feature aggregation is then performed in order to make an efficient image-level feature while reducing the dimensionality of the feature vectors. The aggregation process emphasizes specific features that are more relevant for the localization task. Early feature aggregation techniques for image retrieval rely on BoW representations [160, 132]. It involves counting the number of features associated with each cluster in a large vocabulary and creating a histogram for each set of features from each image. Thus it represents an image in a compact vector. A great example of BoW is FAB-MAP [34] which relies on SURF feature [13]. [48] uses for the first time BoW obtained from BRIEF features [29] along with the very efficient FAST feature detector [143]. It reduces in more than one order of magnitude the time needed for feature extraction, compared to SURF feature. Nevertheless

the use of BRIEF is neither rotation nor scale invariant. DBoW2 [125] extends that work using ORB feature [144] which is invariant to rotation and scale. Fisher vector (FV) based methods [135, 73] improve BoW based methods by using Gaussian Mixture Models (GMMs) to generate a probabilistic visual vocabulary. Another advantage of FV is that it can be computed from much smaller vocabularies, and therefore leads to a lower computational cost. Inspired by Fisher Vectors formulation, [72, 6] introduce Vector of Locally Aggregated Descriptors (VLAD) representation for image-based retrieval. VLAD is an extension of BoW. The difference between feature and its closest visual word is assigned to the final feature, instead of the visual word itself. In simpler terms, it first matches a feature to its closest cluster. Then, each cluster stores the sum of the differences of the features assigned to the cluster and the centroid of the cluster. The underlying idea behind VLAD representation have inspired various methods [76, 171]. Another aggregation solution using sum pooling of deep feature maps is presented in [10].

Compared to local feature aggregation, global features are considered less robust in viewpoint changes, occlusion and local variations in the image. Global feature extraction methods using deep learning has been less successful so far in local-level image retrieval. On most retrieval benchmarks, deep methods perform worse than conventional methods that rely on local feature aggregation. However, global features are computationally less intensive to extract and capture a comprehensive feature for each image.

Nearest images search

After extracting image-level feature, each image (query images as well as all the images of the database) is represented by a feature vector. Now, nearest images search can be processed in the same way as the feature matching described in Section 2.2.2. Due to the high dimension of image-level feature, comparison between features (with $L2$ norm as usually used metric) requires more time than local feature matching of Section 2.2.2. Thus, when the number of images in the database is very large, the searching approach of Section 2.2.2 cannot be immediately considered. Some works suggest to compress the features to improve the storage requirements and retrieval efficiency. The most common approach is to use unsupervised compression through Principal Component Analysis (PCA) or product quantization [135, 73]. Supervised dimensionality reduction approaches have also been proposed in [56].

There are some other approaches aiming at accelerating nearest images search. [180] considers directly the localization problem as a classification task. The database is classified into discrete camera pose classes. The image feature of the query image passes through the classification model to obtain a set of retrieved images in the same class. On the other hand, by leveraging the aggregation process, DBoW2 [48] which is used in ORB-SLAM [124]

builds incrementally a database that contains an invert index. It stores for each visual word in the vocabulary, the keyframes in which it has been seen, so that querying the database can be done very efficiently.

2.4.2 Camera pose estimation

Figure 2.10 shows two ways to estimate the camera pose of the query image based on information of retrieved images. First way, based on the geometric approach, [148, 124, 126] directly estimate the camera pose by using a part of 3D model of a scene being visible in retrieved images. [87, 52, 94, 11, 30] present the second way is to define camera pose through relative pose between the query image and retrieved images. From the known camera pose of one retrieved image T^r and the transformation matrix from the query image to the retrieved image T_r^q, the camera pose of the query image T^q is found by:

$$T^q = T_r^q T^r \tag{2.21}$$

Absolute camera pose estimation

Assume that a 3D point cloud model has been built from images in the database. [148] achieves a 3D model by performing SfM algorithm. The one in [124] is obtained in real-time by mapping frames. Similar to the geometric approach in Section 2.2.2, the 2D-3D point correspondences are required for the absolute camera pose estimation of the query image in the world coordinate system.

A set of sparse feature extracted from the query image are matched with keypoints of retrieved images. Note that only keypoints of retrieved images that are associated with 3D points are nominated for matching. From 2D-2D matches, a set of 2D-3D point correspondences is established. The homography transformation matrix that is calculated by 2D-2D matches can be used to filter these correspondences. The precise camera pose is then computed from 2D-3D correspondences based PnP and RANSAC algorithms.

[124] attempts to reduce the computational complexity of feature matching by using DBoW2 [48]. DBoW2 reports an additional benefit of the bags of words representation for feature matching. To compute the correspondences between two sets of ORB features, it can constraint the brute force matching only to those features that belong to the same node in the vocabulary tree at a certain level, speeding up the search. [124] also refines the correspondences with an orientation consistency test [125] that discards outliers ensuring a coherent rotation for all correspondences.

Relative camera pose estimation

The correlation information between the query image and retrieved images allows to estimate relative camera pose estimation. [87] presents a first solution in order to improve the camera relocalization in PTAM [86]. They exploit the fact that the SLAM system stores full RGB keyframes, and the process relocalization directly from these. Instead of extracting some forms of interest points and features from keyframes and then matching a novel view against them, they find that keyframes are sufficiently densely distributed so that the full image can be used as a descriptor: for each keyframe added to the map, generating a sub-sampled 40×30 pixel image, apply a Gaussian blur, and finally subtract the mean image intensity. This zero-mean heavily blurs image forms the keyframe's descriptor. When tracking is lost, each incoming video frame is similarly subsampled, blurred, and mean-normalised. Next, nearest keyframes are found by using methods presented in Subsection 2.4.1. The camera pose is then set to the position of the nearest keyframe with the lowest image difference. The rotation of the camera is estimated by aligning the requested image with the nearest keyframe by minimizing the square difference of the sum over the whole image. They minimize over the three-parameter group SE(2) in image (pixel) space, allowing ten iterations for convergence. Finally, the resulting 3-DoF image-space transformation is converted to a best-fit 3D camera rotation by considering the motion of a few virtual sample points placed in the image, in a procedure similar to the unscented transform.

[52] provides a camera relocalization solution for KinectFusion system [130] dedicated to only depth sensor. While [87] uses an appearance based method, [52] uses 3D point cloud obtained from depth images to determine the relative transformation between the query image and the nearest keyframes. This transformation can be for instance computed by employing a robust version of the Iterative Closest Point (ICP) algorithm [146].

[11] proposes a Siamese network to generate global features using a continuous metric learning loss based on camera frustum overlap. Given a query image and its nearest retrieved neighbor, their differential pose is defined based on this Siamese network.

2.5 Hybrid approach

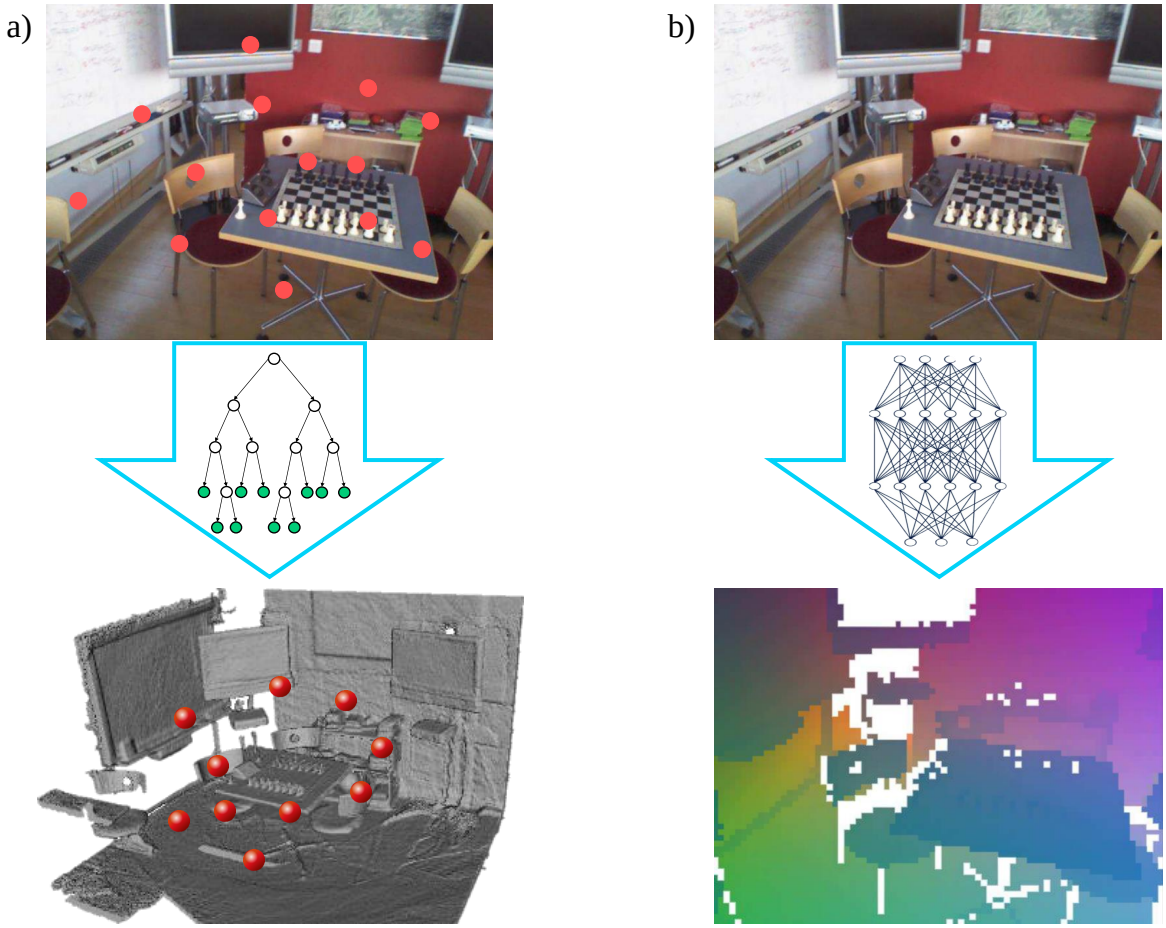


Fig. 2.11 Hybrid approach. a) Sparse random forest based methods. b) Dense deep learning based methods.

Figure 2.1-c shows the pipeline of the hybrid methods. Camera pose is estimated by combining both machine learning approaches and geometric approaches. Machine learning approaches are applied to learn and predict the 3D position of each pixel in world coordinate system. From these correspondences, geometric methods infer camera pose. Instead of directly matching 2D keypoint in a query image to 3D point cloud by feature-based methods (see details in Section 2.2), a world coordinate regression model defines rapidly and efficiently 2D-3D point correspondences. We present world coordinate regression models according to two approaches as shown in Figure 2.11: sparse random forest based methods and dense deep learning based methods.

2.5.1 Sparse random forest based methods

The methods are known as voting methods, which have been successfully used in [46] for object detection. Though, in order to obtain a final 6-DoF camera pose (camera translation and rotation), each pixel does not vote directly for a global quantized 6-DoF because of the dimensionality of the camera pose estimation manifold space. Each pixel instead makes a 3D continuous prediction about its own 3D position in the world coordinates system or the camera coordinates system.

The first hybrid method [156] using a random forest proposes predicts directly corresponding 3D points in world space for all pixels in an RGBD image (each pixel in the image effectively denoting a 3D point in camera space). By generating predictions for thousands of pixels, their approach avoids the explicit detection, description and matching of keypoints typically required by traditional 2D-3D correspondences based methods. This makes it simpler and faster to find larger number of correspondences. Moreover, all features used in [156] are based on simple pixel comparisons [97] and so are extremely fast to evaluate. At testing time, a random subset of pixels from RGB-D image pass through the regression forest to predict 3D world coordinates. The camera pose is then obtained by using Kabsch [77] and RANSAC [43] algorithms to optimize a geometric energy function:

$$\begin{aligned} T^* &= \arg\min_T E(T) \\ E(T) &= \sum_i \rho\Big(\min \|\hat{p}_i - T^{-1} x_i\|_2\Big) \end{aligned} \tag{2.22}$$

Where x_i is 3D coordinates of pixel i in the camera coordinate system. $\hat{p}_i$ is 3D world coordinates prediction. ρ is a top-hat error function.

This initial method have been improved in [59], relying on multiple regression forests to generate a number of camera pose hypotheses. The hypotheses are then clustered, and the mean pose of the cluster minimizing the reconstruction error is selected as the result. [176] introduces mixtures of anisotropic 3D Gaussians to represent the uncertainty associated with the regression forest prediction at leaf nodes and significantly improve the 6-DoF estimation by embedding this information within the full camera pose regression step. However, the methods [156, 59, 176] are limited by the use of RGB-D images in both training and testing phase.

As an extension of [156], [22] uses an auto-context regression forest from only RGB image patches with lower accuracy. [117] performs RGB relocalization by estimating an initial camera pose using a regression forest, then queries a nearest neighbor keyframe image and refines the initial pose by sparse feature matching between the camera input image and

the nearest keyframe. [115] maps parameters between regression forests and neural networks to leverage the performance benefits of neural networks for dense regression while retaining the efficiency of random forests for evaluation. [118] stores a priority queue of non-visited branches whilst passing a feature vector down the forest during testing, and then backtracks to see whether some of those branches might have been better than the one chosen. [119] makes use of both points and line segments (segment feature being based on the features of a points sampling) to achieve more robust relocalization in poorly textured areas and/or in case of motion blur. [31] proposes a new method based on pre-trained regression forest. This method permits to transfer the pre-trained model to a new scene through online adaptation.

2.5.2 Dense deep learning based methods

On the other hand, various related methods have used deep learning approaches to define 2D-3D correspondences. DSAC [20] is the first method using a VGG style architecture for scene coordinate regression. It takes an image patch of 42×42 pixels as input and produces one scene coordinate prediction for the center pixel. This design is not so efficient because the CNN processes neighboring patches independently without reusing computations. They sample 40×40 patches per image instead of making a dense prediction for all possible patches. [20] also shows how to replace the RANSAC stage of the conventional pipeline with a probabilistic approach to hypothesis selection that can be differentiated, allowing end-to-end training of the full system.

[102, 27] use a fully-convolutional encoder decoder network to predict scene coordinates for the whole image at once, thus taking into account the global context. This prevents patch sampling in [20], but needs significant data augmentation to avoid overfitting. [23] that is known as DSAC++ significantly improves the results of [20]. They use a Fully Convolutional Network (FCN) [110], but without upsampling layers. Their FCN takes a RGB image of 640×480 pixels as input and produces 80×60 scene coordinate predictions. They regress more scene coordinates in less time than their previous work [20]. Whilst, they also show how to avoid the need for a 3D model at training time (albeit at a cost in performance). Very recently, [103] has shown how to use an angle-based reprojection loss to remove [23]'s need to initialize the scene coordinates with a heuristic when training without a model. However, despite all of these advances, none of these papers remove the need to train on the scene of interest in advance in order to generalize the learn model.

2.6 Camera relocalization datasets

In this section, we first introduce the public datasets that will be used for our experiments in Chapters 3 and 4. Then we present some metrics that are often used to evaluate the accuracy of camera relocalization methods.

2.6.1 Datasets

Dataset	Scene	Train	Test	Scale	Spatial Extent (m^2)	Data Type	GT method
7-Scenes	Chess	4000	2000	Room	6	RGB-D	KinectFusion
	Fire	2000	2000		3		
	Heads	1000	1000		2		
	Office	6000	4000		6		
	Pumpkin	4000	2000		6		
	Red Kitchen	7000	5000		12		
	Stairs	2000	1000		5		
CoRBS	Human	1273	1273	Room	10	RGB-D	Tracking System
	Desk	1190	1190		9		
	Electrical Cabinet	950	950		6		
Cambridge Landmarks	King's College	1220	343	Outdoor	5600	RGB	SfM
	Street	3015	2923		50000		
	Old Hospital	895	182		2000		
	Shop Facade	231	103		1000		
	St Mary's Church	1487	530		5000		
BCOM	4 sequences (Train a sequence and test on 3 remaining ones)	2400	2400	Desktop	5	RGB-D	Markers
DynaScenes	Dyna-01	669	1362	Desktop	4	RGB	Tracking System
	Dyna-02	831	2211		4		
	Dyna-03	743	1982		4		
	Dyna-04	958	3504		4		

Table 2.1 Camera relocalization datasets

Datasets for visual based camera pose estimation include images and corresponding annotations (camera pose for each frame: 3D translation and 3D rotation in the world coordinate system) as well as the intrinsic parameters of the vision sensors. Depending on the used sensor, images can be RGB or RGB-D. As introduced in Section 1.2, both camera localization and camera relocalization are to estimate camera pose. While localization methods require continuous sequences to measure accuracy, the relocalization methods can evaluate their accuracy for each frame independently. Thus, datasets of unsorted list of images and sampling images are also used. Table 2.1 shows data information of five datasets that we use to evaluate our methods and compare with state-of-the-art methods. Each dataset

consists of some different sequences captured under different environment conditions. This is very interesting to consider the ability of camera relocalization methods to handle the challenges mentioned in Section 1.3.

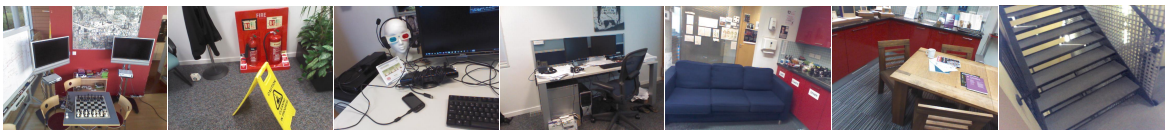


Fig. 2.12 7-scenes dataset: from left to right, this dataset consists of chess, fire, heads, office, pumpkin, red kitchen, stairs.

7 scenes dataset is introduced by [156]. This dataset has 17000 images across seven scenes at room-scale. Each scene includes some sequences which are captured around a single room and annotated by using KinectFusion [130]. A 3D scene model is also provided by KinectFusion. The ground truth computed from KinectFusion are not really perfect. By visualizing the point cloud of each frame to compare with 3D model of the scene, we find that some frames do not match to the 3D model correctly. This affects the pixel labeling in our learning database generation phase where we wish patches corresponding to the same 3D location of the scene to have the same label. However, it is enough accurate to evaluate camera relocalization methods. The data is extremely challenging with pure rotation or fast movement of camera that creates many textureless images. Figure 2.12 shows some examples of 7 scenes dataset. One challenge of the office, red kitchen and stairs scenes is that they have repeated patterns (several similar desks, chairs, stairs).

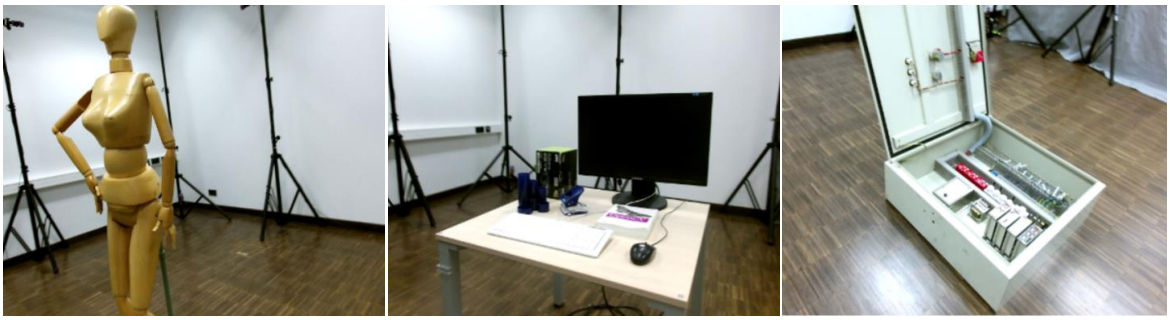


Fig. 2.13 CoRBS dataset: Human, Desktop, Electrical Cabinet (from left to right).

CoRBS dataset [179] is more accurate than 7-scenes dataset thanks to the use of multiple sensors. Visual data is captured by using a Kinect v2. The ground truth is obtained by an external motion capture system providing a tracking based on passive spherical markers

attached to the Kinect. Each scene contains a dense 3D scene model which is created via an external 3D scanner. An overview of the three scenes is provided in Figure 2.13. Each scene contains a main big object on flat surfaces and texture-less background. The Human scene is a simple wooden manikin whose surface is predominantly convex. The Desk scene incorporates more complex geometry and includes a slightly reflective screen. The most challenging scene is that of the electrical cabinet with a dense components in an interior space.

Fig. 2.14 Cambridge Landmark dataset: King's College, Street, Old Hospital, Shop Facade, St Mary's Church (from left to right).

Cambridge landmarks dataset is an outdoor urban relocalization dataset with five scenes introduced in Figure 2.14. The dataset was generated using SfM techniques [186] which is used as ground truth measurements for this paper. A Google LG Nexus 5 smartphone was used by a pedestrian to take high definition video around each scene. This video was subsampled in real-time at 2Hz to generate images to input to the SfM pipeline. There is a baseline of about $1m$ between each camera position. This dataset covers many challenges such as blur motion, occlusion, illumination change. However, since SfM is still an approximate method, obtaining the exact ground truth of outdoor scenes under various conditions is challenging. Additionally, a SfM model containing both training and testing images is built and the resulting poses of the query images are used as ground truth. Yet, this approach relies on local feature matches and can only succeed if the testing images and training images are sufficiently similar.

BCOM dataset [1] is our dataset captured from one indoor scene. This dataset consists of four sequences corresponding to four absolutely different trajectories in the same scene with changing illumination and camera rotation. A kinect v2 is used to capture RGB and depth images. The ground truth for camera pose is performed by using multiple fiducial markers on the scene. Marker based methods limit view point of camera because of the appearance of markers in each frame. However, they allow us to create sequences far away from each other shown in Figure 2.15. This database aims at evaluating some generalization capabilities of the methods.

[1] https://github.com/duongnamduong/bcom-dataset

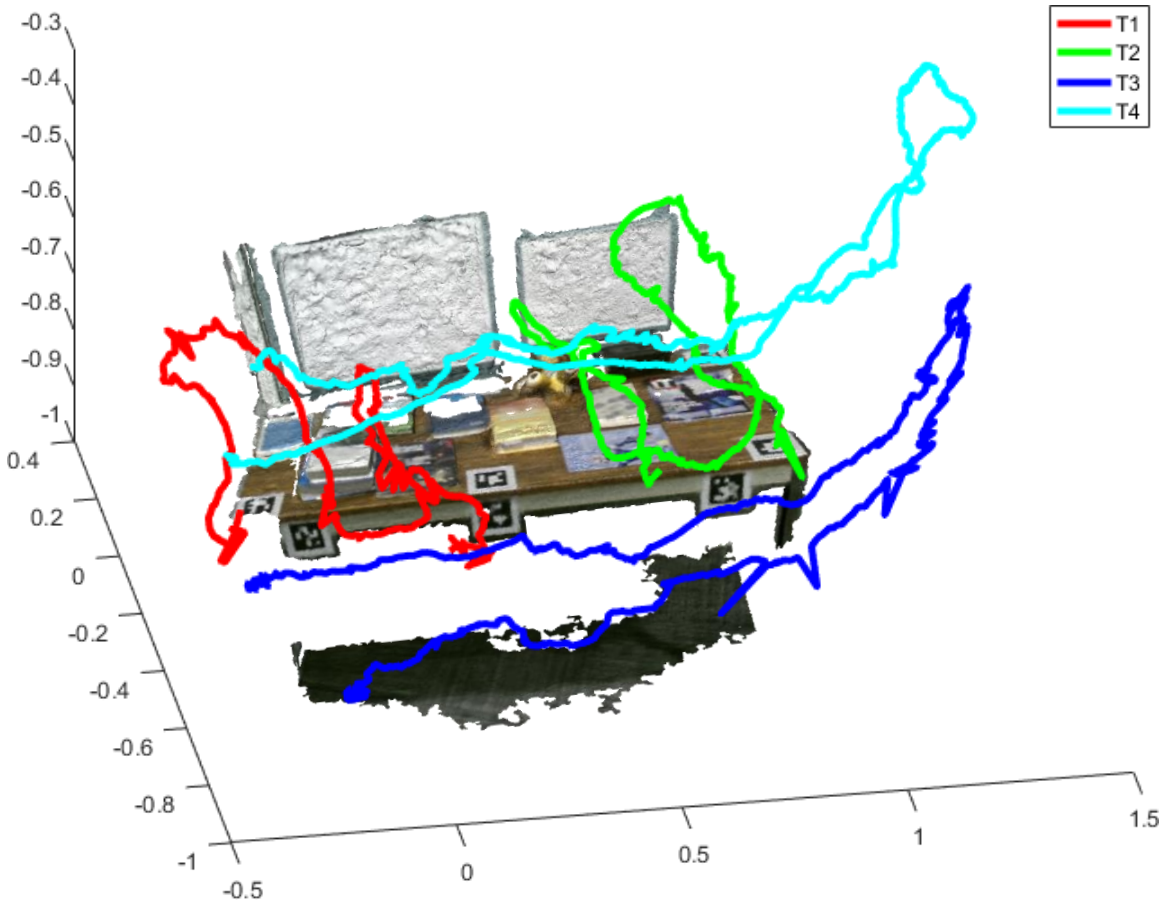


Fig. 2.15 BCOM dataset: including four sequences.

The DynaScenes dataset [2] is a completely new dataset of dynamic scenes that we have created. The ground truth of camera pose is acquired accurately by using a HTC Vive tracker rigidly attached on the camera. The camera can move around within a visible area of two lighthouse base stations. This allows us to continuously track the orientation and trajectory of the tracker. A rigid transformation matrix between tracker pose and camera pose is estimated by using the hand-eye calibration method [174]. Camera pose is found based on tracker pose and this transformation matrix. We perform a synchronization to get image frame and tracker pose at the same time.

This dataset consists of four scenes in desktop-scale for an area of about $3m^2$. Each one contains three absolutely different image sequences of the same scene (one training sequence and two testing sequences). Each sequence has from 600 to over 2000 RGB images. The testing sequences are extremely challenging with occlusion, illumination changes and gradually moving objects. To evaluate precisely camera relocalization methods in dynamic

[2] https://github.com/duongnamduong/DynaScenes-dataset

Infos	DynaScene-01			DynaScene-02			DynaScene-03			DynaScene-04		
	Train	Test		Train	Test		Train	Test		Train	Test	
	Seq-00	Seq-01	Seq-02	Seq-00	Seq-01	Seq-02	Seq-00	Seq-01	Seq-02	Seq-00	Seq-01	Seq-02
#Frames	669	681	681	831	1136	1075	743	842	1140	958	1395	2109
Occlusion	No	No	No	No	Yes	No	No	No	No	No	Yes	Yes
Illumination change	No	No	No	No	No	No	No	Yes	No	No	Yes	Yes
Moving objects	0%	0%	30%	0%	0%	60%	0%	0%	100%	0%	0%	100%

Table 2.2 DynaScenes dataset. A RGB images dataset is used to evaluate camera relocalization methods in dynamic scenes.

Fig. 2.16 DynaScenes dataset. This is some examples in Dyna-03 scene.

scenes, we setup some movable objects on a black background to eliminate rigid features from the background. More detail characteristics of this dataset are given in Table 2.2.

2.6.2 Metrics

The accuracy of camera relocalization methods is evaluated by comparing camera pose estimation and camera pose ground truth for each frame. Let (R,t) and $(\hat{R},\hat{t})$ be the ground truth and the camera pose estimation respectively.

- **Translation error**. It is the distance between the translations t and $\hat{t}$. The translation error is simply measured by the Euclidean distance:

$$E(t) = \|t - \hat{t}\|_2 \tag{2.23}$$

- **Rotation error**. According to [70], the distance between rotations R and $\hat{R}$ represented by unit quaternions q and $\hat{q}$ is the angle of the difference rotation represented by the unit quaternion $\Delta_q = q\hat{q}^*$, q^* denotes the quaternion conjugate. We can extract the rotation error from the first component of Δ_q by:

$$E(R) = 2\arccos|q_1\hat{q}_1 + q_2\hat{q}_2 + q_3\hat{q}_3 + q_4\hat{q}_4| \tag{2.24}$$

The rotation error also can be calculated from the difference rotation matrix $\Delta_R = R\hat{R}^T$:

$$E(R) = \arccos \frac{tr(\Delta_R) - 1}{2} \tag{2.25}$$

Where $tr(\Delta_R)$ is the trace of Δ_R.

To evaluate camera pose estimation error for a scene, we can compute the mean or the median of all translation errors and all rotation errors. It provides an overview accuracy of measurement. [156] reports another metric that consists of the percentage of test frames for which the inferred camera pose is essentially correct. They employ a fairly strict definition of correct: the pose must be within $5cm$ translational error and 5° rotation error of the ground truth. The belief is that correctly localizing the pose under this metric might already be sufficient for some augmented reality applications.

2.7 Conclusion

In above sections, we presented the state-of-the-art of camera relocalization methods according to four approaches: geometric approach, machine learning approach, image retrieval approach, hybrid approach. Although these approaches have proven their strengths, they are still limited by different challenges e.g. accuracy, run-time, training time, scene dynamics, large scale, to apply to a generic camera relocalization system.

Geometric approach is a basic solution based on 2D-3D point correspondences. Geometric approach is simple, accurate and especially useful when the query image has large distance to the training images. However, such methods are restricted to a relatively small scene due to the fact that matching cost, depending on the matching scheme employed, can grow exponentially with respect to the number of keypoints. Matching of local features can be noisy and unreliable on scenes with repeated patterns. Besides, to achieve accurate and effective 3D model, the SfM algorithms take much more time to build and optimize the 3D model.

Image retrieval approach improves the matching time compared to the geometric approach by finding coarse pose from nearest retrieved images. Precise camera pose can be obtained by defining 2D-3D point correspondences between the query image and 3D points seen by the retrieved images or measuring similarity of images for relative pose estimation. However, these methods are often not accurate if the query frame is captured from a viewing pose that is far from those in the database. For localization system, this approach needs to store a large set of keyframes. Consequently, memory usage as well as processing time increase with respect to the size of models.

Machine learning approarch has appeared to overcome these limitations (memory usage for large scale, matching time) of geometric and image retrieval approaches. It provides a compact end-to-end camera pose estimation solution. Contrary to the geometric approach, it struggles to generalize beyond their training data. These methods can estimate camera pose in real-time from each image on a GPU but the training phase takes hours or even days for a small scene. However, important limitations of these methods lie in their moderate accuracy and the lack of confidence score for each pose estimation. These methods are significantly less accurate than geometric based methods as well as image retrieval based methods. This approach seems more consistent with image retrieval than with camera pose regression.

Hybrid approarch is based on both deep learning approach and geometric approach with aims at benefiting from each. The machine learning part defines the points correspondences faster than the geometric based methods. Then geometric part infers camera pose from these correspondences. Although these methods achieve higher accuracy, they need thousands of

predictions about scene coordinates, so that time increases a lot to infer optimal camera pose by RANSAC algorithm.

Inspired by hybrid approach, we propose methods to balance computational time and accuracy. Our methods improve the machine learning part of hybrid approaches aiming at simultaneously reducing the number of 2D-3D point correspondences and increasing their accuracy in order to accelerate computation and keep high accuracy of camera pose. However, in all camera relocalization approaches above, geometrical model or learned model are built from rigid scenes. And they are not updated in real-time. Therefore, for dynamic scenes, especially when some objects move, these models are no longer accurate to infer camera pose. We expand continously our real-time accurate hybrid method based on the regression forest to our adaptive regression forest method. The adaptive regression forest model is able to adapt rapidly to real-time changes to tackle camera relocalization in dynamic scenes. We present our contributions of this thesis in Chapter 3 and 4.

Chapter 3

Balance between Accuracy and Runtime for Camera Relocalization

Contents

3.1 Introduction

The state-of-the-art presented in Chapter 2 has shown accurate methods for camera relocalization. Especially, the hybrid approaches using 3D world coordinate regression currently achieve a high pose accuracy. However, having a both real-time and accurate method is still a challenge. In this chapter, we present our hybrid method combining machine learning and geometric approaches for real-time and accurate camera relocalization from a single RGB image. The main limitation of the state-of-the-art hybrid methods is due to the computational time of the geometric part of the process. This computational time is high, due to the excessive number of data with a low accuracy used as inputs of this geometric part. Therefore, our method focuses on improving the data provided by the machine learning part of the process. This is performed by two axes: decreasing the number of data handled in each step of the machine learning part, and increasing their relevancy at the same time.

This chapter is organized as follows:

- **Section 3.2** presents our camera relocalization from local patches using machine learning approaches. We tackle uncertainty of end-to-end camera pose regression by using multi-output camera pose regression based on local patches of a single RGB image.
- **Section 3.3** presents a light CNN, that we called *xyzNet*, to efficiently and robustly regress 3D world coordinate from local patches. Then, the geometric information about 2D-3D correspondences removes ambiguous predictions and calculates more accurate camera pose.
- **Section 3.4** introduces our sparse feature regression forest to further improve the machine learning part in hybrid approaches. We propose a novel split function that uses a whole feature vector instead of classical binary test function to obtain higher accuracy of 2D-3D point correspondences.
- **Section 3.5** gives some conclusions and perspectives.

3.2 Camera pose regression based on local patches

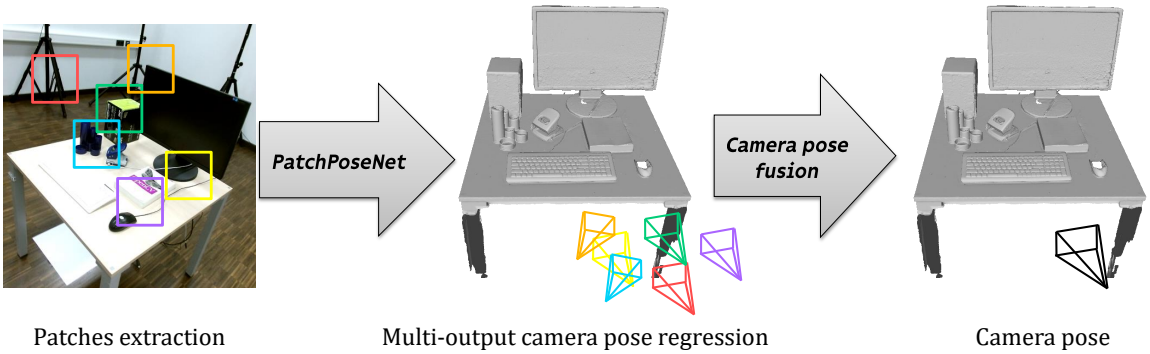


Fig. 3.1 PatchPoseNet pipeline: From a set of relevant patches extracted on a RGB image, PatchPoseNet predicts multi-output camera poses. The final camera pose is computed by fusing the camera poses.

We propose a camera pose regression based on the machine learning approach. Instead of regressing camera pose from a whole RGB image as in [83], we use the MIMO (Multiple-Input Multiple-Output) strategy to overcome the uncertainty of deep learning by predicting multi-output camera pose from local patches. The pipeline of our method is summarized in Figure 3.1. In the following sections, we first present how to generate multi-output camera pose regression from a single RGB image and combine them to achieve a final camera pose prediction in Section 3.2.1. Then, Section 3.2.2 shows our experiments and the limitations of this method.

3.2.1 Multi-output camera pose regression

A problem of end-to-end regression for camera relocalization lies in the lack of confidence score. Therefore, predictions are uncertain. To solve this problem, [81, 32] create a probabilistic model of results by using dropout layers after every convolutional layer as a means of sampling the model weights. Our method, instead, uses a set of patches to generate a set of probabilistic results.

Inspired by the success of deep learning with local patches for object detection and segmentation [58, 50, 80], we propose a CNN based on VGG style architecture for inferring camera pose from each local patches. We called it *PatchPoseNet*. Firstly, we extract local patches of RGB image which contain significant information. Homogeneous patches, such as on the sky or on a road, do not provide distinctive information about camera pose in the scene. The use of these patches produces some noise. That is why we filter them. Indeed, the

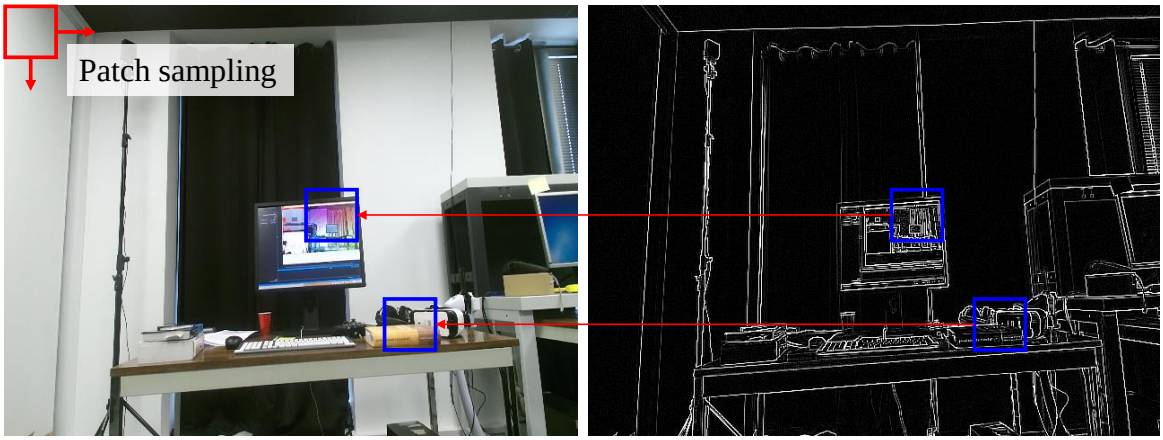


Fig. 3.2 Patch extraction. From sampled patches (red patch), we select patches which have maximum magnitude of image gradient (blue patch).

patch extraction is processed as described in Figure 3.2. From RGB image input, we first sample a grid of 50×50 patches with a fixed size of $0.1 \min(W,H)$, where W, H are width and height of image respectively. We then calculate the gradient of the image input. Finally, we select 500 patches per image. The chosen patches have the greatest magnitude of image gradient.

CNN training phase: We extract patches from the training images. Each patch is attached with a label which is the camera pose ground truth of the corresponding image. Our network is trained by minimizing an Euclidean loss function (Equation 2.19) using stochastic gradient descent.

CNN testing phase: To estimate camera pose from an unseen RGB image, we extract, following the same strategy as in training, a set of patches. All patches are passed through the learned model to give multi-output camera pose. We combine all the predictions to make a final camera pose prediction by performing a non-parametric clustering mean-shift. This allows to reduce the effect of noisy predictions and determine more accurate estimation. Figure 3.3 shows the fusion of camera poses by calculating the mean and mean-shift of votes. The result of mean-shift is more accurate than the one of the mean of predictions.

3.2.2 Experiments

Comparasion. We evaluate our method, *PatchPoseNet*, on two datasets: Cambridge Landmarks and 7 scenes. We compare our method to geometric based method (Active Search [151]) and to machine learning based methods (PoseNet [83], Bayesian PoseNet [81],

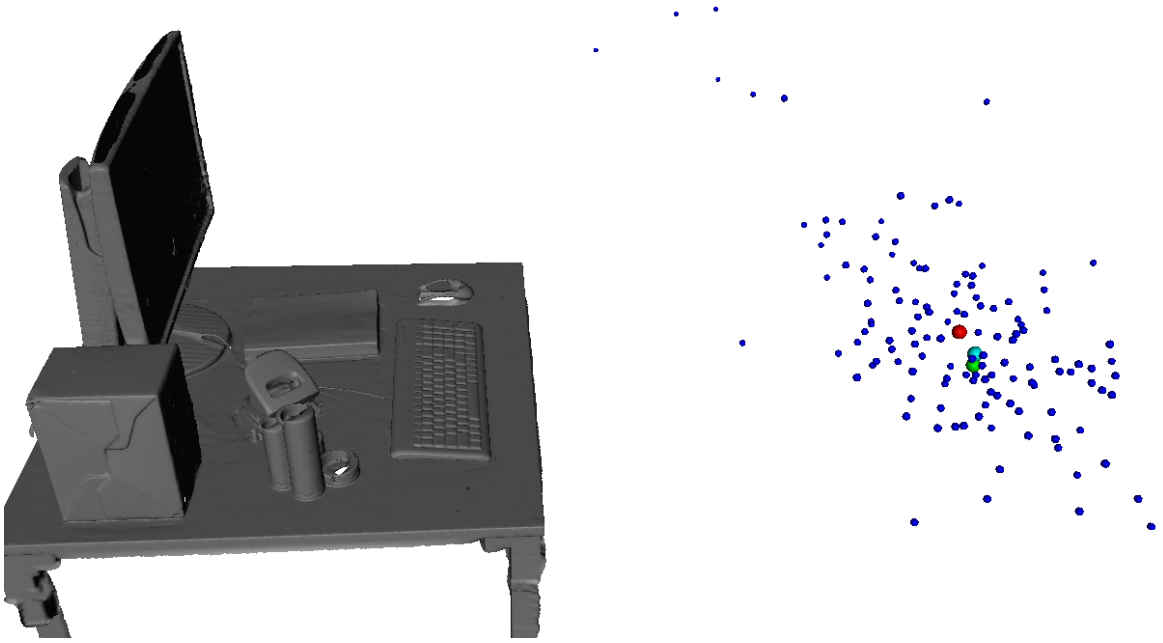


Fig. 3.3 Camera pose fusion. A set of votes for camera pose: blue points represent the translation of camera poses. Green point is the ground truth. Red point and cyan point are mean and mean-shift of camera translations respectively.

PoseNet 2 [82], LSTM-PoseNet [177]). These methods only use RGB image for camera relocalization.

Table 3.1 reports that our method outperforms PoseNet [83] and Bayesian PoseNet [81] on all scenes. It proves that our method provides an effective solution in order to address uncertainty of deep learning regression. However, our results are not as good as PoseNet 2 or LSTM-PoseNet on all scenes. And the accuracy of our method is far lower than geometric methods.

Conclusion. We proposed a solution using multi-output predictions from data to tackle the problem of uncertainty in deep learning regression. We showed results that confirms the effectiveness of our method compared to end-to-end learning from whole image. However, it is also significantly less accurate than geometric based methods. Our method is still restricted to the coefficient weight of the loss function which combines rotation and translation error by coefficient weight. In addition, the multi-patch extraction from each image makes a huge training data. This leads to the training phase more difficult to converge. It takes more than a day on a GPU to train a scene with thousands of images. Moreover, patches extracted from the same position on the scene with different camera rotations are almost unchanged. This

Table 3.1 Median pose errors. Comparison of our methods with the state-of-the-art methods on Cambridge Landmarks dataset and 7 scenes dataset.

Scene	Active Search [151]	PoseNet [83]	Bayesian PoseNet [81]	PoseNet 2 [82]	LSTM-PoseNet [177]	**PatchPoseNet**
King's College	**0.42m, 0.55°**	1.92m, 5.40°	1.74m, 4.06°	0.88m, 1.04°	0.99m, 3.65°	1.28m, 4.86°
Old Hospital	**0.44m, 1.01°**	2.31m, 5.38°	2.57m, 5.14°	3.20m, 3.29°	1.51m, 4.29°	2.12m, 5.20°
Shop Facade	**0.12m, 0.40°**	1.46m, 8.08°	1.25m, 7.54°	0.88m, 3.78°	1.18m, 7.44°	1.33m, 6.98°
St Mary's Church	**0.19m, 0.54°**	2.65m, 8.48°	2.11m, 8.38°	1.57m, 3.32°	1.52m, 6.68°	1.97m, 7.05°
Street	**0.85m, 0.83°**	3.67m, 6.50°	2.14m, 4.96°	20.3m, 25.5°	-	2.12m, 5.84°
Chess	**0.04m, 1.96°**	0.32m, 8.12°	0.37m, 7.24°	0.13m, 4.48°	0.24m, 5.77°	0.23m, 6.81°
Fire	**0.03m, 1.53°**	0.47m, 14.4°	0.43m, 13.7°	0.27m, 11.3°	0.34m, 11.9°	0.40m, 8.32°
Heads	**0.02m, 1.45°**	0.29m, 12.0°	0.31m, 12.0°	0.17m, 13.0°	0.21m, 13.7°	0.24m, 10.8°
Office	**0.09m, 3.61°**	0.48m, 7.68°	0.48m, 8.04°	0.19m, 5.55°	0.30m, 8.08°	0.43m, 6.23°
Pumpkin	**0.08m, 3.10°**	0.47m, 8.42°	0.61m, 7.08°	0.26m, 4.75°	0.33m, 7.00°	0.39m, 6.17°
Red Kitchen	**0.07m, 3.37°**	0.59m, 8.84°	0.58m, 7.54°	0.23m, 5.35°	0.37m, 8.83°	0.50m, 7.65°
Stairs	**0.33m, 2.22°**	0.47m, 13.8°	0.48m, 13.1°	0.35m, 12.4°	0.40m, 13.7°	0.44m, 9.83°

makes our method reduces the discrimination with rotation. In the next section, we present our hybrid method to overcome limitations of the end-to-end learning approach.

3.3 3D world coordinates learning based on local patches

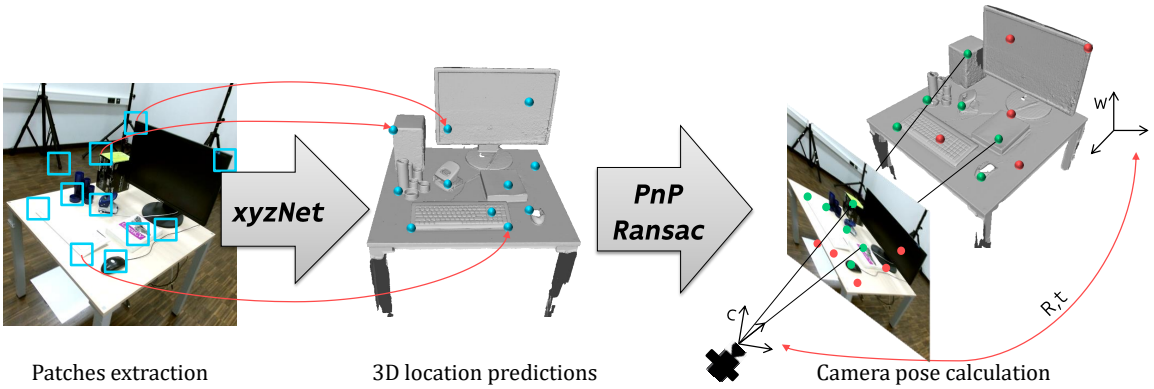


Fig. 3.4 xyzNet Camera Relocalization Pipeline: From a set of relevant patches (blue squares) extracted on each RGB image, xyzNet predicts a set of 3D positions (blue points) in the world coordinate system. PnP and Ransac algorithms are then used to filter inliers (green points) and eliminate outliers (red points). Finally, camera pose is computed by re-running PnP once on all the inliers.

In this section, we propose a hybrid method based on both machine learning approach and geometric approach. Figure 3.4 illustrates our pipeline. Our main contribution is that we propose a real-time camera relocalization method. Inspired by *PatchPoseNet* in Section 3.2, we use sparse patches instead of a whole image for camera relocalization. But contrary to *PatchPoseNet*, we present a light convolutional neural network for scene coordinate regression from local patches. We called this network *xyzNet*. Indeed, the high dimensionality of the camera pose research space is likely to result in a poor estimation. Thus, instead of end-to-end camera pose regression, each patch gives a 3D world coordinate prediction. The originality of our method is to use only patches extracted from the detection of keypoints. The patch extraction based on keypoints is more discriminative on appearance. Therefore, our network predicts efficiently and robustly correspondences between 2D image pixels represented by the centers of RGB patches with fixed size and their 3D positions in the world coordinate system. This allows our method to have a trade-off between runtime and accuracy for camera relocalization. Furthermore, thanks to the use of patches instead of a whole image, our method is robust to occlusion. From 2D-3D point correspondences that are estimated by our network, we run PnP and RANSAC algorithms in order to estimate camera pose. The final camera pose is attached with a confidence score based on the number of inliers.

In the following paragraphs, we present our method including three main steps. Firstly, in Subsection 3.3.1, we express patch extraction based on keypoint detection and 3D world coor-

dinate labelling for training. Then, in the Subsection 3.3.2, we introduce *xyzNet* architecture, ways to train and predict 3D points in world coordinates system. Camera pose calculation from 2D-3D point correspondences is given in Subsection 3.3.3. Subsection 3.3.4 shows and discusses our results on different datasets and provides some conclusions and perspectives.

3.3.1 Patch extraction and labelling

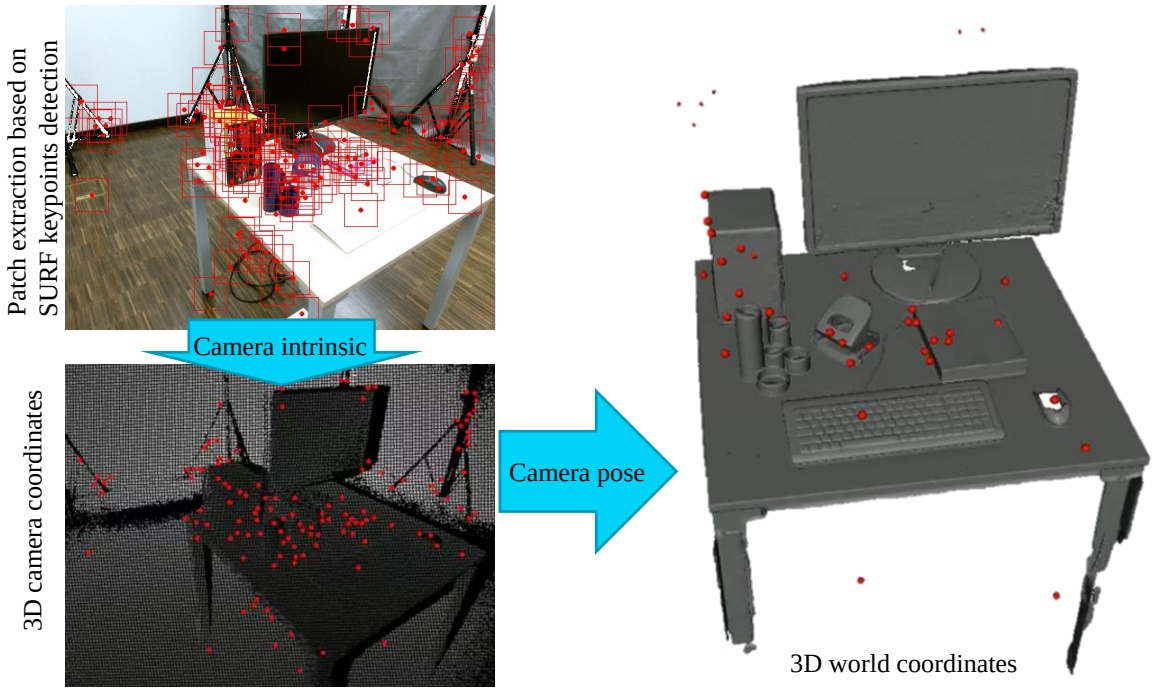


Fig. 3.5 Patch extraction and labelling.

Figure 3.5 presents the means of extracting patches and labelling them. From each RGB image, we extract a set of patches. Instead of randomly choosing them, we deliberately get patches around keypoints to target only regions with a relevant appearance. Thus, each chosen patch is an image region around a keypoint with a fixed size. We use SURF detector [12] to detect some sparse points which are scale and rotation invariant points. This enhances the ability to locate 3D positions from patches. The Hessian minimum threshold is chosen to ensure detection of hundreds of keypoints even on blurry or texture-less images. So, from every image, we have a set of patches $\mathscr{P} = \{\mathscr{P}_i\}$ centered on the SURF keypoints $p = \{p_i = (u_i, v_i)\}$, where p_i defines an image coordinates.

For the training phase of *xyzNet*, we need to label each training data with the corresponding 3D world coordinates, $P_i^w = (X_i^w, Y_i^w, Z_i^w)$. Otherwise, we can execute SfM once on all

the training dataset to carry out a mapping between keypoints and point cloud. Fortunately, we have RGB-D datasets. So we can use RGB-D images from the calibrated camera with their corresponding camera poses to define labels for the training phase. Note that in the testing phase, we only use RGB images and we do not need any depth information. From the position $p_i = (u_i, v_i)$ of each keypoint detected in RGB image and the corresponding depth value D_i in the depth image, a 3D position $P_i^c = (X_i^c, Y_i^c, Z_i^c)$ of the keypoint in camera coordinates system is calculated by using the standard pinhole camera model as follows:

$$P_i^c = D_i K^{-1} \begin{bmatrix} p_i \\ 1 \end{bmatrix} \tag{3.1}$$

Where K is a matrix of camera intrinsic parameters. The world coordinates $P_i^w = (X_i^w, Y_i^w, Z_i^w)$ corresponding pixel p_i are defined based on the transformation equation in the homogeneous coordinates system:

$$\begin{bmatrix} P_i^w \\ 1 \end{bmatrix} = \left[\begin{array}{c|c} R & t \\ 0 & 1 \end{array}\right] \begin{bmatrix} P_i^c \\ 1 \end{bmatrix} \tag{3.2}$$

Where the camera pose $T = [R|t]$ including rotation matrix R and translation vector t for each frame is supposed to be known in advance.

3.3.2 xyzNet for 3D world coordinates regression

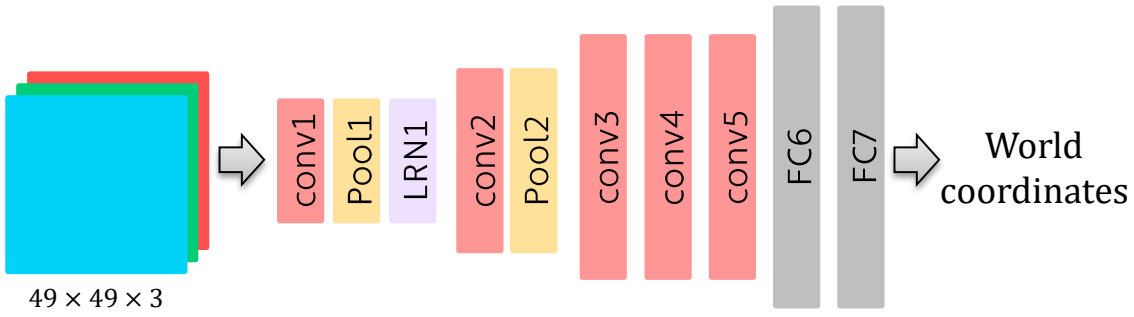


Fig. 3.6 xyzNet: A CNN regression for predicting world coordinates from RGB patches

We design a novel regression network to directly predict from 2D pixels the 3D correspondences in the world coordinate system. We called this network *xyzNet*. It takes RGB image patches with a fixed size of 49×49 pixels as inputs. The most common networks used in deep learning are not suitable for processing multiple patches in terms of computational time. Our xyzNet is a light CNN dedicated to efficient and robust camera relocalization.

Indeed, xyzNet consists of five layers that perform convolution of the input with a set of filters of 3×3 kernels, max-pooling and sub-sampling over a 3×3 area, and a rectified linear (ReLU) activation function. In the first stage, a local response normalization (LRN) is used to normalize patches with different illumination conditions. These five layers are followed by two fully connected layers to regress 3D world coordinates. A dropout layer is added after every fully connected layer to deal with over-fitting. *xyzNet* is illustrated in Figure 3.6. The use of patches based on keypoints instead of random points or grid-points reduces noise of training data as well as search space. This does not require a complex network and our light network, *xyzNet*, can perform efficiently and robustly.

In the training phase, the weights of *xyzNet* are learned by minimizing an Euclidean objective loss function with an optimization algorithm that is a stochastic gradient descent. The loss function is defined as follows:

$$l(p) = \sum_{p_i \in p} \|P_i^w - \hat{P_i^w}\|_2^2$$

Where P_i^w and $\hat{P_i^w}$ are respectively ground truth and prediction about the 3D coordinates of pixel p_i in the world coordinate system.

Contrary to [83] that predicts camera pose (6-DoF) from a whole image, in our system, each patch predicts a 3D continuous prediction about its own position. This reduces considerably the complexity of the loss function and produces more efficient optimization.

3.3.3 Camera pose calculation

Here, we will show how to estimate camera pose from predictions of xyzNet based on the geometric approach. From each RGB image, we extract a set of patches centered on keypoints. Each patch passes through xyzNet to generate a prediction about the 3D position in the world coordinates system. When all patches have passed through xyzNet, we obtain a set of 2D-3D point correspondences. As presented in Section 2.2.1, just three exact 2D-3D point correspondences are theoretically required to infer the camera pose. Nevertheless, a well-known computer vision method can solve this problem, namely Perspective-n-Points (PnP) algorithm. But as *xyzNet* can make some noisy predictions, we do not consider directly all 2D-3D point correspondences to calculate the camera pose. Instead, we first use PnP and Ransac to remove noise (outliers) and keep exact predictions (inliers) as in [99]. RANSAC generates a set of hypothetical poses $\mathscr{T} = \{T_i\}$ by performing PnP on random subsets of 2D-3D point correspondences. Each hypothetical pose defines a set of inliers based on the re-projection error. And the best hypothetical pose which provides the highest number of

inliers is defined as follows:

$$\max_{\forall T_i \in \mathscr{T}} \sum_{p_j \in p} \rho(\alpha_{ij}) \tag{3.3}$$

$$\rho(\alpha_{ij}) = \begin{cases} 1, & \text{if } \alpha_{ij} < \tau \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

Where $\alpha_{ij} = \|p_j - KT_i^{-1}\hat{P}_j^w\|^2$ and τ is the maximum threshold of re-projection error that defines inliers. Then pixel j is considered as an inlier of hypothesis T_i if $\rho(\alpha_{ij}) = 1$.

Let $\mathscr{I}$ be the set of indices of the inliers associated with the best hypothetical pose. The final camera pose is carried out by running PnP once on all the inliers of the set $\mathscr{I}$ to minimize the re-projection error function:

$$E(T) = \sum_{i \in \mathscr{I}} \|p_i - KT^{-1}\hat{P}_i^w\|^2 \tag{3.5}$$

Inferring camera pose from multiple patches with filtering method based on PnP and RANSAC algorithms allows to eliminate outliers on moving objects in the scene. It is to address the challenge of scenes with partial occlusion. Moreover, the number of inliers can be used as a confidence score of the final estimation which is not provided by the previous deep learning based methods. With this confidence score, we determine which frames can be used for the augmented reality. Our results are shown in the following paragraph.

3.3.4 Experiments

All experiments are implemented on a NVIDIA GTX 1080 GPU using Caffe framework [74]. For the training phase, we extract up to 500 patches with fixed size from every image. We experiment many different configurations on a scene of 7 scenes dataset [156] to optimize parameters that are used for other scenes. Our configuration includes: 500 epochs with a batch size of 2048; training begins at a learning rate of 10^{-2}; then the learning rate is dropped by multiplying it with a factor gamma of 0.8 after every 50 epochs; dropout probability is 0.5 for all fully connected layers; xyzNet is trained by using Stochastic Gradient Descent with a momentum of 0.9 and a weight decay of 10^{-5}.

We evaluate our method on 7 scenes dataset [156] and CoRBS dataset [179], presented in Section 2.6. Both datasets are indoor scenes. We show results about accuracy as well as computational time of our method. Our results are compared to state-of-the-art methods. Each one provides RGB-D images, intrinsic matrix of camera and annotations (camera pose for every frame).

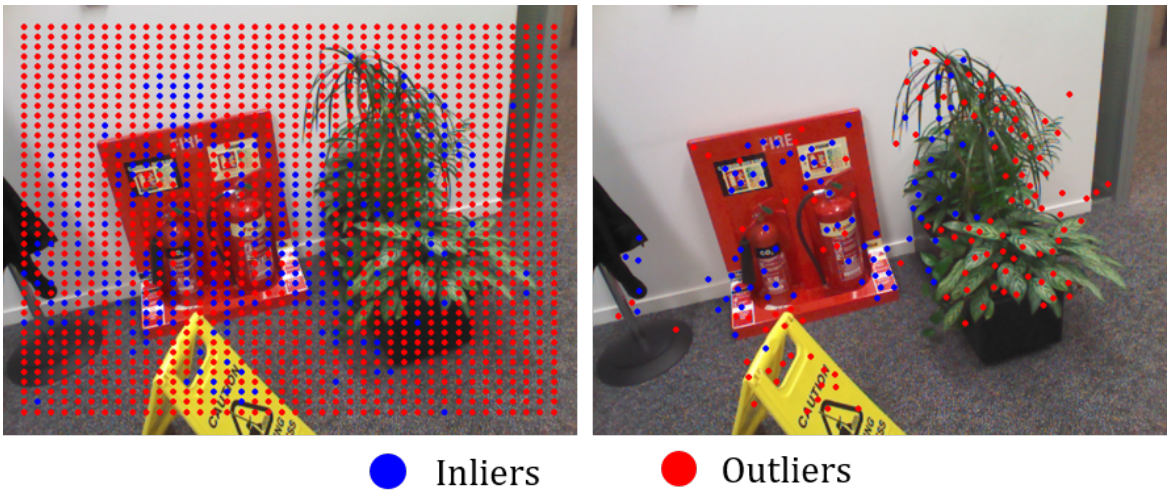


Fig. 3.7 Detected inliers from patches extraction based grid-points (left) as described in [20] and key-points (right) on an image of the fire scene.

Evaluation of patch extraction based on keypoints

To evaluate benefits of patch extraction based on keypoints, we compare it with a patch extraction based on a grid-points of size 40×40 as proposed in DSAC [20], illustrated in Figure 3.7.

Firstly, in the training phase, the use of patch extraction based on a grid of points generates a lot of training patches that includes much noisy data from homogeneous patches. Therefore, *xyzNet* training takes more time and converge with more difficulty. Figure 3.8 shows time of training convergence. Keypoint based method is two times faster than grid-point based method.

Secondly, in the testing phase, grid-point based method takes a lot of time to predict thousands patches including homogeneous patches that does not provide any information about translation and rotation of camera. So, processing these patches is redundant and even counterproductive. As shown in Figure 3.7, all patches extracted from wall or floor are outliers, since these patches do not contain enough individual feature to discriminate against each other. High accurate predictions (inliers) belong to textured patches.

In addition, when having too many predictions, PnP and Ransac algorithms need more time to define the best solution for camera pose. Figure 3.9 shows that the same computational time per image, key-point based is more accurate than grid-point based. Therefore, our method using patches extraction from keypoints can reduce run-time while achieving higher accuracy.

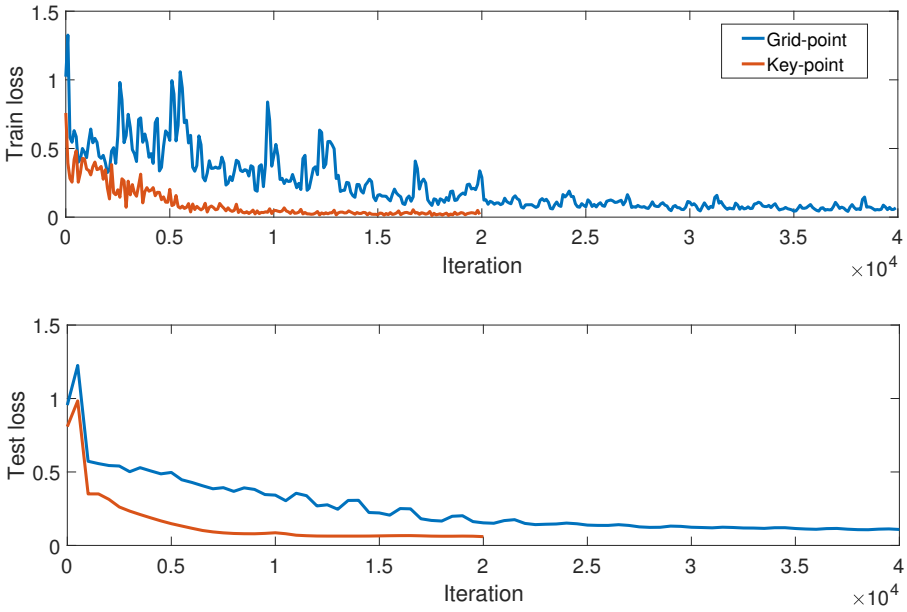


Fig. 3.8 Training performance from patches extracted from keypoints and grid-points on the chess scene.

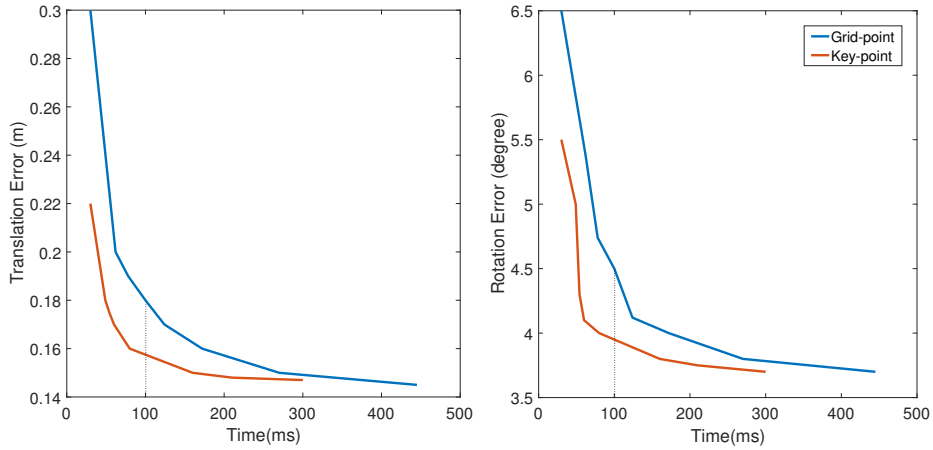


Fig. 3.9 Relation between computational time per image and mean accuracy of camera relocalization. It is obtained by changing number of Ransac iteration for key-point based and grid-point based methods on the chess scene.

Table 3.2 xyzNet's error: The mean distance error between predictions and ground truths, on the set of all predictions (Err_P) and on the set of inliers (Err_I).

Scene	Chess	Fire	Heads	Office	Pumpkin	RedKitchen	Stairs	Average
Err_P	0.25m	0.19m	0.14m	0.65m	0.27m	0.44m	0.34m	0.28m
Err_I	0.13m	0.11m	0.06m	0.26m	0.11m	0.14m	0.13m	0.12m

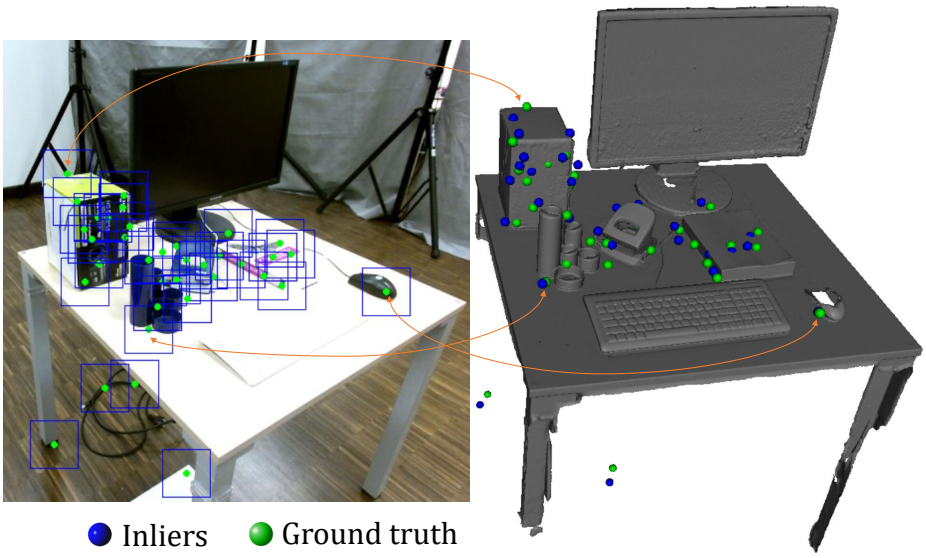


Fig. 3.10 xyzNet's accuracy: From a set of patches (blue squares) extracted around key-points (2D green points) for which correspond a ground truth (3D green points) defined in the world coordinates system, xyzNet predicts a set of world coordinates (3D blue points).

xyzNet accuracy

The accuracy of xyzNet prediction is the core of our pipeline and has a powerful influence on the accuracy of camera pose. We analyze the xyzNet's accuracy by computing the distance error between predictions and ground truth.

In table 3.2, we show the average location error on the testing data of 7 scenes dataset. We calculate two distance errors, one on all predictions and, one on all inliers. The results are $0.28m$ and $0.12m$ respectively. In Figure 3.10, we visualize an example on the *desk* scene of the CoRBS dataset concerning the predictions of the set of inliers. The results indicate that xyzNet provides a higher precision if efficiently filtered with the Ransac and PnP. For the scenes *office, red kitchen, stairs*, we did not achieve good results, what could be explained by the repetitiveness of the scenes. That makes patches extraction on similar object ambiguous.

However, filtering with Ransac and PnP algorithms greatly improves the accuracy of the estimation by decreasing the ambiguous predictions.

In terms of runtime and accuracy of camera relocalization

Table 3.3 Median pose errors. Comparison of our methods with the machine learning based state-of-the-art methods on 7 scenes dataset.

Scene	PoseNet [83]	Bayesian PoseNet [81]	PoseNet 2 [82]	LSTM-PoseNet [177]	**xyzNet**
Chess	0.32m, 8.12°	0.37m, 7.24°	0.13m, 4.48°	0.24m, 5.77°	**0.06m, 2.38°**
Fire	0.47m, 14.4°	0.43m, 13.7°	0.27m, 11.3°	0.34m, 11.9°	**0.06m, 2.16°**
Heads	0.29m, 12.0°	0.31m, 12.0°	0.17m, 13.0°	0.21m, 13.7°	**0.08m, 4.75°**
Office	0.48m, 7.68°	0.48m, 8.04°	**0.19m, 5.55°**	0.30m, 8.08°	0.28m, 6.61°
Pumpkin	0.47m, 8.42°	0.61m, 7.08°	0.26m, 4.75°	0.33m, 7.00°	**0.06m, 2.00°**
Red Kitchen	0.59m, 8.84°	0.58m, 7.54°	0.23m, 5.35°	0.37m, 8.83°	**0.06m, 2.25°**
Stairs	0.47m, 13.8°	0.48m, 13.1°	0.35m, 12.4°	0.40m, 13.7°	**0.20m, 4.42°**
Average	0.44m, 10.4°	0.47m, 9.81°	0.23m, 8.12°	0.31m, 9.85°	**0.11m, 3.51°**

Table 3.4 Comparison of our methods with the state-of-the-art methods on 7 scenes dataset by measuring the percentage of test images where the pose error is below 5*cm* and 5°.

Name of scene	Sparse Features[156]	Brachmann et al.[22]	DSAC[20]	**xyzNet**
Chess	70.7%	94.9%	**97.4%**	41.8%
Fire	49.9%	**73.5%**	71.6%	47.5%
Heads	**67.6%**	48.1%	67.0%	41.6%
Office	36.6%	53.2%	**59.4%**	15.6%
Pumpkin	21.3%	54.5%	**58.3%**	40.1%
Red Kitchen	29.8%	42.2%	42.7%	**42.9%**
Stairs	9.2%	20.1%	13.4%	**25.9%**
Average	40.7%	55.2%	**58.5%**	36.5%

In this paragraph, we evaluate our method on the 7 scene dataset to compare it with the state-of-the-art methods. We consider only RGB image based methods.

Baselines. Several strong methods of the three approaches (geometric, machine learning, hybrid) are used as the baselines. For machine learning based methods, we use the same baselines as in Section 3.2 (PoseNet [83], Bayesian PoseNet [81], PoseNet 2 [82], LSTM-PoseNet [177]). For geometric approach, we compare again to Active Search [151] and a

Table 3.5 Median poses errors of the complete 7 scenes dataset (17000 frames).

Method	Translation Error (cm)	Rotation Error (°)	Time(ms)
Active Search [151]	4.9	2.46	100
Brachmann et al. [22]	4.5	2.0	1000
DSAC [20]	**3.9**	**1.6**	1500
xyzNet	7.7	2.8	**60**

baseline in [156] that uses ORB feature based methods on only RGB image at test time. For hybrid approach, we take two recent methods [22, 20].

Computational time. We measure the time processing of our experiment. It takes about 60*ms* for each frame with 10*ms* for SURF feature detection, 25*ms* for world coordinate prediction of 500 patches on GPU and 25*ms* for 500 iterations of Ransac and PnP. Runtime depends on the number of iterations of Ransac to calculate camera pose. However, we fix the number of iteration at 500 being enough to balance between computational time and accuracy. The training time for each scene in 7 scenes dataset is about 4 hours.

Accuracy. To compare our method to the machine learning approach, we measure the median pose errors on 7 scenes dataset. The results are shown in Table 3.3. Our method clearly outperforms all the machine learning baselines in both translation and rotation error (except office scene). In Table 3.4, we compare our results on 7 scenes dataset with the geometric and hybrid methods. We use a metric based on the percentage of test images where the pose error is below 5*cm* and 5°. And we also show the median pose errors of all frames in the 7 scenes dataset with the runtime per frame in the Table 3.5. Our method is slightly higher than other methods on two scenes: *red kitchen* and *stairs*. For the other scenes, our method is not as good as the methods in [22], [20]. And the geometric based method in [156] outperforms our methods for scenes: *chess*, *fire* and *heads*. However, our method is able to relocalize camera for each frame in 60*ms*. Whereas, the sparse feature based method takes approximately 250*ms* per frame, [22] and [20] require more than a second for each frame, making difficult to use them for augmented reality systems requiring real-time processing. We obtain the worst result on *office* scene. As our analysis above, RANSAC can eliminate ambiguities on repetitive scenes such as *office, red kitchen* and *stairs*. Unfortunately, too many predictions are considered as outliers on the *office* scene, what results in too few 2D-3D inliers to achieve good results by the PnP.

Confidence score

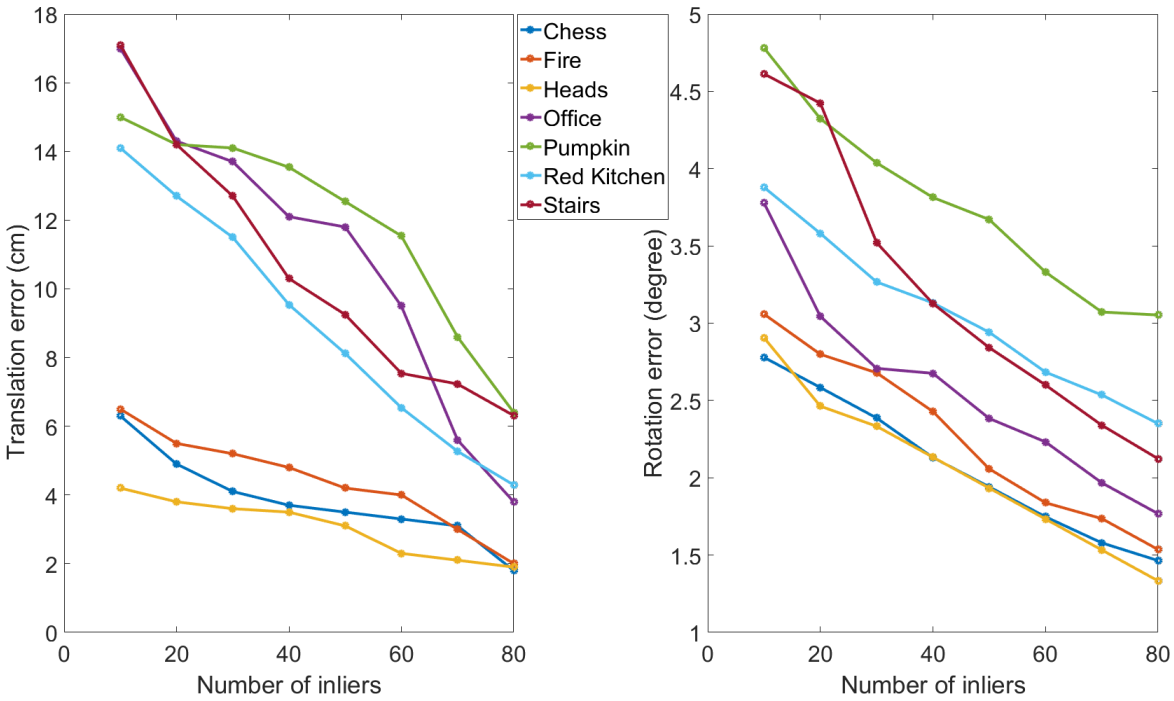


Fig. 3.11 The camera pose error according to number of inliers: For each scene, we calculate the median of the translation error (in the left) and rotation error (in the right) on the frames which have at least x inliers.

The confidence score of camera pose is an important issue in camera relocalization as well as in deep learning regression, which is not provided in the state-of-the-art methods. In our solution, no confidence score is given from xyzNet. However, we leverage the number of inliers to quantify the accuracy of our method. The number of inliers is not an absolute confidence score, but the accuracy is correlated to it. Figure 3.11 shows the increasing accuracy of our method according to the number of inliers. This allows us to determine which frames can be used for augmented reality applications.

Table 3.6 Mean of median poses errors on three scenes of CoRBS dataset.

Method	Translation Error (cm)	Rotation Error (°)
PoseNet	12.0	4.72
Kacete et al.	4.7	2.46
Ours	**3.5**	**0.97**

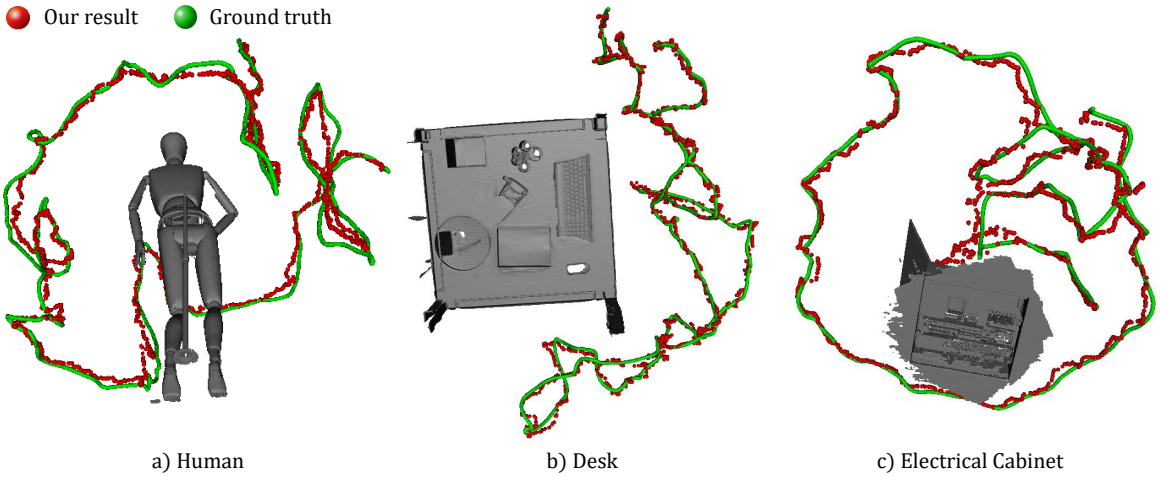


Fig. 3.12 Our results on the scenes of CoRBS dataset: our results by the red trajectories and the ground truth in the green

Performance on texture-less datasets

In this experiment, we evaluate our method on CoRBS dataset. With respect to the scale, the scenes in this dataset are simpler than those in 7 scenes dataset. CoRBS dataset contains scenes in desktop-scale, each of which focuses on a small environment such as around a desk or a cabinet. However, this dataset is challenging due to the presence of texture-less and many flat surfaces. We choose three sequences (each scene contains over 2000 images) corresponding to three scenes of *human, desk, electrical cabinet* for our experiment as performed in [79].

Figure 3.12 shows three resulting trajectories for the translation estimation. We evaluate our method with this dataset in comparison with PoseNet [83] and the method proposed in [79]. [79] uses a random forest to directly predict camera pose from each image patch, final pose being defined then by running mean-shift on all pose predictions. Only both two methods [79, 83] are experimented and achieved on this dataset. Table 3.6 shows that our method outperforms [83, 79] both on translation and rotation error. Moreover, in Figure 3.13, we compare the results of our work with those of [83] on electrical cabinet scene. Our results prove to be more stable.

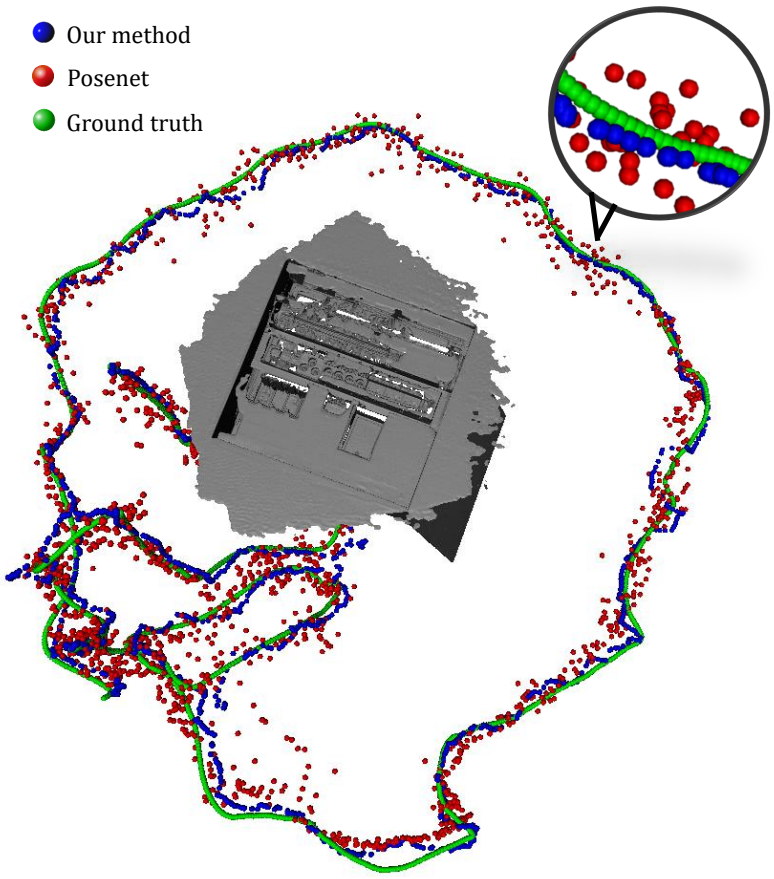
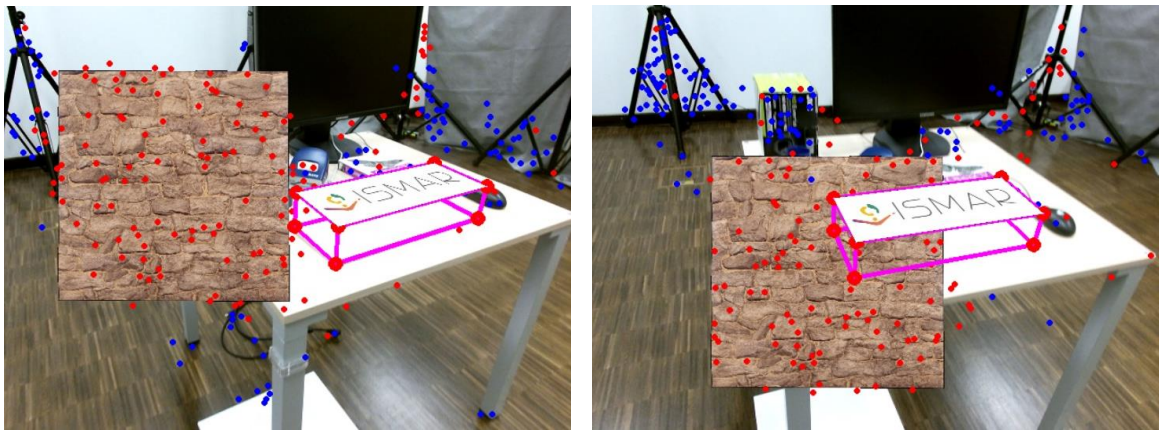


Fig. 3.13 Comparative result between our method (blue) and PoseNet (red) about accurate translation.

Robustness to occlusion



a) Frame #005 b) Frame #100

Fig. 3.14 Capacity of our method to process partial occlusion during a demonstration of an augmented reality application.

In this paragraph, we present an interesting advantage of our method that uses patches instead of a whole image. For camera relocalization, the handling of dynamic scenes is difficult. When objects move through a scene, it makes partial occlusion on the scene. In our method, we extract a set of patches to generate a set of predictions for 3D world coordinate, in which we take even patches of moving objects. For example in Figure 3.14, we extract patches on the black mask that is moving in the scene. However, the use of Ransac and PnP eliminates outliers on the moving object and retains correspondences on static objects. So camera pose is always stably estimated, which allows the method to be applied in the context of augmented reality as illustrated in Figure 3.14.

Conclusion

In this section, we proposed a hybrid method combining machine learning and geometric approach for real-time camera relocalization. We presented our light convolutional neural network to robustly define 2D-3D point correspondences. A set of probabilistic results is generated to address uncertainty of deep learning regression in camera relocalization. Simultaneously, we exploit geometric information about 2D-3D point correspondences to resolve the challenge of partial occlusion and calculate camera pose by using Ransac and PnP algorithms. Besides, we also consider the number of inliers as a confidence score for each frame.

Although, we obtain better results on most of the test scenes compared to the state-of-the-art methods that can process in real-time, our method faces difficulties on repetitive scenes. Indeed, our method removes almost ambiguity of unimodal prediction achieved by our network. Thus, we do not obtain enough inliers to compute camera pose. In the next section, we wish to improve the accuracy of the prediction by using multi-output world coordinate prediction.

3.4 Efficient multi-output world coordinate prediction

Our goal in this section is to propose a both real-time and accurate camera relocalization method using only RGB images. The main limitation of the state-of-the-art hybrid methods concerns the computational time of the geometric part of the process. This computational time is high, due to the excessive number of data used as inputs of this geometric part, and also their low accuracy. Therefore, we present in this section a data-oriented method, that focuses on improving the data provided by the machine learning part of the process. Our contributions consist of decreasing the number of data handled in each step of the machine learning part, and increasing their relevancy at the same time.

Our work in Section 3.3 improved significantly the accuracy of machine learning part by using our xyzNet to regress 3D world coordinate based on local patches. It selects patches based on sparse feature detection (SURF detection) instead of being random [156, 22]. It drastically decreases the number of data and increases their relevancy. However, each patch passes through xyzNet to achieve a 3D world coordinate prediction without a confidence score. Hence, for scenes with numerous repeated patterns, patches extracted from similar texture give ambiguous predictions. This reduces the accuracy of machine learning part.

To address the limitations of our previous work, we use a regression forest which provides multi-output probabilistic prediction. Concerning the regression forest, we propose a new split function used at each internal node of the regression forest, which takes the whole feature vector as input. This is different from the classical binary test function of the models in [156, 59, 176, 22]. These methods use features proposed by [97] that are based on the difference of intensities of two pixels taken in the neighborhood of the keypoint. Even if those features are fast to compute, they are less robust. Indeed, the intensity of pixel is powerfully influenced by illumination change. Therefore, with Hough voting, they are very efficient for some tasks such as object detection, when all patches vote for same objective and the mean-shift is used to rapidly filter and combine votes. Nevertheless, for camera relocalization, each patch votes for a 3D world coordinates separately. So it need more accurate prediction to fast define accurate camera pose from PnP and RANSAC (RANSAC stop early when it finds enough inliers). Our new split function enables the accuracy of 2D-3D correspondences to be improved, taking into account all the relevant information of the feature vector.

Another originality of our method concerns the feature extraction algorithm itself. We experiment two type of features: hand-crafted descriptor (SURF) and a learned descriptor extracted from xyzNet. Both features provide a discriminate ability. We evaluate and compare the effectiveness of both features.

Combining these originalities, our camera relocalization method allows a trade-off between computational time and accuracy. We will see in Subsection 3.4.3 that our method is as accurate as state-of-the-art method, and in addition perform relocalization in real-time.

In Subsection 3.4.1, we first present our learning based on a sparse feature regression forest and especially our new split function. Subsection 3.4.2 introduces our feature extraction methods. Subsection 3.4.3 shows our experiments and a conclusion.

3.4.1 Accurate sparse feature regression forest learning

The sparse feature methods in [149, 151] match features extracted from each image to a 3D scene model which includes a set of 3D points in world coordinates system attached with feature vectors. However, the matching time is expensive and depends on the size of the scene model. Our method uses a regression forest to robustly define 2D-3D point correspondences. Regression forest has been successfully achieved for object detection and object pose estimation [46, 22]. Recently, it has been used for camera relocalization as in [156, 59, 176, 22]. We propose two novelties in our regression forest implementation to improve the accuracy of correspondences. Firstly, we propose a novel split function at each node. This split function takes the whole feature vectors as inputs to optimally separate data. Secondly, we use only sparse feature extraction to train our regression forest. This reduces the training space. Hence, our regression forest improves the accuracy of predictions.

As introduced in [26], a regression forest is a set of N decision trees. Each tree consists of split nodes and leaf nodes. Each split node i is considered as a weak learner parameterized by $\theta_i = \{ref_i, \tau_i\}$ where ref_i is a reference feature and τ_i is a threshold.

For each decision tree, a feature vector f_j is considered as starting from the root node and descending to the leaf node by repeatedly evaluating weak learner at each node i:

$$h(f_j, \theta_i) = \begin{cases} 0, & \text{if } d(ref_i, f_j) < \tau_i, \text{ go to left child node} \\ 1, & \text{if } d(ref_i, f_j) \geqslant \tau_i, \text{ go to right child node} \end{cases} \tag{3.6}$$

Where $d(ref_i, f_j) = \|ref_i - f_j\|_2^2$ is an euclidean distance function. In the followings, we introduce how we implement the regression forest training and the regression forest prediction to estimate camera pose.

Regression forest training

During the training step, a set of feature vectors is extracted from a set of training images. For each tree, a subset of features S is randomly chosen. The weak learner parameters θ_i at

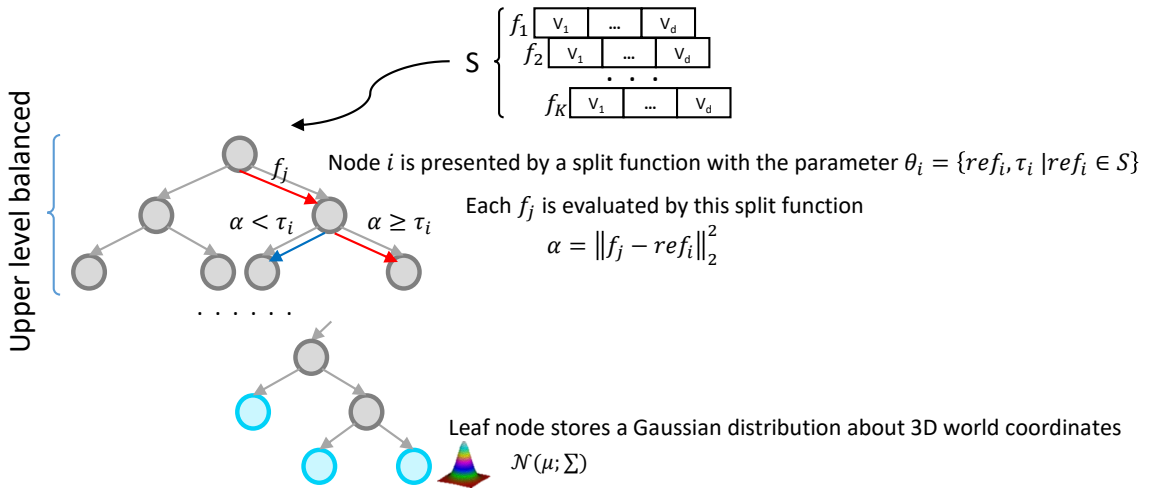


Fig. 3.15 Sparse feature regression forest training.

each split node i is used to split the set of features S_i into left child node S_i^L and right child node S_i^R. The weak learner θ_i is trained to minimize an objective function $Q(S_i, \theta_i)$ aiming to reduce spatial variance as shown in [156]:

$$Q(S_i, \theta_i) = \sum_{d \in \mathscr{L}, \mathscr{R}} \frac{|S_i^d(\theta_i)|}{|S_i|} V(S_i^d(\theta_i)) \tag{3.7}$$

$$\text{with } V(S) = \frac{1}{|S|} \sum_{m \in S} \|m - \overline{m}\|_2^2 \tag{3.8}$$

Where m is world coordinates of each feature, $\overline{m}$ is the mean of world coordinates in S. Figure 3.15 illustrates our sparse regression forest training.

However, in case of unbalanced tree, it requires to increase tree depth in order to separate data into different clusters. That leads to increase testing time. Therefore, to reduce processing time for both training and testing as well as improve the accuracy of predictions, we modify the algorithm in the uppest levels of trees according to [118]: weak learner parameters are learned so as to equally divide samples S_i at node i into its two children by finding a reference feature ref_i and choosing a threshold τ_i.

$$\theta_i = \left\{ ref_i, \tau_i \,\middle|\, |S_i^L| = |S_i^R| \right\} \tag{3.9}$$

The training tree terminates when a maximum depth D_{max} is reached or the set S_i has a few data. Each leaf node stores the 3D positions of data, which is presented by a Gaussian distribution $\mathcal{N}(m, \bar{m}, \Sigma_m)$ where m is the set of 3D world coordinates corresponding to

keypoints. $\bar{m}$ and Σ_m are the mean and the covariance of the distribution respectively. Our regression forest clusters 3D positions of keypoints into a set of leaves.

Regression forest prediction to estimate camera pose

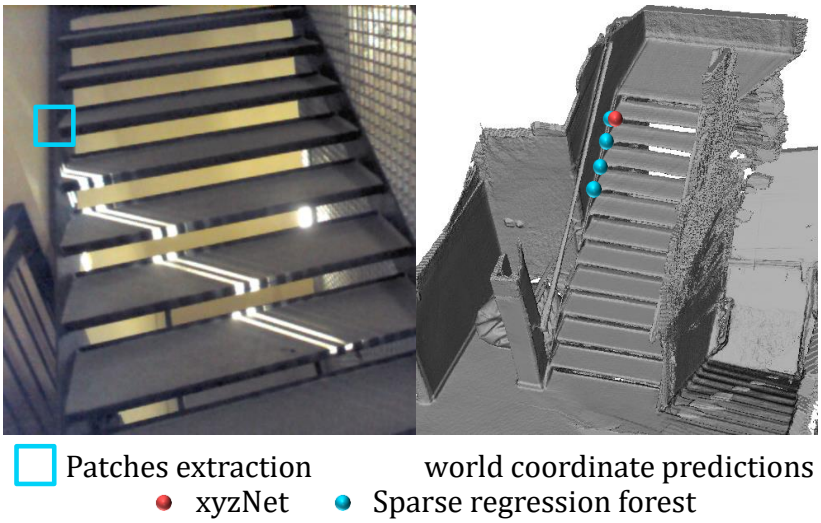


Fig. 3.16 Example of 3D world coordinates predictions on a repetitive scene (stairs scene): Multi-output predictions using sparse regression forest (blue points) versus only one uncertain prediction using xyzNet (red point).

For scenes containing repetitive patterns, it is difficult to define exactly the world coordinates of patches with similar appearances. Two patches can look very similar, whereas they come from two different parts of the scene. Our method gives a set of possible 3D world coordinates for each patch to address this ambiguity. In the testing phase, each feature extracted from a RGB image passes through a regression forest to obtain a set of predictions as shown in Figure 3.16. Moreover, the covariance of each leaf provides a confidence score for each prediction. This parameter helps us to select predictions being more reliable to compute camera pose. This exploits uncertainty of prediction in the deep learning method [83].

From each image, a set of features from SURF keypoints detection is extracted. When all patches have passed through our regression forest, we obtain a set of 2D-3D point correspondences. Each feature obtains multiple world coordinate predictions (one to many) instead of only one prediction from xyzNet in Section 3.3 (one to one). A common post-processing is to combine predictions from the same patch to make a final prediction. We instead keep all predictions to estimate camera pose. This helps handle the ambiguity of

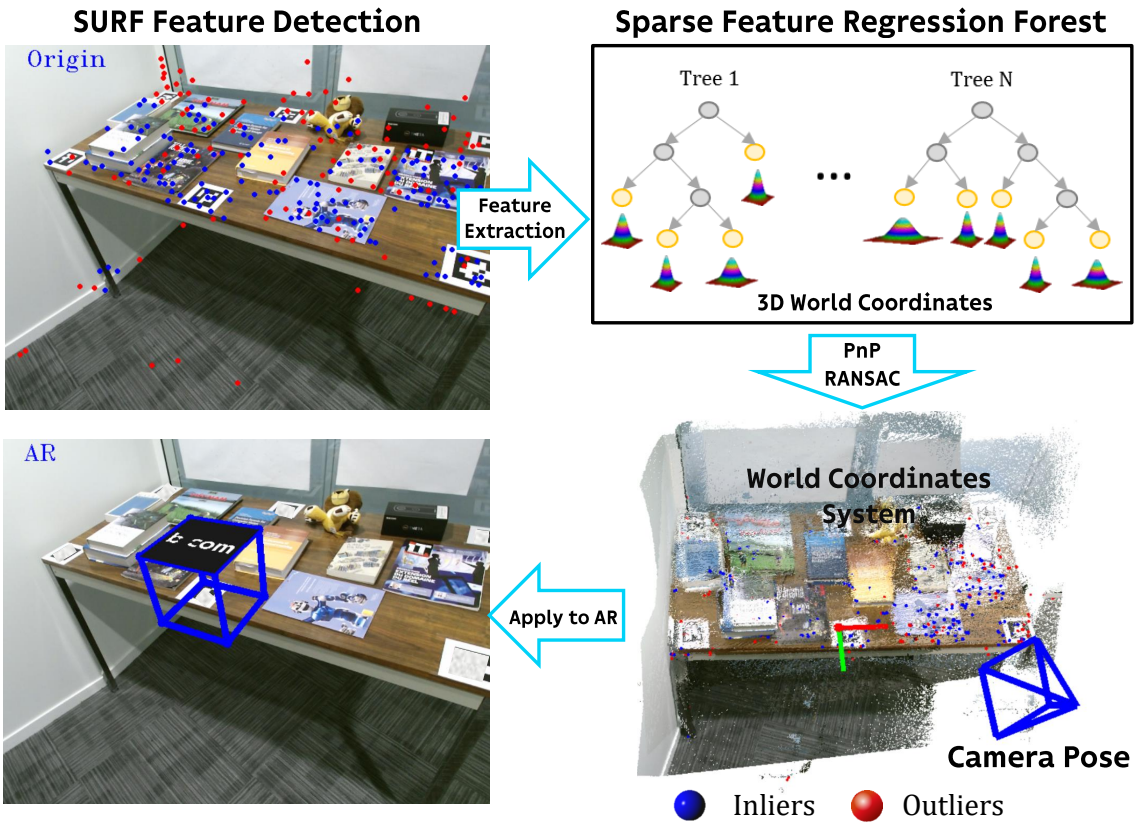


Fig. 3.17 Our accurate and real-time camera relocalization in AR.

scenes with repeated patterns. Then, similar to 3.3.3, we use PnP and RANSAC algorithms to remove noise (outliers) and keep relevant predictions (inliers) to compute camera pose. Figure 3.17 shows the pipeline of our sparse regression forest for camera relocalization in AR applications.

3.4.2 Hand-crafted descriptor versus learned descriptor

In this paragraph, we consider two types of descriptors for keypoints extraction from the SURF detection: hand-crafted descriptor and learned descriptor. And we compare the accuracy as well as the computational time of our camera relocalization method based on these descriptors in Subsection 3.4.3.

Hand-crafted descriptor

Known as an efficient and fast hand-crafted descriptor, we select SURF descriptor [12] which is invariant to scale and rotation. It is extremely interesting because this descriptor has

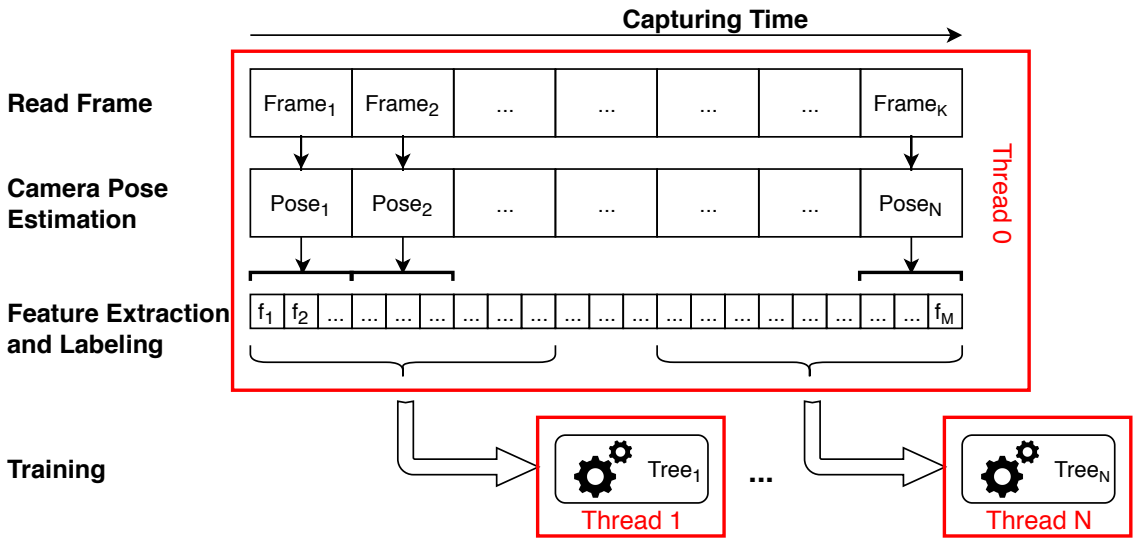


Fig. 3.18 Our online training system. Our sparse feature regression forest is learned by multi-threads in the same time of capturing data.

repeatability, stability that enhances the capability of regressing 3D world coordinates. We take whole SURF descriptors as input of our regression forest. We called *SURF-RF*.

This descriptor allows us to perform online learning for our sparse regression forest during images capturing. The online learning is integrated in tracking systems (e.g. SLAM, marker based,...) which provide camera pose for each frame. To label 3D world coordinates for detected keypoints, we use a RGB-D camera. From each captured RGB-D frame with estimated camera pose, we extract a set of SURF descriptors. Each descriptor is labeled by projecting 2D image coordinates in the world coordinates system. Once our system reaches sufficient amount of feature, we create a new thread to train a tree of our regression forest in the same time of capturing image. Our training step finishes when all trees are learned. Capturing and training phase takes about 2 minutes for a desktop-scale scene with the configuration of our sparse feature regression forest including: extraction of up to 500 SURF features for each image; 4 trees in the forest; 16 depth of each tree. Figure 3.18 shows the online learning of our system.

Learned descriptor

We use a learned descriptor from our dedicated convolutional neural network, xyzNet. Our learned descirptor is more robust to viewpoint change compared to hand-crafted descriptor. This improves the accuracy of locating 3D world coordinates from different view points. And it allows our method to address generalization challenge in camera relocalization.

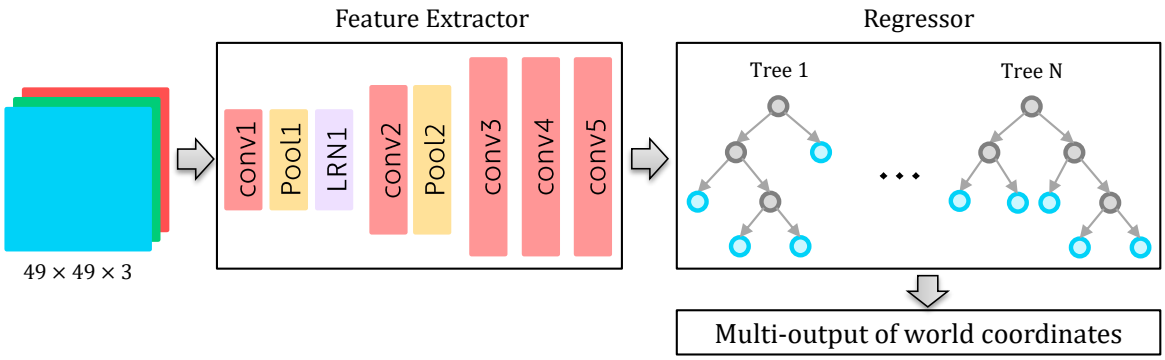


Fig. 3.19 Our *Deep-RF* for 2D-3D correspondences. The feature extraction is performed by xyzNet and the multi-output 3D world coordinates regression is performed by our regression forest.

Our learned descriptor is performed based on xyzNet which is described in Subsection 3.3.2. From each detected SURF keypoint, it takes RGB image patches with a fixed size of 49×49 pixels as inputs. We adapt xyzNet model without its fully connected layers. Thus the descriptor is provided as the output of the final convolution layer. Note that the two final fully connected layers are just used to train the xyzNet. But only feature descriptor layers are used in our sparse feature regression forest.

Then, we combine xyzNet and our regression forest to efficiently predict multi-output 3D world coordinates from each patch. That exploits uncertainty of deep learning regression. This combination, illustrated in Figure 3.19, is the heart of our method, that we called *Deep-RF*.

3.4.3 Experiments

Our method is implemented on an Intel Core i7 @2.9GHz, except for xyzNet feature descriptor, which is calculated on a NVIDIA GTX 1080 GPU. The configuration of our sparse feature regression forest includes: extraction of up to 500 SURF features for each image; 8 trees in the forest; the depth of each tree is 15 with balance samples for the 8 uppest level of each tree; 10^5 training features per tree; minimum 10 features at each node. For the testing, the maximum number of iterations of PnP-RANSAC is 500, and it stops as soon as 50% of correspondences are considered as inliers. The training parameters of xyzNet are setup as in Subsection 3.3.4. We evaluate our method on 7 scenes dataset, CoRBS dataset and BCOM dataset presented in Section 2.6. All datasets are indoor scenes. Each one provides thousands of RGB-D images, intrinsic matrix of camera and annotations (camera pose for each frame).

Accuracy of 3D world coordinates regression

Table 3.7 Accuracy of 3D world coordinates prediction. Location error (in centimeters) computed by the mean of distance error between ground truths and predictions on the set of inliers. Comparison between the sparse feature regression forest using the hand-crafted descriptor (SURF), the learned descriptor (Deep) and xyzNet. 2-elements: use only two elements of feature vector for the split function. whole: use the whole feature vector.

Scene		**Chess**	**Fire**	**Heads**	**Office**	**Pumpkin**	**Kitchen**	**Stairs**	**Average**
xyzNet		13.4	10.8	6.2	25.6	11.2	13.7	13.3	13.4
SURF	2-elements	13.7	15.7	10.2	19.6	14.3	17.7	17.4	15.7
	whole	5.8	8.3	4.3	15.2	6.5	11.5	11.2	9.0
Deep	2-elements	10.8	10.2	6.8	18.6	14.2	14.9	12.3	12.5
	whole	**5.5**	**7.3**	**3.6**	**14.6**	**6.1**	**9.7**	**10.4**	**8.2**

Table 3.8 Inference computational time per frame for each part of our process in milliseconds (ms). Comparison between the sparse feature regression forest using the hand-crafted descriptor (SURF), the learned descriptor (Deep) and xyzNet. 2-elements: use only two elements of feature vector for the split function. whole: use the whole feature vector.

Computational Time		**Keypoint Detection**	**Feature Extraction**	**Regression Forest**	**Geometric**	**Total**
xyzNet		10	20	0	30	60
SURF	2-elements	10	5	5	30	50
	whole	10	5	15	20	50
Deep	2-elements	10	20	5	30	65
	whole	10	20	20	20	70

We evaluate the accuracy of 2D-3D point correspondences by computing the distance error between 3D position predictions and ground truth on the set of inliers.

In Table 3.7, we show an average of location error on the testing data of 7 scenes dataset. We compare the accuracy of prediction performed by our sparse feature regression forest using hand-crafted descriptor and learned descriptor by two different strategies. The first one is xyzNet prediction (**xyzNet**). And The second one is a modification of our method using the substraction of two elements of the feature vector which is used in [156] (**2-elements**). The results indicate that our method improves significantly the accuracy of 3D world coordinate regression compared to xyzNet. Our split function using a whole feature vector is more accurate for the 3D world coordinate regression task than the split function using only two elements of the feature vector. Our method based on the learned feature (*Deep*) is slightly better than the one based on the SURF feature. The *office, red kitchen, stairs* scenes are extremely challenging because they contain repeated patterns and similar

structures. Nevertheless, our results on these scenes remain relatively correct and much more accurate compared to **xyzNet**. This can be explained by our regression forest which predicts multi-output possibilities about 3D world coordinates of repetitive pattern as illustrated in Figure 3.16. From geometric knowledge, RANSAC and PnP algorithms then filter outliers and retain inliers to calculate camera pose.

Regarding computational time, our split function takes more time to predict 3D world coordinates as it computes the distance on the whole feature. Table 3.8 shows that our method (**whole**) requires more time for random forest regression than **2-element**. However, the computational time related to the regression forest is low compared to the overall process. Besides, our method provides more accurate predictions, which makes the computational time of geometric part decreases (it can stop earlier when enough inliers have been obtained). Our method based the SURF feature is faster than the other methods and does not require a training step. Therefore, this allows us to perform an online learning for camera relocalization.

To summarize, our regression forest achieves fast 3D points location with high accuracy. It allows our method to process in real-time. In the next paragraph, we compare our method to the state-of-the-art methods in terms of accuracy and computational time of camera relocalization.

Accuracy and computational time of camera relocalization

In this paragraph, we present and discuss the results of our method on the 7 scenes dataset. We compare it with the state-of-the-art methods and our previous work xyzNet.

Baselines. We compare our method to all four approaches (geometric, machine learning, hybrid, image retrieval) for camera relocalization. Geometric method baselines are [151] (Active search) and a baseline in [156]. We also compare with an end-to-end camera pose regression PoseNet 2 [82] and accurate hybrid methods [22, 20]. Please refer to Subsection 3.3.4 for more details of these above. The last is Bag of words (BoW) relocalization belonging to the image retrieval approach. This method is a relocalization component in ORB-SLAM method [124]. A bag of words (BoW) is used to find keyframes that are most consistent with the current frame. After that, camera pose is defined by matching features of the current frame to the nearest keyframes. We perform ORB-SLAM relocalization by considering all training data as keyframes and using the pre-trained BoW of ORB features in [124].

Accuracy and computational time. To compare our method to the recent machine learning [82] and geometric [151] methods, we measure the average of all median pose errors on 7 scenes dataset. Table 3.9 shows that both our methods clearly outperforms all two

Table 3.9 Comparison of our method with geometric approach and deep learning approach in terms of accuracy and computational time. The accuracy is measured by averaging all median pose errors over all 7 scenes dataset.

Method	Translation Error (cm)	Rotation Error (°)	Time(ms)
Active search [151]	4.9	2.46	100
PoseNet 2 [82]	23.1	8.12	**5**
SURF-RF	4.2	1.7	**50**
Deep-RF	**3.8**	**1.51**	70

Table 3.10 Comparison of our method with the hybrid methods in terms of accuracy and computational time. The accuracy is measured by the median poses error of the complete 7 scenes dataset (17000 frames). The computational time is measured for one frame in milliseconds.

Method	Translation Error (cm)	Rotation Error (°)	Time(ms)
Brachmann et al. [22]	4.5	2.0	1000
DSAC [20]	3.9	1.6	1500
SURF-RF	3.9	1.7	**50**
Deep-RF	**3.5**	**1.4**	70

baselines in both translation and rotation error. Regarding the two types of descriptors for our regression forest, the learned descriptor (Deep-RF) is slightly more accurate than the hand-crafted descriptor (SURF-RF). On the contrary, in terms of runtime, SURF-RF is faster than Deep-RF. The sparse feature method [151] using SIFT feature is less efficient than our methods in terms of both accuracy and computational time. This method requires over 100*ms* per frame and the computational time increases with the scene size. Although PoseNet 2 [82] is very fast to relocalize camera with 5*ms* in the testing, its results are still moderately accurate. Indeed, our methods are six times more accurate than theirs. Regarding the testing time, our methods still perform in real-time, even if it is more time-consuming than PoseNet 2 [82]. Note also that the training of our sparse feature regression forest is faster than PoseNet 2 [82]: SURF-RF takes less than two minutes for training one tree on a CPU; Deep-RF requires about four hours on a GPU; the training of PoseNet 2 [82] takes from four hours to a day on a GPU Titan X.

In Table 3.10, we also compare our method on 7 scenes datasets to the hybrid methods [22, 20] which currently obtain high accuracy for camera relocalization from RGB images. Table 3.10 shows that our method (Deep-RF) is slightly more accurate than [22, 20]. Moreover, our methods greatly improves the hybrid approaches in terms of computational time. Our method

Table 3.11 Accuracy of camera relocalization. Comparison of our various contributions with the state-of-the-art methods on 7 scenes dataset by measuring the percentage of test images where the pose error is below 5*cm* and 5°. xyzNet: mono output of 3D world coordinates prediction.

Scene	[156]	[22]	DSAC[20]	BOW [124]	xyzNet	**SURF-RF**	**Deep-RF**
Chess	70.7%	94.9%	**97.4%**	60.0%	41.8%	73.1%	72.5%
Fire	49.9%	73.5**%**	71.6%	40.8%	47.5%	79.5%	**82.0%**
Heads	67.6%	48.1%	67.0%	56.0%	41.6%	**80.1%**	71.6%
Office	36.6%	53.2%	59.4%	45.8%	15.6%	54.5%	**59.5%**
Pumpkin	21.3%	54.5%	**58.3%**	32.3%	40.1%	55.1%	51.6%
Red Kitchen	29.8%	42.2%	42.7%	35.1%	42.9%	52.2%	**75.2%**
Stairs	9.2%	20.1%	13.4%	35.9%	25.9%	41.0%	**45.3%**
Average	40.7%	55.2%	58.5%	43.7%	36.5%	62.2%	**65.4%**

is able to relocalize camera for each frame at $50-70ms$, whereas [22] and [20] require more than a second per frame, because the camera pose optimization requires much time to process thousands input data. Unlike [22, 20], our method allows a real-time relocation that is generally necessary for augmented reality applications.

Table 3.11 shows the results of our methods, hybrid methods [22, 20], the sparse feature baseline in [156], BoW relocalization and the our previous work xyzNet in Section 3.3. The results are evaluated by using another metric: the percentage of test images where the pose error is below 5*cm* and 5°. This metric is suitable for evaluating camera relocalization methods addressing augmented reality systems. Regarding the result on each scene, our method performs better on five scenes: *Fire*, *Heads*, *Office*, *Red Kitchen* and *Stairs*. Table 3.11 also indicates that our method which gives multi-output predictions improves considerably the accuracy of mono prediction of xyzNet. Especially, we achieve approximately two times as accuracy as xyzNet on the scenes with repeated patterns such as *Office*, *Red kitchen* and *Stairs*.

Transfer learning

SURF-RF allows us to obtain a real-time camera relocalization for both training and testing, whereas, Deep-RF is not real-time for training. Because it consists of two training phases: xyzNet and the regression forest. Our regression forest can be fastly learned in less than 1 minute per tree on a CPU, whereas our xyzNet takes about four hours on a GPU. To reduce the training time of our deep-forest, we perform transfer learning as follows. First, we pre-train our xyzNet on each scene of 7 scenes. Next, we use this xyzNet model to extract descriptors on the other scenes and we just re-train our regression forest. Table 3.12 shows

Table 3.12 Results of transfer learning of xyzNet on 7 scenes dataset. xyzNet feature extraction is learned from each scene. Then it is used to evaluate on all scenes (translation error (cm) / rotation error (°))

Scene	**Scene for xyzNet model learning**						
	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
Chess	**3 / 1.40**	4 / 1.67	4 / 1.69	5 / 1.98	4 / 1.75	4 / 1.64	7 / 2.84
Fire	6 / 1.61	**2 / 1.16**	4 / 1.61	5 / 1.62	4 / 1.84	5 / 1.73	8 / 2.13
Heads	5 / 2.72	5 / 2.60	**3 / 1.79**	6 / 2.39	5 / 2.12	6 / 2.71	7 / 3.12
Office	11 / 3.37	11 / 2.87	9 / 2.97	**5 / 1.96**	8 / 2.87	12 / 3.47	18 / 4.55
Pumpkin	7 / 1.95	7 / 1.92	6 / 2.05	7 / 2.06	**5 / 1.60**	6 / 1.97	9 / 2.15
Kitchen	7 / 2.47	7 / 2.34	8 / 2.53	8 / 2.59	6 / 2.26	**3 / 1.15**	9 / 2.91
Stairs	11 / 3.18	8 / 2.51	7 / 1.69	9 / 2.00	7 / 2.11	11 / 3.40	**6 / 1.48**

that our results are still accurate even if the xyzNet model is learned on texture-less scene such as *stairs*.

Performance on texture-less dataset

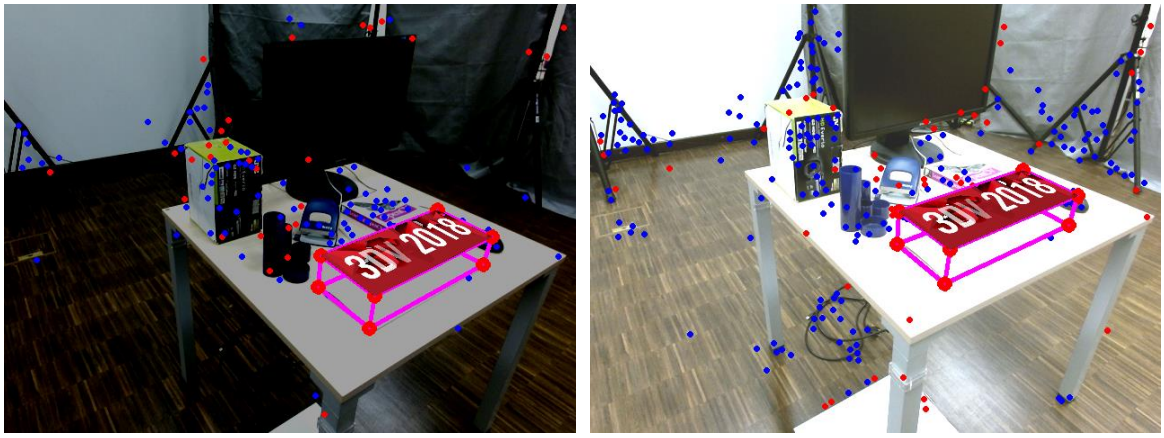
Table 3.13 Performance on texture-less dataset. Comparison between our methods and baseline machine learning methods on CoRBS dataset. The accuracy is measured by averaging all camera pose errors over all three scenes.

Method	Translation Error (cm)	Rotation Error (°)
PoseNet [83]	12.0	4.72
Kacete et al. [79]	4.7	2.46
xyzNet	3.5	0.97
SURF-RF	**1.9**	**0.65**
Deep-RF	**1.3**	**0.56**

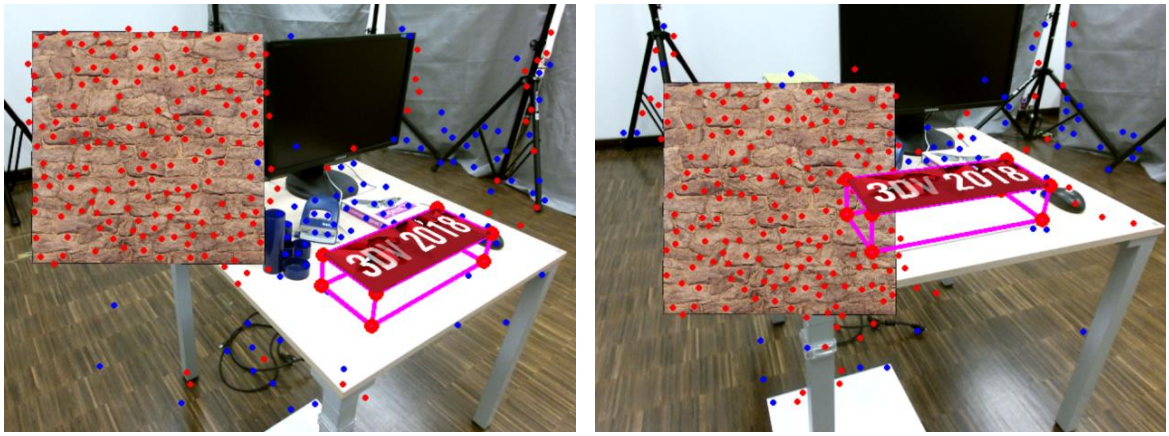
In this paragraph, we evaluate our methods on CoRBS dataset [179] which is challenging due to the presence of texture-less and flat surfaces. We compare our method to baselines [79, 83] presented in Subsection 3.3.4 and to our previous work xyzNet (Section 3.3). Table 3.13 shows the average of all camera poses errors over all three scenes. Both our methods outperform [83, 79] and xyzNet by a large margin whether on translation or rotation error for all three scenes.

Robustness to dynamic scenes

The dynamic environments are still challenges to camera relocalization. To evaluate the robustness of our method to dynamic scenes with partial occlusion and illumination changes,



(a) Illumination changes



(b) Partial occlusion

Fig. 3.20 Some examples of dynamic scene in augmented reality. Our method is robust to illumination changes and partial occlusion.

we randomly synthesize illumination changes (Figure 3.20-a) and a partial occlusion covering about 25% surface of each image (Figure 3.20-b) on CoRBS dataset. The mean of our results for the two challenges are respectively $3.8cm, 0.84^\circ$ and $4.0cm, 0.87^\circ$. Our methods are able to address illumination changes thanks to the invariant SURF feature. For partial occlusion challenge, our method uses of a set of sparse feature instead of a whole image in order to generate a set of 3D world coordinates. We take even key-points of moving objects (a textured mask in Figure 3.20-b). However, Ransac and PnP algorithms eliminate outliers (red points) on the moving object and retains inliers (blue points) on rigid objects. So camera pose is always stably estimated, which allows the method to be applied in the context of augmented reality as illustrated in Figure 3.20.

Viewpoint generalization performance

Table 3.14 Generalization performance on the different trajectories of the BCOM dataset. Our sparse feature regression forest is trained on each trajectory, then it is evaluated on the other trajectories. The results are shown with the following format: *translation error (cm) / rotation error (°)*. Gray rows and white rows are results for Deep-RF and SURF-RF respectively.

Test \ Train	T_1	T_2	T_3	T_4
T_1	2.3 / 1.61	3.6 / 1.70	3.1 / 1.79	2.7 / 1.92
	3.1 / 1.72	3.5 / 1.92	4.8 / 2.75	3.4 / 3.41
T_2	2.1 / 1.47	1.8 / 1.12	2.5 / 1.46	2.9 / 2.00
	4.4 / 2.35	2.6 / 1.63	3.5 / 2.36	4.1 / 3.05
T_3	2.9 / 1.85	3.1 / 1.52	1.4 / 1.12	3.2 / 1.41
	3.8 / 2.80	4.2 / 3.84	2.7 / 1.86	4.4 / 2.96
T_4	3.4 / 1.75	2.6 / 1.34	2.7 / 1.48	1.5 / 0.97
	4.5 / 3.65	4.9 / 4.68	5.2 / 3.62	2.5 / 2.38

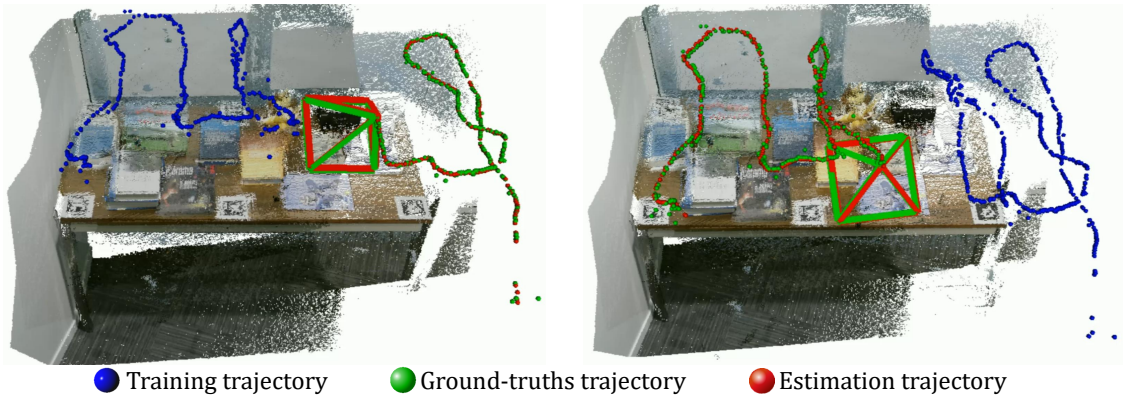


Fig. 3.21 Visualization of generalization performance on two sequences T_1 and T_2 of the BCOM dataset.

We evaluate generalization capability of our method on BCOM dataset, which contains four absolutely different trajectories. The trajectories are shown in figure 2.15. We choose respectively each trajectory to learn our sparse feature regression forest model and we then use this model to estimate camera pose for each frame of the other three trajectories. Obviously, Table 3.14 shows that the best results are obtained when the testing trajectory is the same as the learning trajectory. On the other trajectories, the results are slightly worse but remain correct (lower 5cm/5°), which shows the generalization performance of our method.

Table 3.14 also indicates that Deep-RF is more accurate than SURF-RF on all sequences. Although SURF feature is invariant to scale and rotation, our learned feature has higher accuracy for viewpoint changes. Figure 3.21 visualizes the generalization performance of our methods for camera relocalization on different trajectories.

Conclusion

In this section, we have presented a novel hybrid method merging machine learning approach and geometric approach for accurate and real-time camera relocalization from a single RGB image. We proposed a new split function using whole feature extraction at each node of the regression forest to efficiently define 2D-3D point correspondences. We compared two different feature types in our regression forest: hand-crafted feature and learned feature. Moreover, we show the favorable results of our method in comparison with the state-of-the-art methods. Finally, we demonstrate that our method is able to address dynamic scenes as well as viewpoint generalization.

3.5 Conclusion

In this chapter, we have proposed methods in order to address limitations of state-of-the-art methods addressing camera relocalization from a single RGB image both in terms of accuracy and computational time (including processing time and training time).

We first proposed the MIMO strategy based local patches to overcome the uncertainty of end-to-end machine learning camera regression. Then, we present the evolution of our hybrid methods that balances both accuracy and computational time of the camera relocalization. Our hybrid method is based on both machine learning approach and geometric approach and aims at benefiting from each. We present our xyzNet and our sparse feature regression forest that focus on improving both runtime and accuracy by inferring 2D-3D point correspondences. Our methods show high accuracy for camera relocalization. The last method based on a sparse feature regression forest addresses some challenges of camera relocalization: accuracy, computational time (both training time and runtime), dynamic scenes with partial occlusion, illumination change.

However, all our methods presented in this chapter learn a regression model from a static scene where the 3D physical model is always preserved without moving objects. Thus, when some objects move, the learned model is no longer accurate, and the camera relocalization can no longer be successfully performed. In the next chapter, we present a novel adaptive machine learning to tackle this challenge.

Chapter 4

Camera Relocalization in Dynamic Environment

Contents

4.1 Introduction

Our methods presented in Chapter 3 handled dynamic scenes with occlusion and illumination changes. In this chapter, we focus more particularly on the complicated challenge of dynamic scenes when individual objects move over time. Because our previous works use the regression model that is learned on a rigid scene, it cannot perform if a large part or whole scene changes.

Dynamic data is a common challenge for machine learning not only for camera relocalization. In machine learning, a model is learned from a set of training data. If data is changing and the learned model is not able to update itself, this model is no longer accurate. So these methods require an adaptive learning model which has the capacity to update rapidly to changes of data in order to maintain the accuracy of the learned model.

In camera localization community, dynamic environment with moving objects is always known as a difficult challenge. This challenge appears in real scenarios as augmented reality based maintenance, assembly and quality control tasks for Manufacturing and Construction application where: devices are equipped with a single RGB camera, scenes are changing gradually over time, tasks requires workbench scale 3D registration. The most typical methods to address this challenge are RANSAC algorithms. RANSAC considers data on moving objects as outliers in order to eliminate them in camera pose estimation process. However, if a large number of objects moves, the number of outliers will be superior to the number of inliers. In this case, RANSAC algorithm processes incorrectly. In recent years, several SLAM methods focus on this specific issue of dynamic scenes. Most works perform detection and segmentation of moving objects in each frame. This is to avoid their influence on the camera pose estimation. [178, 142, 15, 193] detect and track the known dynamic objects or movable objects e.g. people, vehicles. [165, 4] detect moving objects by using optical flow between consecutive frames. Nevertheless, these methods try to remove moving objects to compute camera pose. So they cannot process in scenes where the whole scene changes gradually. To overcome this, [129] proposed a real-time dense dynamic scene reconstruction and tracking based on depth images. It uses a single rigid volumetric TSDF (Truncated Signed Distance Function). However, this method requires a RGB-D camera. [169] proposes another SLAM approach that can detect any changes by projecting the map feature into the current frame. In real-time, it can update the 3D points cloud and keyframes stored in a memory to add new elements or remove those which do not exist anymore. This allows to address the challenge of whole scene changing gradually. However, it has the limitations due to the fact that matching cost increases with respect to the number of keypoints. Moreover, matching local features is noisy and unreliable on scenes with repeated patterns.

Our main contribution is to propose a machine learning algorithm based on a regression forest process, that adapts itself in real time to a predictive model. It evolves by part over time without having to re-train the whole model from scratch.

The process is based on a regression forest ; and we call it **Adaptive Regression Forest - ARF**. The main idea is that the Adaptive Regression Forest updates in real time a subset of leaves which gives uncertain predictions. It is performed by two main originalities based on detection and update of passive leaves.

The first originality of our ARF is to detect the leaves giving poor information. There are two criteria for a leaf to become a passive leaf: having a high variance of predictive model; giving repeatedly a result rather different from the other leaves.

The second originality of our ARF is to update in real time passive leaves of the regression forest model. This is performed by re-modeling their predictive model from new computed labels. These labels are computed based on the results given by the other leaves (actives leaves). Note that the update is only performed on passive leaves and not on the whole regression forest model.

The second contribution is to present our **DynaLoc**. It is a real time camera relocalization in dynamic scenes based on the ARF. For this application, the ARF predicts 3D points in the world coordinates system which correspond to 2D points in the image. The originality is to keep the structure of the forest (trees and nodes) by using invariant SURF feature at split nodes as proposed in Section 3.4. Indeed, when objects move, the descriptors of detected keypoints are almost unchanged. Only their label, namely the 3D position in the world coordinates system corresponding to keypoints, changes.

This chapter is organized as follows:

- **Section 4.2** presents our Adaptive Regression Forest - ARF including two main steps: passive leaves detection and passive leaves update.
- **Section 4.3** presents our DynaLoc that is a real-time and accurate camera relocalization from only RGB images in dynamic scenes based on the adaptive regression forest.
- **Section 4.4** shows and discusses our results on rigid scenes dataset and on our dynamic scenes dataset.
- **Section 4.5** gives some conclusions and perspectives.

4.2 Adaptive Regression Forest

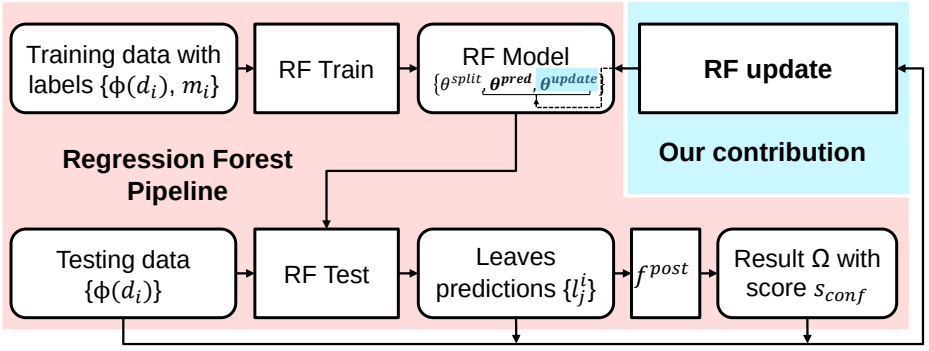


Fig. 4.1 Adaptive regression forest pipeline. The common regression forest method (red components) is extended with an online update step (blue component). It refines predictive model in real-time from new computed labels.

In this section, we first introduce the common regression forest pipeline which our method is based on. Then, we explain the limitations of this pipeline and how we overcome them by using our ARF method. Finally, we detail our ARF methodology including: definition ARF parameters, passive leaves detection and passive leaves update.

4.2.1 Regression Forest pipeline

Figure 4.1 presents the common regression forest pipeline (components in red) including two phases: training and testing. A regression forest is a set of N decision trees $\mathscr{F} = \{\mathscr{T}_j\}$. Each tree $\mathscr{T}_j = \{\theta^{split}, \theta^{pred}\}$ consists of split functions θ^{split} at internal nodes and predictive models θ^{pred} at leaf nodes. These parameters are learned from a set of labeled data $\{\phi(d_i), m_i\}$, where $\phi(d_i)$ is the feature vector extracted from data d_i with the label m_i. The split function θ^{split} is a weak learner. It is used to split a subset of data into left child node and right child node. The weak learner is trained to maximize an objective function aiming at reducing the variance. The training terminates when the tree reaches a maximum depth or when a node has few data. The predictive model at each leaf node is represented by a distribution $\theta^{pred} = \mathscr{N}(m, \bar{m}, \Sigma_m)$. It is computed from a set of labels m of data reaching this leaf. $\overline{m}$ and Σ_m are respectively the mean and the covariance of the Gaussian distribution.

For the testing phase, each testing data that is represented by a set of features $\{\phi(d_i)\}$ passes through the regression forest model to obtain multiple leaves predictions $\{l_j^i\}$, in which l_j^i is a prediction of the decision tree j for the feature $\phi(d_i)$. All predictions are combined to

compute the final output result Ω with a confidence score s_{conf} by using a post-processing function f^{post}.

4.2.2 Limitations of Regression Forest

In the regression forest, leaves with a high variance are not informative. They make noisy predictions. Thus, all leaves whose variance is greater than a threshold T_{var} are discarded to eliminate noise and improve accuracy. This leads to the fact that a subset of leaves are stored in the regression forest, but they are never used.

Another challenge of regression forest and of machine learning methods in general is facing dynamic data. Because a regression model is learned from static data, the pretrained regression model will be no longer accurate if some data changes. In this case, it requires re-training from scratch a whole model with a redefinition of data labels (new ground truth).

To overcome the limitations of the regression forest, we propose an adaptive regression forest. It is an extension of the common regression forest pipeline by adding an update step, as shown in Figure 4.1 (in blue). At the beginning, we assume that a regression forest is learned from initial training data. Then, in runtime, their labels change. We place our method in the case of features which are extracted from this data are almost unchanged. Hence, the split nodes of ARF keep accurate. The update step of our ARF improves consecutively accuracy of the regression forest model. It adapts robustly to dynamic data as well as refines unused leaves. This is performed by updating leaf nodes based on input data $\{d_i\}$, predictions $\{l_j^i\}$ and final output result Ω. Subsection 4.2.3 details our algorithm.

4.2.3 Methodology

In this paragraph, we first give the definitions of our ARF parameters. Then, we describe the two main steps of our update process: passive leaves detection and passive leaves update. This ARF update is detailed in Figure 4.2.

Definitions of ARF parameters

Several concepts are used in our ARF: active/passive leaf, good/bad prediction, validation function, confidence score. We present them in this subsection.

The parameters of each tree of a regression forest is extended to the ARF model:

$$\mathscr{T} = \{\theta^{split}, \theta^{pred}, \theta^{update}\}$$

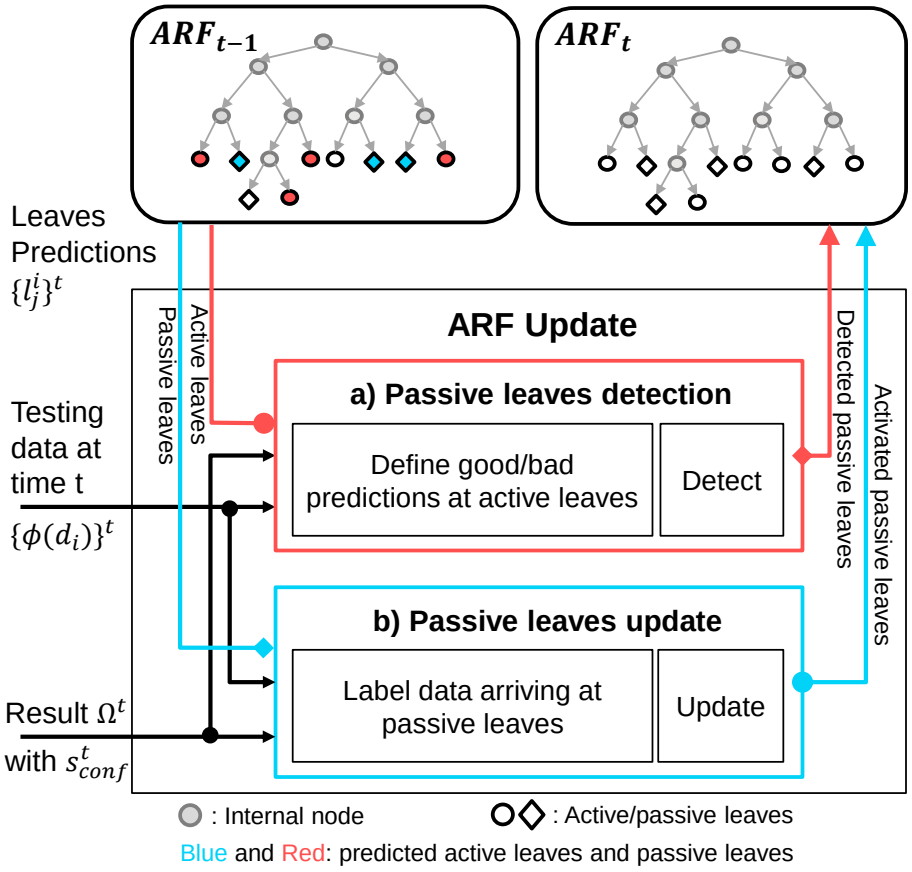


Fig. 4.2 Adaptive regression forest update process. The predictive models at leaf nodes evolve by part over time without training from scratch a whole new model. ARF performs simultaneously two steps: passive leaves detection and passive leaves update.

Where $\theta^{update} = \{s, n_{false}, S_{data}\}$ are the parameters used in the update process. s denotes the status of a leaf: $s \in \{0 : passive, 1 : active\}$. A passive leaf is a leaf whose prediction is not accurate, and is detailed in the next paragraph. We call it **passive** because those leaves are not used during the final process that use the random forest prediction. In contrary, predictions which come from **active** leaves are used to estimate final result. Therefore, the post-processing function in our ARF is defined as follows:

$$\Omega = f^{post}(\{d_i, l_j^i | s_j^i = 1\}) \tag{4.1}$$

Where s_j^i is the status of the leaf of the tree j corresponding to the feature vector $\phi(d_i)$.

n_{false} denotes the number of consecutive times an active leaf gives a bad prediction. S_{data} is a stack of data that stores the data d_i and the final result Ω at each passive leaf. We need it for passive leaves update.

We introduce a **validation function** f^{val} to define **good** and **bad predictions**.

$$f^{val}(f^{err}(\Omega, d_i, l_j^i)) = \begin{cases} 1, & f^{err}(.) < T_{val}, \text{ good} \\ 0, & otherwise, \text{ bad} \end{cases} \tag{4.2}$$

Where $f^{err}(\Omega, d_i, l_j^i)$ denotes an **error function** of each leaf prediction l_j^i based on the output Ω and input data d_i. T_{val} is an error threshold in order to determine good/bad predictions.

After the validation step, we obtain n_{good} and n_{bad} predictions. A score s_{conf} is calculated to evaluate the **confidence score** of final result:

$$s_{conf} = \frac{n_{good}}{n_{good} + n_{bad}} \tag{4.3}$$

If this confidence score is greater than T_{conf}, the update of ARF will be proceeded based on the output result Ω. T_{conf} is a confidence threshold to ensure that the output result is reliable. It aims at limiting accumulation of errors during the update process.

Passive leaves detection

Passive leaves detection aims at detecting leaves being no longer relevant and change their status to passive. After initial training phase, the status of each leaf is defined by:

$$s = \begin{cases} 1, active, & tr(\Sigma_m) < T_{var} \\ 0, passive, & \text{otherwise} \end{cases} \tag{4.4}$$

In the testing phase, an active leaf becomes a passive leaf, when it gives consecutively uncertain results. Figure 4.2-a) illustrates the passive leaves detection. We first define good/bad predictions from a set of active leaves predictions based on the validation function 4.2. We then use n_{false} to count the number of consecutive times an active leaf is considered as a bad prediction. Finally, if $n_{false} > T_{false}$, this active leaf becomes a passive leaf and its status is assigned to 0 (passive). T_{false} is a detection threshold to ensure that this leaf is really an uncertain leaf. n_{false} is reassigned to 0 as soon as the active leaf gives a good prediction once. This aims at avoiding mistakes of determining good/bad prediction from the error function in 4.2 due to noisy data.

Passive leaves update

The passive leaves update aims at remodeling the predictive models of passive leaves from new estimated labels. These new labels are calculated based on the estimated result Ω and input data $\{d_i\}$. Figure 4.2-b) show the passive leaves update step. It only processes at predicted passive leaves. When a feature $\phi(d_i)$ passes through our forest, if it terminates in a passive leaf. Firstly, we collect the corresponding input data d_i and the result Ω in a stack of data S_{data} associated to this passive leaf. These elements stored in the stack of data allow to compute new labels of data. When the number of data in the stack at a passive leaf is large enough $|S_{data}| > T_{data}$ (T_{data} is a threshold to ensure that the number of data is sufficient to learn a new distribution), we calculate labels m_i of d_i by using a labeling function:

$$m_i = f^{label}(S_{data}) \tag{4.5}$$

And then a gaussian distribution of this leaf is modeled from these labels $\{m_i\}$. Finally, the status of this leaf is defined by the function 4.4. The stack of data S_{data} is reset. The passive leaves update and passive leaves detection are performed at the same time to accelerate ARF system.

4.3 ARF applied to camera relocalization in dynamic environments

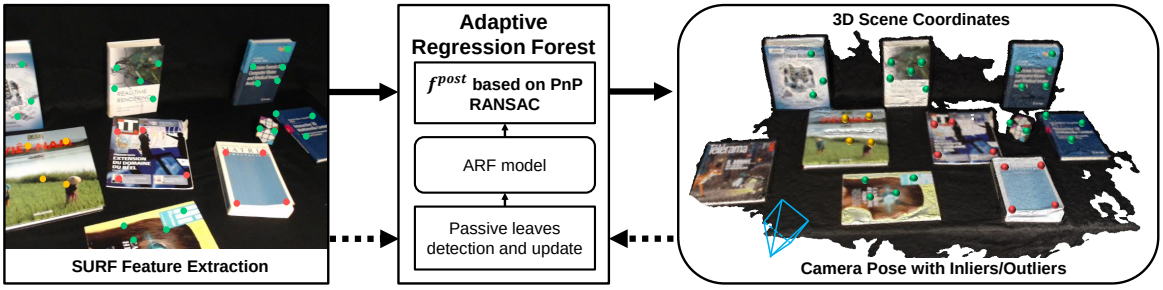


Fig. 4.3 DynaLoc: Real-time camera relocalization in dynamic scenes based on the ARF. We apply the ARF to hybrid camera relocalization: from a set of SURF feature detected on each RGB image, the ARF predicts a set of 3D positions in the world coordinate system. They correspond to active leaves (green and red points) and passive leaves (yellow points). Then PnP and RANSAC algorithms determine inliers (green points) to estimate camera pose and reject outliers (red points). Finally, if the camera pose is precisely estimated, the ARF is updated by estimating new 2D-3D correspondences based on a triangulation using the estimated camera poses.

In this section, we introduce our DynaLoc, a real-time camera relocalization method from only RGB images in dynamic scenes. Inspired by our sparse feature regression forest (see Section 3.4), we propose a hybrid method merging our ARF (described in Subsection 4.2.3) and geometric methods. Our ARF is applied to learn and predict 2D-3D point correspondences. The geometric part uses PnP and RANSAC algorithms in order to compute camera pose from these correspondences. Figure 4.3 illustrates our DynaLoc pipeline. Our method is summarized in three principal steps. Firstly, we present how to initially train the ARF from RGB images. Then, camera pose estimation is performed based on the ARF and geometric algorithms. Finally, we detail the online ARF update process for camera relocalization.

4.3.1 Initial training

The ARF is initialized according to the training step detailed in Section 3.4. An ARF is learned from a set of labeled feature vectors $\{\phi(d_i), m_i\}$ which are extracted from a set of RGB training images. $\phi(d_i)$ is a SURF feature vector extracted around a 2D keypoint position d_i. m_i is the 3D point in the world coordinates system of d_i. In this chapter, our method uses only RGB images for both training and testing phases. Thus, the label m_i is

defined by running triangulation algorithm [61] for each pair of matching keypoints (d_k, d_l) of two RGB images, whose poses (Ω_k, Ω_l) are supposed to be known in advance by using a localization system (marker based, 3D model based, SLAM, tracking, etc.):

$$\begin{cases} d_k \times (K\Omega_k^{-1} m_i) = 0 \\ d_l \times (K\Omega_l^{-1} m_i) = 0 \end{cases} \tag{4.6}$$

Where K is the matrix of the camera intrinsic parameters.

Each tree of ARF is initially trained by a random subset of data to determine the split functions. We use whole SURF feature vector as proposed in Section 3.4 and adaptive leaf nodes $\{\theta^{split}, \theta^{pred}, \theta^{update}\}$ (see 4.2.3). Each leaf node stores the 3D positions of data which is represented by a Gaussian distribution $\mathcal{N}(m, \overline{m}, \Sigma_m)$. The status s of each leaf of ARF is defined by the status definition function 4.4. n_{false} is assigned to 0 and S_{data} is initialized by empty set.

4.3.2 Camera pose estimation

Firstly, a set of SURF keypoints and features $\{d_i, \phi(d_i)\}$ is extracted from each RGB input image. They pass through the ARF $\{\mathcal{T}_j\}$ to achieve a set of predictions $\{l_j^i\}$ that contains 3D world coordinates predictions $\{\hat{m}_i\}$ corresponding to 2D SURF keypoints $\{d_i\}$. All 2D-3D correspondences coming from active leaves are used to estimate camera pose based on the post-processing function 4.1. In camera relocalization, f^{post} function of ARF is defined by PnP and RANSAC functions in order to remove bad predictions (outliers) and keep good predictions (inliers). RANSAC generates a set of hypothetical poses $\{\Omega_i\}$ by performing PnP on random subsets of 2D-3D point correspondences. The best inliers are defined by maximizing the number of inliers corresponding to each hypothesis. This is performed based on the validation function 4.2, in which the error function $f^{err}(.)$ is defined as a re-projection error function:

$$f^{err}(\Omega, d_i, l_j^i) = \|d_i - K\Omega_i^{-1}\hat{m}_i\|^2 \tag{4.7}$$

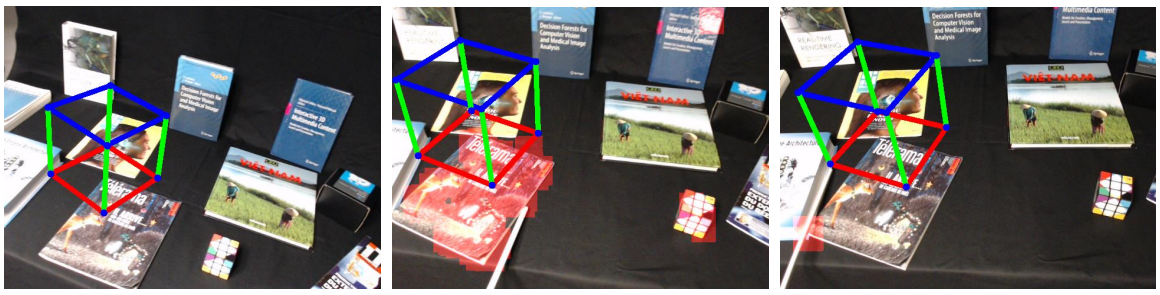
The final camera pose Ω is carried out by running PnP once on all inliers to minimize the sum of re-projection error.

4.3.3 Online adaptive regression forest update

The ARF in our DynaLoc is continuously updated to adapt to changes in dynamic scenes. This keeps predictions about 2D-3D correspondences being still accurate even if the scenes have changed. By using invariant SURF feature, when objects move, features which are

extracted from these objects are almost unchanged. So the relevancy of the split functions of ARF is maintained. Therefore, the ARF only updates predictive models at leaf nodes based on two main steps: passive leaves detection and update, as described in Subsection 4.2.3.

The camera pose estimation (see 4.3.2) defines inliers and outliers that correspond to good and bad predictions respectively. A confidence score is computed by function 4.3 to limit accumulation error during the ARF update. The inliers and the outliers are used to detect passive leaves. Simultaneously, the passive leaves update step is performed as in 4.2.3. Each passive leaf collects constantly 2D positions of SURF keypoints and camera poses (d_i, Ω_i) in a stack S_{data}. When passive leaves have a large number of data in the stack $|S_{data}| > T_{data}$, we calculate 3D position labels $\{m_i\}$ based on the labelling function 4.5 defined by a triangulation algorithm [61]. From a pair of data in the stack (d_k, Ω_k) and (d_l, Ω_l), the label m_i is defined based on the triangulation function 4.6. Thus, a set of T_{data} data at each passive leaf defines a set $\{m_i\}$ of $\frac{T_{data} \cdot (T_{data}-1)}{2}$ 3D points. A new Gaussian distribution $\mathcal{N}(m, \bar{m}, \Sigma_m)$ is modeled based on 3D points. The function 4.4 validates the status of new leaf model. According to this update process, even if incorrect estimated



a) At beginning b) Moving objects c) Updating ARF

Fig. 4.4 An example of ARF update. a) the scene with some movable objects. b) two objects move and corresponding active leaves are detected and become to passive, regions with blending red color. c) these passive leaves are updated to return to active state. The estimation of the camera pose remains accurate, which is indicated by the unchanged position of the virtual cube.

camera poses are taken into account, they do not affect too much the update. Indeed, if the camera pose estimation is incorrect, triangulation will fail and make false 3D points. Then a new predictive model which is computed from these 3D points will have very high variance. The function 4.4 will eliminate the new model and the data buffer will be reseted.

Figure 4.4 shows an example of camera relocalization in a dynamic scene. When some objects are moving, active leaves attached to these objects are detected and become passive leaves (red blending regions in Figure 4.4-b). After a few moments, almost all these passive leaves are updated to return to an active state as shown in Figure 4.4-c. The stability of the

camera pose estimation is illustrated by a virtual cube whose position remains fixed during the update process.

4.4 Experiments

Our method is implemented on a single CPU Intel Core i7 @2.9GHz. We extract up to 600 SURF features for each image. We train the ARF of 4 trees, the depth of each tree is 16. The threshold values T_{var}, T_{val}, T_{conf}, T_{false}, T_{data} are the same for all the experiments. Although the parameters have an influence on the accuracy, they do not highly depend on scenes or datasets. Indeed, we found a set of parameters to achieve the best results on a scene. And they work well with the other scenes. Our computational time for full process (including camera pose estimation and ARF update) is approximately $55ms$.

In the following paragraphs, we demonstrate the usefulness of our ARF on both static and dynamic scenes by comparing it with a regression forest (RF). Finally, we compare our DynaLoc with state-of-the-art methods.

4.4.1 ARF versus RF

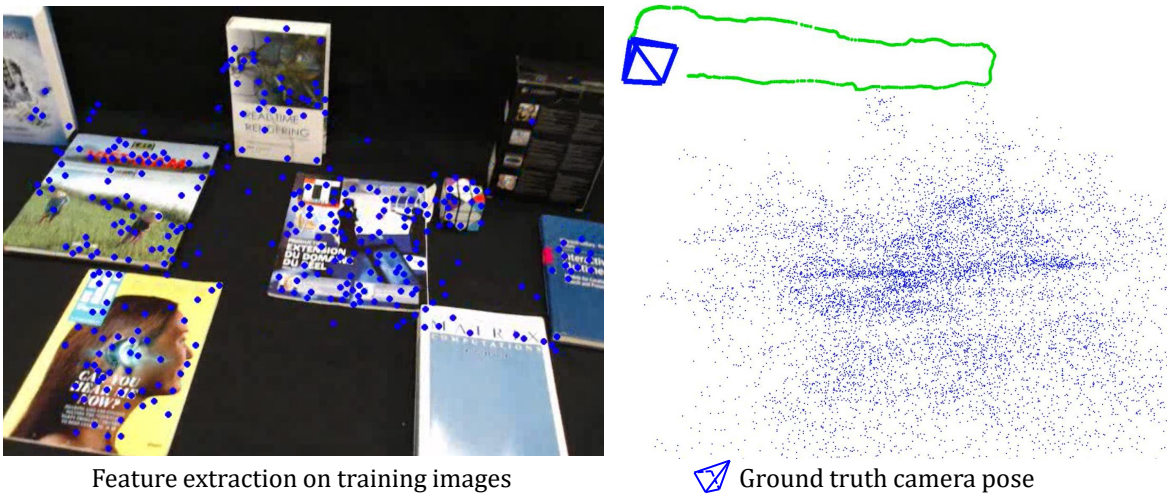


Fig. 4.5 Fast labelling 2D-3D point correspondences by using triangulation algorithm without a bundle adjustment optimization resulting in noisy data.

Fine-tune predictive model on static scenes. Supervised machine learning algorithms require high accurate ground truth to learn precisely a predictive model. But the ground truth is difficult to acquire in machine learning regression. Our ARF training only requires RGB images and their corresponding camera pose. The labelling of 2D-3D point correspondences is rapidly performed by using triangulation algorithm without a bundle adjustment optimization. Therefore, it makes some noisy correspondences data, as illustrated in Figure 4.5. However, our ARF can address this problem by fine-tuning predictive model from online

data. Firstly, we discard the uncertain predictive models given by passive leaves. Then these predictive models are recalculated based on active leaves update.

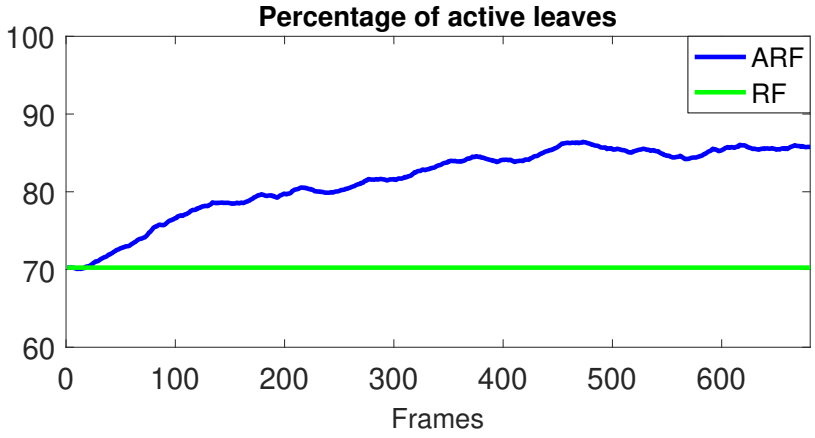


Fig. 4.6 The percentage of active leaves in the whole regression forest at each frame for the ARF strategy (blue) and a regression forest strategy (green) on the static sequence 01/01.

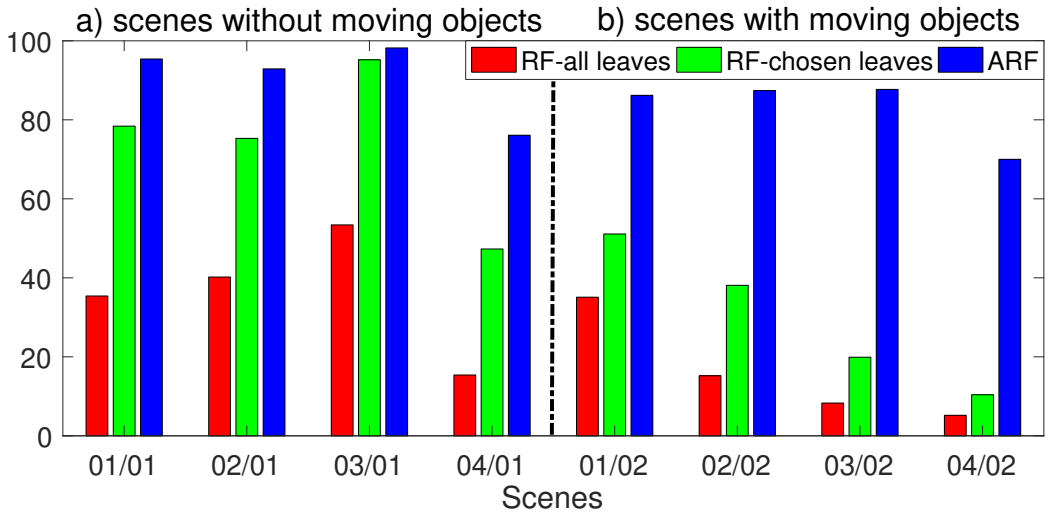


Fig. 4.7 Comparison our DynaLoc based on the ARF (blue) to RF approaches using all leaves (red) and chosen leaves (green) with $T_{var} = 0.1$ on DynaScenes dataset by measuring the percentage of test images where the pose error is below 5cm and 5°.

Figure 4.6 shows the percentage of active leaves in the forest on the static sequence DynaScene-01/Seq-01 with a variance threshold $T_{var} = 0.1$. Our ARF update increases the number of active leaves. The corresponding accuracy of camera relocalization on this sequence is shown in Figure 4.7-a). We compare our ARF with two other approaches based

on the random forest: the first one uses all the leaves of the forest (RF-all leaves) and the second one only uses leaves having a variance less than a threshold $T_{var} = 0.1$ (RF-chosen leaves). The results demonstrate that the removal of uncertain leaves significantly improve accuracy (RF chosen leaves gives better result than RF all leaves). And the ARF update increases accuracy further than the static strategy (RF).

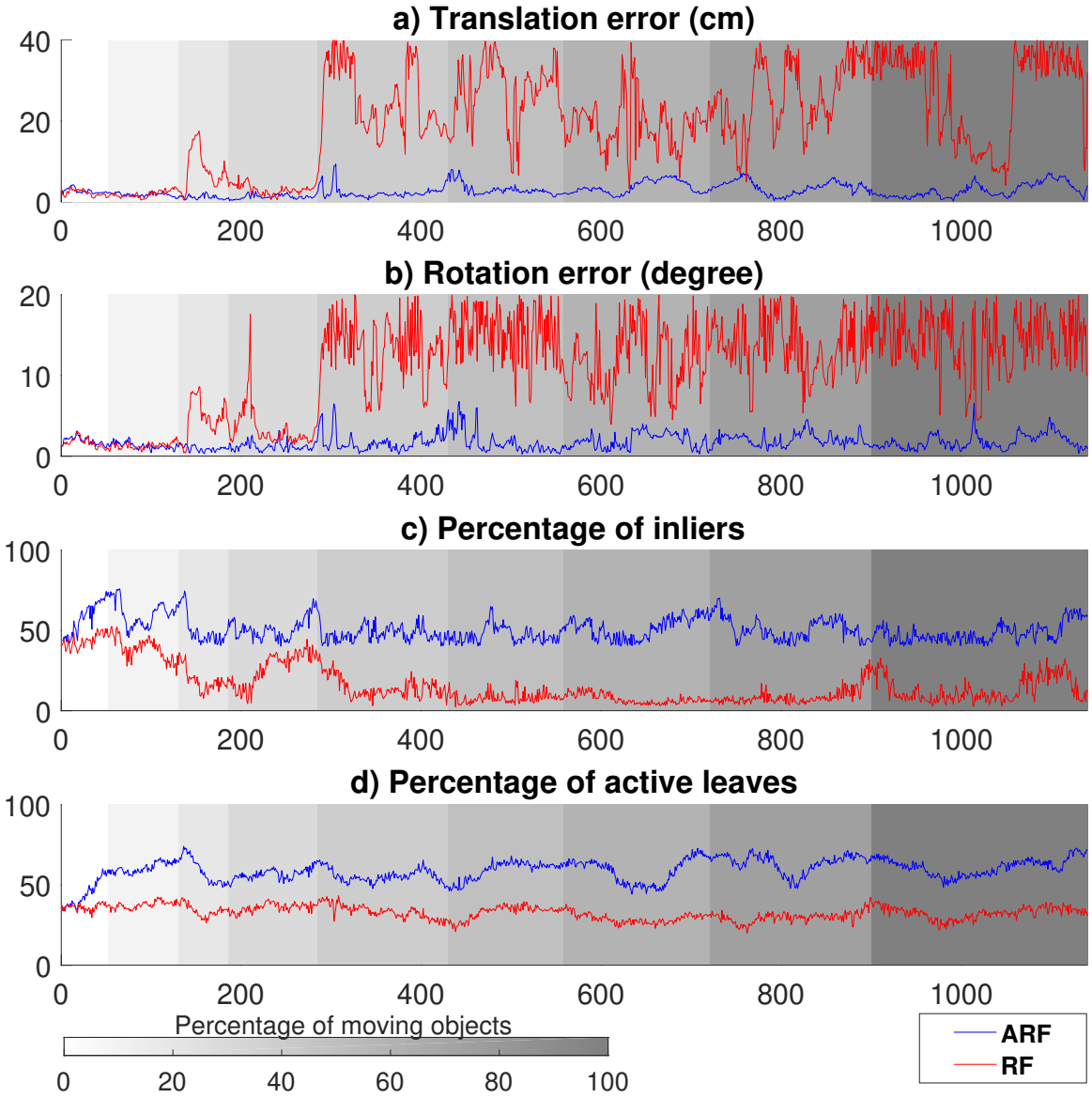


Fig. 4.8 Detail results of our DynaLoc based the ARF (blue) and RF (red) on DynaScene-03/Seq-02. a), b) translation error in centimeter and rotation error in degree. c) the percentage of number of inliers at each frame. d) the percentage of active leaves compared to the number of leaves used at each frame for predictions. The background color present the percentage of objects in the scene that have moved since the beginning.

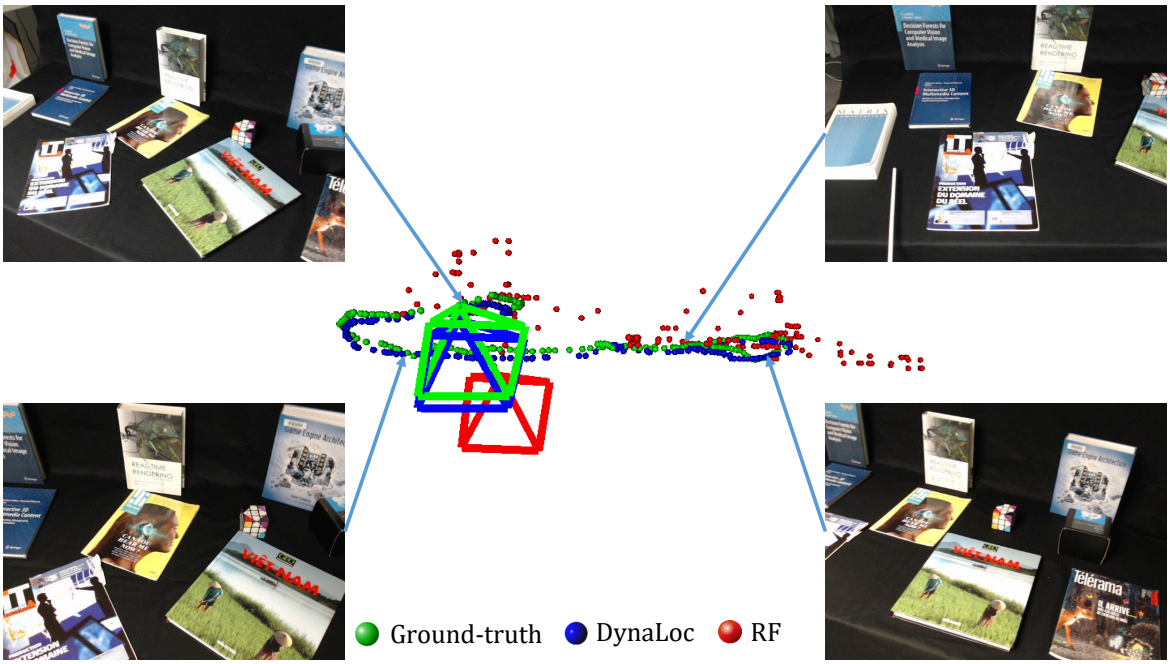


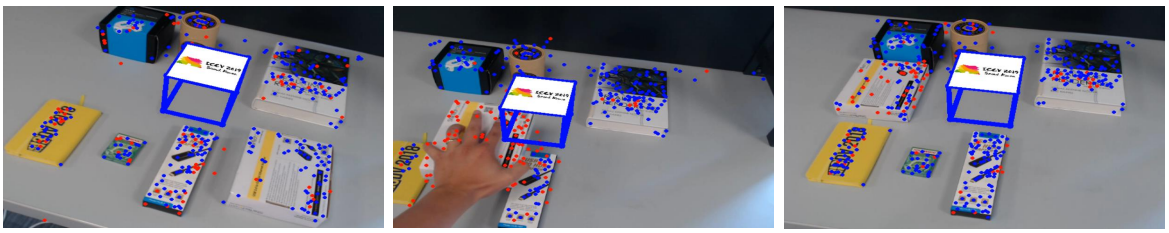
Fig. 4.9 Comparison results between DynaLoc and RF on a dynamic sequence.

ARF update performance on dynamic scenes. Figure 4.8 shows the performance of our ARF compared to a regression forest with chosen leaves (RF) on a dynamic sequence which contains objects moving gradually. The results demonstrate that when objects are static, both approaches achieve high accurate localization as shown at beginning of Figure 4.8-a,b). As soon as more than 30% of objects move, the RF approach has large error because the number of inliers reduces rapidly. On the other hand, the accuracy of the DynaLoc is maintained thanks to the update process.

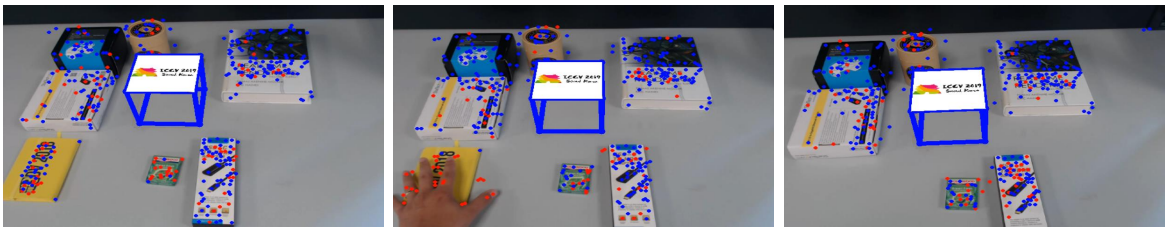
Figure 4.8-d) gives the evolution of the number of active leaves over time. When some objects move, the corresponding active leaves are defined as passive leaves and the percentage of active leaves drops. When these objects return to a static state, this percentage increases again thanks to the passive leaves update step. That is why the inliers percentage remains sufficiently high, as shown in Figure 4.8-c).

Furthermore, although the ARF update is proceeded based on estimated results, the error accumulation is very small. The rotation and translation errors before and after movements are approximately equal as shown in Figure 4.8-a,b). Therefore, our DynaLoc can handle a whole scene with gradual changes. In Figure 4.7-b), we compare the results of our DynaLoc with the RF (with or without chosen leaves) on dynamic scenes. We report that the accuracy of RF (with or without chosen leaves) drops drastically when the scene changes, whereas our DynaLoc has high accuracy on these scenes. Figure 4.9 also shows results on a part of the

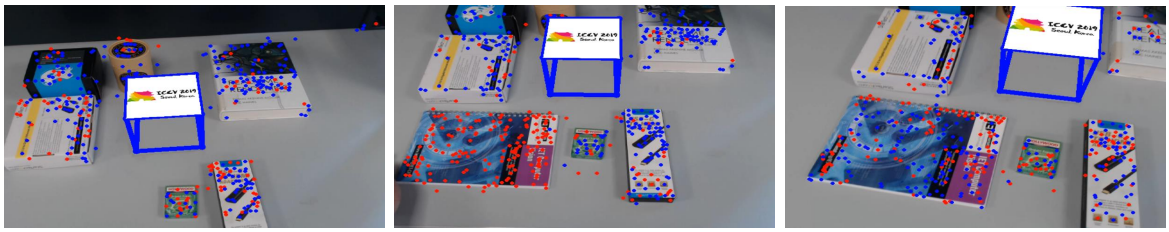
sequence Dyna-03/Seq-02. For this example, when some objects move gradually, the result of our DynaLoc remains stable. Conversely, the RF approach fails completely.



a) Move objects by the hand



b) Remove objects



c) Add new objects

c) Move objects outside observations

Fig. 4.10 Robustness of our method to different scenarios of dynamic scenes. The stability of the camera pose estimation is illustrated by a virtual cube whose position remains fixed during the update process over time. Inliers (blue points) and outliers (red points) are displayed on objects.

Figure 4.10 show some scenarios of dynamic scenes such as moving objects by the hand, removing objects, adding new objects or moving objects outside observations. Our method handles instant moving objects. We detect passive leaves corresponding to moving objects and the passive leaves update successes immediately when these objects are stationary. Our

method does not require any motion tracking of the objects. Technically, if moving objects or new objects are observed, corresponding features will be considered as outliers, and the rest of the scene will be used to estimate the camera pose. When object becomes static again, they will be updated to the model and their features will be considered as inliers. Thus, our method achieves high accuracy for these scenarios as well.

4.4.2 Comparison to state-of-the-art methods

Baselines. We compare our method to three different approaches: geometric based [151], machine learning based [82] and hybrid based [23] methods. We also compare our DynaLoc to our hybrid method based on static regression forest (SURF-RF) presented in Subsection 3.4.

Table 4.1 Comparison of our method with state-of-the-art methods. The accuracy is evaluated by median pose errors on 7-scenes dataset.

Scene	Active Search [151]	PoseNet2 [82]	DSAC++ [23]	SURF-RF	DynaLoc-1	DynaLoc-2
Chess	0.04m,2.0°	0.13m,4.5°	**0.02m,0.5°**	0.034m,1.45°	0.031m,1.31°	0.029m,1.27°
Fire	0.03m,1.5°	0.27m,11.3°	**0.02m,0.9°**	0.029m,1.34°	0.024m,1.27°	**0.021m**,1.15°
Heads	0.02m,1.5°	0.17m,13.0°	**0.01m,0.8°**	0.020m,1.32°	0.019m,1.30°	**0.016m**,1.21°
Office	0.09m,3.6°	0.19m,5.6°	**0.03m,0.7°**	0.045m,1.70°	0.043m,1.52°	0.037m,1.37°
Pumpkin	0.08m,3.1°	0.26m,4.8°	**0.04m,1.1°**	0.046m,1.70°	0.045m,1.57°	**0.042m**,1.42°
kitchen	0.07m,3.4°	0.23m,5.4°	**0.04m,1.1°**	0.047m,1.97°	0.042m,1.73°	**0.038m**,1.60°
Stairs	**0.03m**,2.2°	0.35m,12.4°	0.09m,2.6°	0.074m,2.05°	0.068m,1.87°	0.065m,**1.75°**
Average	0.05m,2.5°	0.23m,8.1°	**0.04m,1.1°**	0.042m,1.65°	0.039m,1.51°	**0.035m**,1.40°

Table 4.2 Comparison of our method with DSAC++ in term of runtime. Training time per scene and testing time per image.

	DSAC++ [23]	DynaLoc
Configuration	GPU Tesla K80 Intel Xeon E5-2680	**Intel Core i7-7820HK**
Training time	1-2 days	**5-10 minutes**
Testing time	220ms	**55ms** (5ms for ARF update)

7 scenes dataset. For this experiment, all methods use RGB-D training images and RGB testing images. To evaluate the performance of our DynaLoc on these static scenes, we run

DynaLoc on a testing sequence to have the first result **DynaLoc-1** in Table 4.1 and the refined model. We repeat once again the evaluation on testing sequences with the refined model to obtain the second result **DynaLoc-2**. It can be noted that the refined model only uses RGB testing images without ground truth labels. In Table 4.1, both our results clearly outperform PoseNet2 [82] and our real-time hybrid method (SURF-RF) on all scenes. And they are slightly better than Active Search [151] on this dataset except the translation estimation error on the *stairs* scene. The results also shows that the accuracy of our method is approximately equal to DSAC++ [23]. In term of the runtime, our method is much faster than DSAC++ for both training and testing, as shown in Table 4.2. When we compare Dynaloc-1 and DynaLoc-2, we notice that the refinement over time of ARF from real data improves significantly the accuracy of the model as well as the results.

Table 4.3 Comparison of our DynaLoc with our sparse feature regression forest (SURF-RF). The accuracy is evaluated by median pose errors on DynaScenes dataset.

Sequence	SURF-RF	DynaLoc
Dyna-01/Seq-01	2.9cm, 1.9°	**2.6cm, 1.7°**
Dyna-02/Seq-01	3.2cm, 2.6°	**3.0cm, 2.5°**
Dyna-03/Seq-01	1.9cm, 1.4°	**1.4cm, 1.2°**
Dyna-04/Seq-01	4.3cm, 3.8°	**3.3cm, 1.9°**
Dyna-01/Seq-02	4.7cm, 2.9°	**2.2cm, 1.6°**
Dyna-02/Seq-02	7.2cm, 5.1°	**3.5cm, 2.7°**
Dyna-03/Seq-02	19.8cm, 14.3°	**2.2cm, 1.5°**
Dyna-04/Seq-02	25.6cm, 20.6°	**3.2cm, 1.7°**
Average	8.7cm, 6.6°	**2.7cm, 1.9°**

DynaScenes dataset. It is a completely new dataset of dynamic scenes that we created and presented in Section 2.6. Table 4.3 compares median camera pose errors of our DynaLoc and of our previous work, SURF-RF, on the DynaScenes dataset. The results of DynaLoc are moderately better than SURF-RF on the sequences Dyna-$\{01,02,03,04\}$/Seq-01 where there are challenging partial occlusion, illumination changes without moving objects. Both two methods achieve high accuracy thanks to the use of SURF features and RANSAC algorithm. However, on the remaining sequences, SURF-RF only obtains moderate accuracy for two scenes Dyna-$\{01,02\}$/Seq-02 that contain respectively 30% and 60% moving objects. For the scenes Dyna-$\{03,04\}$/Seq-02 where all the objects move gradually, the accuracy of SURF-RF drops significantly because RANSAC cannot eliminate a lot of outliers on moving

objects. Inversely, our method still estimates precisely thanks to passive leaves detection and update.

4.5 Conclusion

In this chapter, we proposed an adaptive regression forest that can update itself during runtime with current observations to tackle the challenge of dynamic data. This is performed by detecting and updating passive leaves of a regression forest. We applied our adaptive regression forest to our DynaLoc, a real-time and accurate RGB camera relocalization for dynamic scenes with moving objects. The results of camera relocalization in dynamic scenes show that our method is able to address a large number of gradually moving objects. Our method achieves results as accurate as the best state-of-the-art methods on static scenes dataset but performs more quickly for both training and testing time. Our DynaLoc is robust to occlusion and illumination changes. Moreover, we also obtain high accuracy even on our dynamic scenes dataset.

Chapter 5

Conclusions and Perspectives

Contents

5.1 Conclusions

In this thesis, we investigated **camera relocalization** based on RGB images. Image-based camera relocalization has recently been applied in many areas, such as augmented/virtual/mixed reality, robotics and autonomous vehicles, but many limitations prevent its use for many use cases.

We started by presenting theory of computer vision and deep learning for camera relocalization. We also summarized state-of-the-art methods of camera relocalization according to different approaches. We then introduced camera relocalization datasets and metrics to evaluate methods in terms of accuracy, computational time, generalization, robustness to dynamic scenes (occlusion, illumination changes, moving objects).

The goals of this thesis were to propose a real-time and accurate camera relocalization from RGB images in dynamic environments. We first presented our **PatchPoseNet**, which uses MIMO strategy based on local patches to overcome the uncertainty of end-to-end machine learning camera regression. Afterward, we presented the evolution of our hybrid method that balances between accuracy and computational time. Our hybrid methods are based on both machine learning approach and geometric approach and aim at benefiting from both. We proposed our **xyzNet** and our sparse feature regression forest (including **SURF-RF** and **Deep-RF**) that focus on improving both computational time and accuracy for camera

relocalization while, at the same time, addressing challenges of occlusion and illumination changes. Our latest method **ARF** is considered as the most important one of this thesis. Based on regression forest process, it adapts itself in real-time to predictive model. We applied it to the camera relocalization. The resulting hybrid method, **DynaLoc**, is a real-time and accurate camera relocalization from only RGB images in dynamic environments. Based on DynaLoc, we also developed a ***Smart AR Toolbox* [Instant LeARning]** in **IRT b-com**. The results of our various contributions are summarized and illustrated in Table 5.1.

Table 5.1 Summary of our methods regarding the camera relocalization challenges. For runtime and training time, gray color denotes methods using GPU, white color denotes methods using only CPU. + means **good** handling and ++ means **very good** handling.

Methods	Accuracy	Runtime	Training time	Dynamic Scene		
				Occlusion	Illumination	Moving objects
PatchPoseNet		++		+	+	
xyzNet	+	++	+	++	+	
Deep-RF	++	++	+	++	++	
SURF-RF	++	++	++	++	++	
DynaLoc	++	++	++	++	++	++

5.2 Limitations and future works

Although our work provides a real-time and accurate camera relocalization component for localization system, it still encounters challenges of texture-less scenes and large-scale scenes. Therefore, in this section, we discuss the remaining problems of our proposed methods and suggests further research directions.

- **Camera relocalization in texture-less scenes**. Our sparse feature based methods allow us to have a trade-off between runtime and accuracy for camera relocalization. They can also perform in texture-less scenes. However, in the computer vision, texture-less is still known as one of the main problem of sparse feature approaches. It is due to the fact that it is difficult to detect stable keypoints and their feature is less discriminative. Therefore, this takes more time to learn regression models and it reduces the accuracy of 2D-3D point correspondences. To handle this challenge, a well-known approach is based on both points and lines in texture-less scenes. Nevertheless, line-based feature is still less efficient. It is more efficient for tracking than recognition. So we expect to formulate hand-crafted or learned line-based feature to improve the accuracy of camera relocalization in texture-less scenes.

- **Camera relocalization in large-scale scenes**. Scalability in large-scale scenes allows the localization systems to keep the accuracy when the scene is expanded. This leads to the storage of huge set of known information. Consequently, memory usage as well as processing time increase with respect to the size of the models. For extremely large-scale scenes, although our learned models only require a fixed memory and processing time to relocalize each frame, prediction accuracy of our light model may decrease as the scale of the scene expands. We can address large-scale scenes by learning a network with more neurons and more layers network or deeper trees in regression forest. But doing so needs more time for both training and testing. Thus, we suggest a future approach using multi-model learning for camera localization in large-scale scenes. Instead of learning a big model for the whole scene, we can rapidly learn multiple models corresponding to each part of the scene based on our online regression forest learning. It also allows to develop a collaborative multi-agent (re)localization system.

- **Improving camera relocalization by fusing world coordinate regression and object detection**. Recently, multi-task learning has been widely adopted in many computer vision tasks to enhance overall computation efficiency or boost the performance of individual tasks, with the assumption that those tasks are correlated and complementary to each other. Our current work, presented in this thesis, employed machine learning for regressing camera pose or regressing 3D world coordinate. However, the appearance of each frame also provides semantic information of scenes. Methods such as objects detection or segmentation can help to rapidly remove outliers of 2D-3D point correspondences. Such methods could speed up and increase accuracy of PnP and RANSAC algorithms. We thus suggest an improvement of camera relocalization based on multi-task learning.

Résumé en français

Introduction

Ces dernières années, la Réalité Virtuelle/Augmentée/Mixte (RV/RA/RM), la robotique, les véhicules autonomes sont devenus de plus en plus à la mode dans l'Industrie 4.0. Tous les systèmes de RV/RA/RM nécessitent un composant de localisation. La RA utilise des capteurs et des algorithmes pour déterminer la pose de la caméra et la pose des objets dans l'environnement, avec une grande précision et en temps réel. La RV/RM doit définir la pose du casque HMD (Head-Mounted Display) dans l'environnement simulé. Pour la robotique et les systèmes autonomes, l'estimation de la pose est un élément clé pour la navigation automatique dans un espace 3D. Compte tenu de la criticité de l'estimation de la pose de caméra dans les domaines mentionnés ci-dessus, l'amélioration de ses performances est vraiment essentielle.

L'estimation de la pose de caméra consiste à définir la pose de caméra qui comporte six degrés de liberté (6-DoF) exprimés dans le système de coordonnées du monde. La plupart des solutions existantes pour l'estimation de la pose de caméra sont liées à l'utilisation de plusieurs capteurs tels que caméra, GPS, LIDAR et IMU. Parce que la navigation dans les environnements extérieurs peut impliquer des scénarios compliqués qui sont difficilement gérés par une seule modalité de capteur.

Récemment, avec le développement rapide de la vision par ordinateur et de l'apprentissage automatique, de nombreuses méthodes basées sur des images ont été développées. Parmi elles, les deux solutions les plus courantes d'estimation de pose de caméra pour les systèmes commerciaux sont SfM (Structure from Motion) et SLAM (Simultaneously Localization And Mapping). Les méthodes SfM traitent hors ligne un ensemble d'images non ordonnées sans contrainte temporelle pour estimer la pose de caméra en utilisant des caractéristiques correspondantes parmi des paires d'images. Grâce à ces appariements, ils peuvent à la fois reconstruire un modèle de scène 3D et estimer la pose de caméra. Inversement, les méthodes SLAM peuvent fonctionner en temps réel sur une séquence ordonnée d'images acquises à partir de caméra, potentiellement associées à une unité de mesure inertielle. La majorité des

systèmes SLAM sont basés sur le suivi du mouvement de la caméra, en utilisant des trames consécutives pour calculer robustement la pose de caméra. Malheureusement, dans le cas d'un mouvement de caméra rapide ou d'un changement de point de vue soudain, l'échec du suivi interrompt l'estimation de la pose de caméra. Lorsque cela se produit, la relocalisation de caméra est nécessaire pour récupérer la pose de caméra, plutôt que de redémarrer la localisation à partir de zéro.

La relocalisation de caméra est un élément important des systèmes de localisation. Il permet de définir la pose de caméra à partir d'images individuelles sans contrainte temporelle sur la base de modèles construits à partir de l'information connue d'une scène. Cependant, il est encore difficile d'avoir une méthode précise, temps réel et robuste aux environnements dynamiques. Par conséquent, cette thèse se concentre sur la recherche de solutions afin de relever les défis de relocalisation de caméra mentionnés ci-dessus.

État de l'Art

Sur la base du mécanisme de modélisation des informations préalables, nous présentons des méthodes de relocalisation de caméra selon quatre approches différentes : des approches géométriques ; des approches d'apprentissage automatique ; des approches hybrides ; des approches de récupération d'images.

Approches géométriques

Les approches géométriques pour la relocalisation de caméra sont basées sur un nuage de points 3D pré-calculé. Ces approches associent d'abord directement les caractéristiques 2D (telles que SIFT, SURF, ORB) de l'image de requête aux points 3D dans la carte pour définir les correspondances de points 2D-3D pour les images RVB ou les correspondances de points 3D-3D pour les images RVB-D. La pose de caméra est ensuite estimée en résolvant un problème de pose absolue de caméra via des algorithmes PnP (2D-3D) ou Kabsch (3D-3D). En présence de fausses correspondances (outliers), l'algorithme RANSAC est appliqué pour les éliminer et accélérer le calcul.

Les deux étapes les plus importantes de cette approche consistent:

- À créer un modèle 3D de la scène contenant un ensemble de points 3D dans le système de coordonnées de la scène associé à des vecteurs des caractéristiques.
- À faire correspondre les caractéristiques 2D d'une image de requête au modèle 3D pour définir les correspondances de points 2D-3D.

Un modèle de scène 3D est construit à partir d'un ensemble d'images observées autour d'une scène. Ceci est effectué par des méthodes hors ligne (SfM) ou en ligne (SLAM). La correspondance directe de points 2D-3D est ensuite effectuée selon deux stratégies:

- F2P (Feature-to-Point), consistant à prendre chaque caractéristique dans l'image de la requête, et chercher le meilleur point correspondant dans le modèle 3D.
- P2F (Point-to-Feature), consistant inversement à faire correspondre les points du modèle 3D aux caractéristiques 2D de l'image de requête.

L'algorithme le plus largement utilisé pour la recherche du voisin le plus proche est kd-tree [158], k-means [132].

Récemment, [149, 151] ont utilisé un vocabulaire visuel pour réaliser une correspondance 2D-3D efficace. Les approches géométriques sont simples, précises et particulièrement utiles lorsque les images de requête sont trop éloignées des images d'apprentissage. Toutefois, ces méthodes sont limitées à des scènes relativement petites en raison du coût d'appariement qui, selon le schéma d'appariement utilisé, peut croître de façon exponentielle en fonction du nombre de points-clés. La mise en correspondance des caractéristiques locales peut être bruitée et peu fiable dans les scènes comportant des motifs répétés. En outre, pour obtenir un modèle 3D précis et efficace, la construction et l'optimisation du modèle 3D prennent beaucoup de temps.

Approches d'apprentissage automatique

Pour les approches d'apprentissage automatique, la relocalisation de caméra est un problème de régression résolu par un apprentissage supervisé. Elle est réalisée à partir des informations connues à l'avance de chaque scène. La phase d'apprentissage utilise des images étiquetées (les images et leurs poses correspondantes). La phase de test utilise ce modèle appris pour relocaliser caméra à partir de chaque image.

[83, 81] ont été les premiers à proposer l'utilisation de l'apprentissage en profondeur comme approche d'estimation de pose de caméra de bout en bout. Dans leurs méthodes, un modèle de réseau de neurones convolutif (CNN) est appris à partir d'images entières labellisées à l'aide des poses de la caméra.

Ensuite, le modèle entraîné prédit directement la pose de la caméra à partir de chaque image RVB. [83] présente la façon d'adapter le modèle de GoogleNet de la classification à la régression en modifiant ses couches finales afin de régresser la pose de la caméra. La phase d'entrainement est réalisée avec une fonction de perte qui est la somme de l'erreur de localisation et de l'erreur d'orientation. Un facteur d'échelle est utilisé pour maintenir les

deux valeurs d'erreur à peu près égales. Cependant, la valeur du facteur d'échelle dépend de chaque scène, ce qui rend les paramètres d'une nouvelle scène difficiles à configurer. [81] donne un moyen de générer une estimation de pose probabiliste en utilisant le dropout après chaque couche de convolution comme moyen d'échantillonnage des poids du modèle. [82] résout une ambiguïté du facteur d'échelle entre erreur de localisation et erreur d'orientation dans la fonction de perte de [83] par une nouvelle fonction de perte basée sur une erreur de reprojection. [32] exploite l'information temporelle en utilisant plusieurs images pour la prédiction de pose. Un LSTM (Long Short Term Memory) qui est l'extension du réseau de neurones récurrent (RNN) est utilisé pour prédire la pose de caméra pour chaque image de la séquence à partir du vecteur de caractéristiques du CNN.

L'apprentissage automatique (machine learning) est apparu comme un moyen de surmonter certaines limitations des approches géométriques (utilisation de la mémoire à grande échelle, temps d'appariement). Il fournit une solution compacte d'estimation de la pose de caméra de bout en bout. Mais la phase d'entraînement prend des heures, même des jours pour une petite scène. Cependant, les limites les plus importantes de ces méthodes résident dans leur précision modérée et l'absence de score de confiance pour chaque estimation de pose. Ces méthodes sont nettement moins précises que les méthodes basées sur la géométrie. Elles semblent plus compatibles avec la récupération d'image qu'avec la régression de pose de caméra.

Approches de récupération d'images

Tandis que les deux approches ci-dessus estiment directement la pose de caméra à partir de correspondances géométriques ou d'un apprentissage de régression, à l'inverse, l'approche de récupération d'image regroupe des méthodes indirectes qui transforment la relocalisation de caméra en problème de récupération d'image et fournissent une pose grossière relative à l'image de la requête.

Le but des méthodes de récupération d'images les plus proches est de récupérer un ensemble d'images présentes dans la base de données et similaires à l'image de requête d'entrée. Les images récupérées fournissent des informations sur l'emplacement possible de l'image de requête. Ce problème de récupération d'images comprend deux étapes: extraction d'une caractéristique de niveau image (en utilisant BoW [48] ou caractéristiques apprises [5]) pour l'image de requête et la base de données et recherche de la récupération d'images la plus proche dans la base de données. Ensuite, la pose finale de l'image de la requête est calculée à partir des images récupérées. Il est possible d'utiliser une partie du modèle 3D d'une scène visible dans les images récupérées pour définir directement la pose de caméra en utilisant les approches géométriques [148, 124, 126]. La pose précise de caméra peut

également être obtenue en utilisant des correspondances de caractéristiques entre l'image de requête et les images récupérées pour estimer une pose relative [87, 52, 94].

L'approche de récupération d'image améliore le temps d'appariement dans l'approche géométrique en proposant une pose grossière à partir des images récupérées les plus proches. Toutefois, ces méthodes sont souvent imprécises si l'image de requête est capturée à partir d'une pose très éloignée de celle de la base de données. Pour le système de localisation, cette approche doit stocker un grand nombre d'images clés. Par conséquent, l'utilisation de la mémoire et le temps de traitement augmentent en fonction de la taille des modèles.

Approches hybrides

Les approches hybrides estiment la pose de caméra en combinant les approches d'apprentissage automatique et les approches géométriques. La partie d'apprentissage automatique est appliquée pour apprendre et prédire une position 3D de chaque pixel en coordonnées du monde au lieu d'une pose de caméra 6-DoF directement fournie dans les approches d'apprentissage automatique ci-dessus. En effet, la dimension élevée (6-DoF) de l'espace de recherche risque d'aboutir à une mauvaise estimation. De plus, la partie apprentissage machine définit les correspondances de points plus rapidement que les approches géométriques. Puis la partie géométrique déduit la pose de caméra à partir de ces correspondances.

Les méthodes présentées dans le présent paragraphe apprennent et prédisent la position 3D de chaque pixel d'une image. Dans [156, 59, 176], chaque pixel effectue une prédiction continue de sa propre position 3D dans le système de coordonnées du monde. Ensuite, la pose de la caméra est calculée en utilisant l'algorithme de Kabsch sur les correspondances de points 3D-3D, à savoir la prédiction 3D en coordonnées du monde avec le point 3D correspondant dans les coordonnées de la caméra déduites de l'image de profondeur. [170] propose une méthode CNN-SLAM, où un CNN est intégré pour prédire une carte de profondeur à partir d'une seule image RVB. Cependant, la prédiction par CNN de la carte de profondeur est très consommatrice de temps. Ce travail est donc effectué uniquement sur les images clés et les poses de caméra pour les images intermédiaires sont estimées en fonction de l'image-clé la plus proche.

Bien que ces méthodes atteignent une plus grande précision, elles nécessitent des milliers de prédictions sur les coordonnées du monde, de sorte que le temps de calcul nécessaire pour déduire la pose optimale de caméra par l'algorithme RANSAC augmente beaucoup.

Contributions

Régression de la pose de caméra basée sur des patchs locaux

L'un des problèmes posés par la régression de bout en bout pour relocaliser une caméra réside dans le manque d'un score de confiance attaché à la pose fournie. Afin de résoudre ce problème, [81, 32] créent un modèle probabiliste des résultats en utilisant des couches de dropout après chaque couche de convolution pour échantillonner les poids du modèle. Notre méthode utilise à la place un ensemble de patchs pour générer un ensemble de résultats probabilistes à partir des données.

Nous proposons un CNN basé sur une architecture de style VGG permettant de déduire la pose de caméra à partir de chaque patch local, appelé **PatchPoseNet**. Nous extrayons au préalable des patchs locaux d'images RVB contenant des informations importantes (amplitude maximale du gradient de l'image). Pour la phase d'entraînement, chaque patch extrait est associé à une étiquette qui représente la vérité terrain de l'image correspondante. Notre réseau est formé en minimisant une fonction de perte similaire à celle utilisée dans [83]. Pour la phase de test, tous les patchs d'une image testée sont passés à travers le modèle appris pour donner une pose de caméra multi-sorties. Le regroupement non paramétrique par l'algorithme mean-shift est utilisé pour combiner les votes de la pose de caméra. Cela peut réduire l'effet de la prédiction bruitée sur l'estimation finale.

Notre méthode **PatchPoseNet** s'attaque efficacement au problème de l'incertitude dans l'apprentissage en profondeur par régression. Cependant, il est également nettement moins précis que les méthodes géométriques. Notre méthode est toujours limitée à la fonction de perte combinant rotation et erreur de translation par coefficient de poids. En outre, l'extraction de multiples patchs dans chaque image augmente drastiquement le volume des données d'entraînement. Cela rend la convergence de cette phase d'entraînement difficile à assurer. De plus, en cas de rotation, les patchs extraits de la même position sont presque inchangés. Cela réduit le pouvoir discriminant de la méthode.

Régression des coordonnées 3D du monde basée sur des patchs locaux

Nous proposons ensuite une méthode hybride mélangeant à la fois l'approche de l'apprentissage automatique et l'approche géométrique. Notre principale contribution est de proposer une méthode de relocalisation de caméras en temps réel.

Inspiré par PatchPoseNet, nous présentons un réseau de neurones convolutif (CNN) léger, appelé **xyzNet** pour calculer robustement par régression les coordonnées 3D dans le repère du monde des points réels associés aux pixels d'une image, et ainsi s'affranchir d'une dimension

excessive de l'espace de recherche, susceptible de fausser l'estimation. Ainsi, au lieu d'une régression de bout en bout de la pose de caméra, chaque patch donne une prédiction des coordonnées 3D de son centre dans le référentiel de la scène. L'originalité de notre méthode réside dans le choix délibéré de patchs centrés autour de points-clés (de préférence à un tirage au sort) pour cibler uniquement les régions géométriquement pertinentes. Ainsi, chaque patch retenu est une portion d'image de taille fixe autour d'un point-clé. Nous utilisons le détecteur SURF pour détecter automatiquement des points dispersés invariants au facteur d'échelle et à la rotation, en tant que représentants répétitifs dans une scène. Cela améliore la capacité à localiser les positions 3D à partir des patchs. Pour la phase d'entraînement de xyzNet, nous devons labelliser les données d'entraînement avec les coordonnées du monde 3D correspondantes. À cette fin, nous pouvons exécuter l'algorithme SfM une fois sur tous les ensembles de données d'apprentissage ou utiliser les images RVB-D de caméra calibrée avec leurs poses de caméra correspondantes pour effectuer une cartographie entre les points-clés et le nuage de points. Enfin, à partir des correspondances de points 2D-3D établies par notre réseau, nous implémentons des algorithmes PnP et RANSAC afin d'estimer la pose de caméra. La pose finale de caméra est attachée à un score de confiance qui est défini comme le nombre d'inliers.

Notre méthode hybride xyzNet rend possible une relocalisation de caméra en temps réel. De plus, grâce à l'utilisation de patchs au lieu d'une image complète, elle se révèle robuste à l'occlusion. Bien que nous obtenions sur la plupart des scènes de meilleurs résultats que les méthodes de l'état de l'art aptes au temps réel, notre méthode rencontre des difficultés sur des scènes répétitives où nous n'obtenons pas assez d'inliers après élimination de l'ambiguïté de prédiction uni-modale de notre réseau.

Régression des coordonnées 3D du monde de multi-sorties efficace

Notre objectif dans ce paragraphe est de proposer une méthode de relocalisation temps réel et précise en utilisant uniquement des images RVB. La limitation principale des méthodes hybrides de l'état de l'art concerne le temps de calcul de la partie géométrique du processus. Ce temps du calcul est élevé, en raison du nombre excessif de données utilisées comme entrées de cette partie géométrique, et aussi de leur faible précision. Par conséquent, nous présentons une méthode hybride, qui se concentre sur l'amélioration des données fournies par la partie apprentissage automatique du processus.

Nos contributions visent à diminuer le nombre de prédictions des correspondances 2D-3D traitées à chaque étape de la partie d'apprentissage automatique, et à augmenter leur pertinence dans le même temps. Nous proposons une forêt de régression efficace à sorties multiples basée sur une détection de caractéristiques éparses. Nous définissons une nouvelle

fonction de répartition à chaque nœud de la forêt de régression, qui prend des vecteurs de caractéristiques entiers comme entrées. Cette fonction de répartition permet d'améliorer la précision des correspondances de points 2D-3D grâce à la prise en compte de toutes les informations pertinentes du vecteur de caractéristiques. En particulier, la régression à coordonnées multiples de notre forêt permet de faire face à des structures répétitives ambiguës. En outre, notre méthode est robuste pour les changements d'éclairage ainsi que les changements d'échelle et de rotation, car nous utilisons des caractéristiques SURF ou apprises. De plus, nous abordons le défi de l'occlusion partielle en utilisant des caractéristiques éparses (sparse feature) au lieu d'une image entière.

Notre méthode s'avère aussi précise que les méthodes de l'état de l'art. Mais en plus, elle permet d'effectuer une relocalisation de caméra en temps réel.

Relocalization de caméra dans des environnements dynamiques

Les données dynamiques représentent un défi commun pour l'apprentissage automatique ainsi que pour la relocalisation de caméra. Dans ce paragraphe, nous nous concentrons plus particulièrement sur le défi complexe de scènes dynamiques lorsque des objets individuels se déplacent au fil du temps.

Notre principale contribution est de proposer un tout nouvel algorithme d'apprentissage automatique basé sur le principe d'une forêt de régression, qui s'adapte en temps réel au modèle prédictif. Il évolue partiellement au fil du temps sans avoir à se reformer intégralement à partir de zéro. Le processus est basé sur une forêt de régression et nous l'appelons **ARF** (Adaptive Regression Forest). L'idée principale est que l'ARF met à jour en temps réel un sous-ensemble de feuilles qui donne des prédictions incertaines. Il est effectué selon deux originalités principales basées sur la détection et la mise à jour des feuilles passives. La première originalité de notre ARF est de détecter les feuilles, qui fournissent des informations non pertinentes. Une feuille doit devenir une feuille passive selon deux critères: avoir une variance élevée du modèle prédictif; donner à plusieurs reprises un résultat assez différent des autres feuilles. La deuxième originalité de notre ARF est de mettre à jour en temps réel des feuilles passives du modèle de forêt de régression. Ceci est effectué en remodelant leur modèle prédictif à partir de nouvelles étiquettes calculées. Ces étiquettes sont calculées à partir de résultats donnés par les autres feuilles (feuilles actives). Notez que la mise à jour est effectuée uniquement sur les feuilles passives et non sur le modèle entier de forêt de régression. Nous illustrons l'efficacité de ce mécanisme sur certains exemples d'application de la relocalisation de caméra.

Ensuite, nous appliquons notre ARF à la relocalisation de caméra en temps réel dans des scènes dynamiques, où certaines parties de la scène bougent ou dans lesquelles la

scène entière change progressivement. Pour cette application, l'ARF prédit des points 3D dans le système de coordonnées de la scène, qui correspondent à des points 2D dans l'image. L'originalité est de conserver la structure de la forêt (arbres et nœuds) en utilisant la caractéristique SURF invariante aux nœuds internes, comme proposé dans nos travaux précédents. En effet, lorsque des objets se déplacent, les descripteurs des points clés détectés restent presque inchangés. Seule leur étiquette, à savoir la position 3D des points clés dans le système de coordonnées de la scène varie.

Des expériences montrent que notre méthode permet d'obtenir des résultats aussi précis que les meilleures méthodes de l'état de l'art sur des ensembles de données de scènes statiques, mais qu'elle est plus rapide à la fois en termes d'entraînement et de test. Notre DynaLoc est robuste aux changements d'occlusion et d'éclairage. De plus, nous obtenons également des résultats plus précis même sur notre ensemble de données de scènes dynamiques en évitant l'accumulation d'erreurs.

Conclusions

Dans cette thèse, nous avons étudié les méthodes de relocalisation de caméra basées sur des images. Tout d'abord, nous avons commencé par présenter la théorie de la vision par ordinateur et l'apprentissage en profondeur pour la relocalisation de caméra. Nous avons également décrit des méthodes de l'état de l'art en relocalisation de caméras par différentes approches. Nous avons ensuite introduit des ensembles de données de relocalisation de caméras.

Les objectifs de cette thèse étaient de proposer une relocalisation de caméra précise et en temps réel à partir d'images RVB dans des environnements dynamiques. L'évolution de notre contribution est résumée dans le Tableau 5.1. À partir de notre dernière contribution DynaLoc, nous avons développé un ***Smart AR Toolbox* [Instant LeARning]** à **IRT b-com**, voir plus de détails en Annexe A.

Bien que nos travaux fournissent un composant de relocalisation de caméra précis et en temps réel pour les systèmes de localisation, ils rencontrent encore des problèmes face aux scènes sans texture et à grande échelle. Cela suggère des pistes de recherche pour l'avenir afin d'améliorer encore la relocalisation de caméra.

Publications

International Journal Papers

- **Nam-Duong Duong**, Catherine Soladie, Amine Kacete, Pierre-Yves Richard, Jérôme Royan, *Efficient multi-output scene coordinate prediction for fast and accurate camera relocalization from a single RGB image*, Computer Vision and Image Understanding, 2019.

International Conference Papers

- **Nam-Duong Duong**, Amine Kacete, Catherine Soladie, Pierre-Yves Richard, Jérôme Royan, *DynaLoc: Real-Time Camera Relocalization from a Single RGB Image in Dynamic Scenes based on an Adaptive Regression Forest*, 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2020.

- **Nam-Duong Duong**, Amine Kacete, Catherine Sodalie, Pierre-Yves Richard, Jérôme Royan, *xyzNet: Towards Machine Learning Camera Relocalization by Using a Scene Coordinate Prediction Network*, In IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), pp. 258-263, Munich, Germany, 2018.

- **Nam-Duong Duong**, Amine Kacete, Catherine Soladie, Pierre-Yves Richard, Jérôme Royan, *Accurate Sparse Feature Regression Forest Learning for Real-Time Camera Relocalization*, In IEEE International Conference on 3D Vision (3DV), pp. 643-652, Verona, Italy, 2018.

Demonstration Papers

- **Nam-Duong Duong**, Amine Kacete, Catherine Soladie, Pierre-Yves Richard, Jérôme Royan, *Online Sparse Scene Coordinates Learning for Real-Time Camera Relocalization*, In IEEE International Conference on 3D Vision (3DV), Verona, Italy, 2018.

National Communications

- **Nam-Duong Duong**, Catherine Soladie, Amine Kacete, Pierre-Yves Richard, Jérôme Royan, *Forêt de Régression Précise basée sur des Caractéristiques Éparses pour la Relocalisation de Caméra en Temps-Réel*, GRETSI, Lille, France, 2019.

- **Nam-Duong Duong**, Amine Kacete, Catherine Soladie, Pierre-Yves Richard, Jérôme Royan, *Relocalisation Robuste de Caméra en Temps Réel pour la Réalité Augmentée par une Approche Hybride combinant Réseaux de Neurones et Méthodes Géométriques*, Dans le congrès Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP), Marne-la-Vallée, France, 2018.

Patents

- **Nam-Duong Duong**, Amine Kacete, Catherine Soladie. *Method for Estimating The Installation of a Camera in The Reference Frame of a Three-Dimensional Scene, Device, Augmented Reality System and Associated Computer Program.* Patent WO2019091787. May 16, 2019.

- **Nam-Duong Duong**, Amine Kacete, Catherine Soladie. *Procédé de prédiction d'une représentation en trois dimensions (3D), Dispositif, Système et Programme d'ordinateur correspondant*. Patent FR1873626.

Appendix A

Smart AR Toolbox [Instant LeARning]

Fig. A.1 Our demo at the IBC exhibition 2018

The work of this thesis has been put up to use in **Smart AR Toolbox**, which is a toolbox of AR computer vision developed in the Institute of Research and Technology b-com. The **Instant LeARning** component is based on our **DynaLoc** algorithm. In order to help understanding the value of this component, we use it to develop an application to display a problem on a data center, as shown in Figure A.1. This demo showcases how our AR core technology is solving critical problems whenever AR has to be used for maintenance applications, considering conditions of real operation and not those of lab environments.

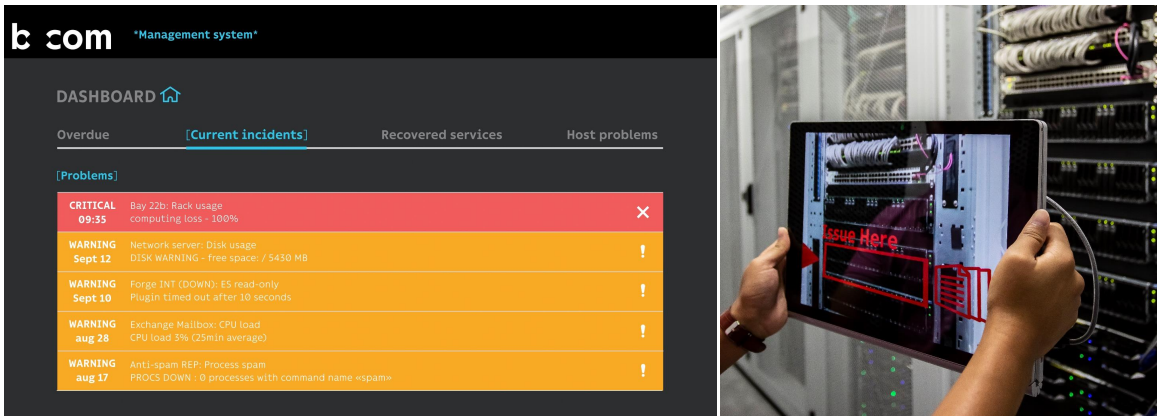


Fig. A.2 A data center infrastructure management combining with our AR.

The broadcast industry, as this is the case for other industries, is significantly migrating from specialized equipment towards more IP and IT based infrastructures or data center and inevitably more software. While those infrastructures rely on commodities equipment, the detection and resolution of problems become more complex. Typically a functional service view in relationship to appliances and wiring views becomes less obvious. AR can definitely help improve this. A study from Ponemon Institute about data center outages shows that:

- The average cost of a data center outage rose to $750k in 2016, up 38% since 2010.
- Cost per minute rose to $9k, it was $5k in 2010, $7k in 2015.
- Downtime costs for the most data center dependent businesses are rising faster than average.

Data Center Infrastructure Management (DCIM) applications aim at helping maintenance team concretely dealing with outages and troubleshooting but under stress conditions. Figure A.2 shows the DCIM system with our **Smart AR Toolbox [Instant LeARning]** which offers a kind of digital twin of infrastructure. When a data center has an issue, this issue is notified on the DCIM. With our AR toolbox, contextual DCIM information can be displayed on a rack allowing the operator to immediately identify what needs to be done to maintain the data center.

Our application is a real-time and accurate camera relocalization in dynamic environments using only RGB images for both training and testing. It is developed by Unity3D and is easy to use by an interactive interface, as shown in Figure A.3 (on right side of the right image), including six buttons: training, stop, relocalization, inliers display, outliers display, quit. At beginning, our application requires a training step which is performed based on fiducial marker as shown in Figure A.3 (the left image): First, place the marker in a position

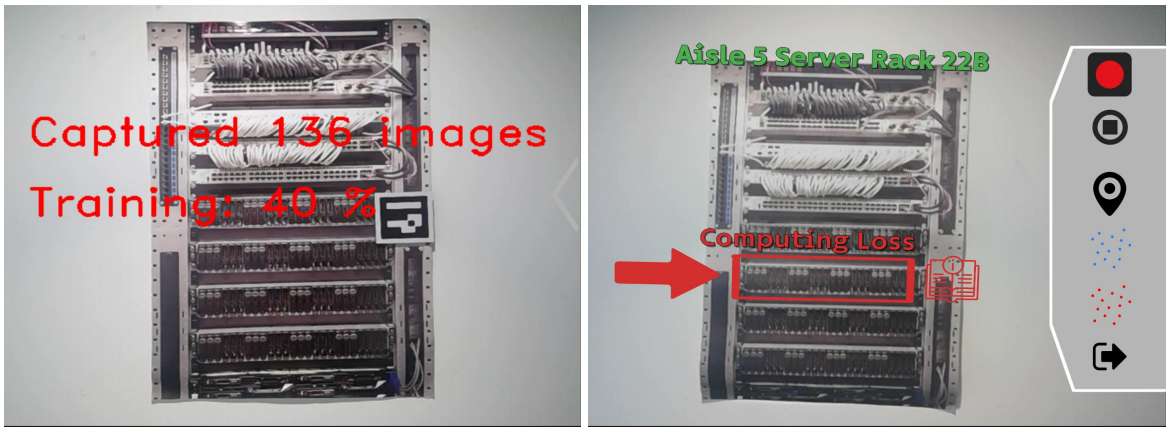


Fig. A.3 Our AR application for maintaining data center: Training phase based on a fiducial marker (on the left) and running phase (on the right).

marked on the data center. Then, move the camera to capture different view points around the fiducial marker. Finally, the learning process automatically stops when it obtains enough information about the data center. This step takes about two minutes. Now we can use the learned model in the running phase to determine the location of issues, as illustrated in Figure A.3 (the right image).

References

[1] Agarwal, S., Furukawa, Y., Snavely, N., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Reconstructing rome. *Computer*, 43(6):40–47.

[2] Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building rome in a day. In *2009 IEEE 12th international conference on computer vision*, pages 72–79. IEEE.

[3] Alcantarilla, P. F., Bartoli, A., and Davison, A. J. (2012a). Kaze features. In *European Conference on Computer Vision*, pages 214–227. Springer.

[4] Alcantarilla, P. F., Yebes, J. J., Almazán, J., and Bergasa, L. M. (2012b). On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1290–1297. IEEE.

[5] Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307.

[6] Arandjelovic, R. and Zisserman, A. (2013). All about vlad. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1578–1585.

[7] Arth, C., Wagner, D., Klopschitz, M., Irschara, A., and Schmalstieg, D. (2009). Wide area localization on mobile phones. In *2009 8th ieee international symposium on mixed and augmented reality*, pages 73–82. IEEE.

[8] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923.

[9] Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385.

[10] Babenko, A. and Lempitsky, V. (2015). Aggregating local deep features for image retrieval. In *The IEEE International Conference on Computer Vision (ICCV)*.

[11] Balntas, V., Li, S., and Prisacariu, V. (2018). Relocnet: Continuous metric learning relocalisation using neural nets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 751–767.

[12] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359.

[13] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

[14] Beis, J. S. and Lowe, D. G. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *cvpr*, volume 97, page 1000. Citeseer.

[15] Bescos, B., Facil, J. M., Civera, J., and Neira, J. (2018). Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076—-4083.

[16] Bleser, G. and Stricker, D. (2009). Advanced tracking through efficient image processing and visual–inertial sensor fusion. *Computers & Graphics*, 33(1):59–72.

[17] Bottou, L. (2010). *Large-scale machine learning with stochastic gradient descent*. Springer.

[18] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[19] Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., and Rother, C. (2014). Learning 6d object pose estimation using 3d object coordinates. In *European Conference on Computer Vision*, pages 536–551. Springer.

[20] Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., and Rother, C. (2017). Dsac - differentiable ransac for camera localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] Brachmann, E., Michel, F., Krull, A., Yang, M. Y., Gumhold, S., and Rother, C. (2016a). Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Conference on Computer Vision and Pattern Recognition*.

[22] Brachmann, E., Michel, F., Krull, A., Ying Yang, M., Gumhold, S., et al. (2016b). Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3364–3372.

[23] Brachmann, E. and Rother, C. (2018). Learning less is more-6d camera localization via 3d surface regression. In *Proc. CVPR*, volume 8.

[24] Bradski, G. (2000). The opencv library. *Dr Dobb's J. Software Tools*, 25:120–125.

[25] Brahmbhatt, S., Gu, J., Kim, K., Hays, J., and Kautz, J. (2018). Geometry-aware learning of maps for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625.

[26] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[27] Bui, M., Albarqouni, S., Ilic, S., and Navab, N. (2018). Scene coordinate and correspondence learning for image-based localization. In *BMVC*, page 3.

[28] Cai, M., Shen, C., and Reid, I. D. (2018). A hybrid probabilistic model for camera relocalization. In *BMVC*.

[29] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer.

[30] Camposeco, F., Cohen, A., Pollefeys, M., and Sattler, T. (2018). Hybrid camera pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 136–144.

[31] Cavallari, T., Golodetz, S., Lord, N. A., Valentin, J., Di Stefano, L., and Torr, P. H. S. (2017). On-the-fly adaptation of regression forests for online camera relocalisation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[32] Clark, R., Wang, S., Markham, A., Trigoni, N., and Wen, H. (2017). Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[33] Criminisi, A. and Shotton, J. (2013). *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media.

[34] Cummins, M. and Newman, P. (2011). Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123.

[35] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067.

[36] Dementhon, D. F. and Davis, L. S. (1995). Model-based object pose in 25 lines of code. *International journal of computer vision*, 15(1-2):123–141.

[37] Donoser, M. and Schmalstieg, D. (2014). Discriminative feature-to-point matching in image-based localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 516–523.

[38] Engel, J., Koltun, V., and Cremers, D. (2017). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[39] Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer.

[40] Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 617–624. IEEE.

[41] Faugeras, O. and FAUGERAS, O. A. (1993). *Three-dimensional computer vision: a geometric viewpoint*. MIT press.

[42] Feng, Y., Fan, L., and Wu, Y. (2016). Fast localization in large-scale environments using supervised indexing of binary features. *IEEE Transactions on Image Processing*, 25(1):343–358.

[43] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[44] Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226.

[45] Gal, Y. (2016). *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge.

[46] Gall, J. and Lempitsky, V. (2013). Class-specific hough forests for object detection. In *Decision forests for computer vision and medical image analysis*, pages 143–157. Springer.

[47] Gall, J., Yao, A., Razavi, N., Van Gool, L., and Lempitsky, V. (2011). Hough forests for object detection, tracking, and action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2188–2202.

[48] Gálvez-López, D. and Tardós, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197.

[49] Gao, X.-S., Hou, X.-R., Tang, J., and Cheng, H.-F. (2003). Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943.

[50] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448.

[51] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[52] Glocker, B., Shotton, J., Criminisi, A., and Izadi, S. (2015). Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding. *IEEE transactions on visualization and computer graphics*, 21(5):571–583.

[53] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer.

[54] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[55] Gordo, A., Almazan, J., Revaud, J., and Larlus, D. (2017). End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254.

[56] Gordoa, A., Rodríguez-Serrano, J. A., Perronnin, F., and Valveny, E. (2012). Leveraging category-level labels for instance-level image retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3045–3052. IEEE.

[57] Gupta, S., Arbeláez, P., Girshick, R., and Malik, J. (2015). Aligning 3d models to rgb-d images of cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4731–4740.

[58] Gupta, S., Girshick, R., Arbeláez, P., and Malik, J. (2014). Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer.

[59] Guzman-Rivera, A., Kohli, P., Glocker, B., Shotton, J., Sharp, T., Fitzgibbon, A., and Izadi, S. (2014). Multi-output learning for camera relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1114–1121.

[60] Haralick, R. M., Lee, D., Ottenburg, K., and Nolle, M. (1991). Analysis and solutions of the three point perspective pose estimation problem. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 592–598. IEEE.

[61] Hartley, R. and Zisserman, A. (2005). Multiple view geometry in computer vision. *Robotica*, 23(2):271–271.

[62] Hartley, R. I. and Sturm, P. (1997). Triangulation. *Computer vision and image understanding*, 68(2):146–157.

[63] Hays, J. and Efros, A. A. (2008). Im2gps: estimating geographic information from a single image. In *2008 ieee conference on computer vision and pattern recognition*, pages 1–8. IEEE.

[64] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[65] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 282–290. AUAI Press.

[66] Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. *Josa a*, 4(4):629–642.

[67] Horn, B. K., Hilden, H. M., and Negahdaripour, S. (1988). Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135.

[68] Huang, C., Ding, X., and Fang, C. (2010). Head pose estimation based on random forests for multiclass classification. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 934–937. IEEE.

[69] Huang, J., Shao, X., and Wechsler, H. (1998). Face pose discrimination using support vector machines (svm). In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 154–156. IEEE.

[70] Huynh, D. Q. (2009). Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164.

[71] Irschara, A., Zach, C., Frahm, J.-M., and Bischof, H. (2009). From structure-from-motion point clouds to fast location recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2599–2606. IEEE.

[72] Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311. IEEE Computer Society.

[73] Jegou, H., Perronnin, F., Douze, M., Sánchez, J., Perez, P., and Schmid, C. (2012). Aggregating local image descriptors into compact codes. *IEEE transactions on pattern analysis and machine intelligence*, 34(9):1704–1716.

[74] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[75] Jiang, N., Cui, Z., and Tan, P. (2013). A global linear method for camera pose registration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 481–488.

[76] Jin Kim, H., Dunn, E., and Frahm, J.-M. (2015). Predicting good features for image geo-localization using per-bundle vlad. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1170–1178.

[77] Kabsch, W. (1976). A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923.

[78] Kacete, A., Royan, J., Seguier, R., Collobert, M., and Soladie, C. (2016). Real-time eye pupil localization using hough regression forest. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–8. IEEE.

[79] Kacete, A., Wentz, T., and Royan, J. (2017). [poster] decision forest for efficient and robust camera relocalization. In *Mixed and Augmented Reality (ISMAR-Adjunct), 2017 IEEE International Symposium on*, pages 20–24. IEEE.

[80] Kehl, W., Milletari, F., Tombari, F., Ilic, S., and Navab, N. (2016). Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European Conference on Computer Vision*, pages 205–220. Springer.

[81] Kendall, A. and Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization. *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

[82] Kendall, A. and Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[83] Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2938–2946.

[84] Kim, H. J., Dunn, E., and Frahm, J.-M. (2017). Learned contextual feature reweighting for image geo-localization. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3251–3260. IEEE.

[85] Klein, G. and Drummond, T. (2004). Sensor fusion and occlusion refinement for tablet-based ar. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 38–47. IEEE Computer Society.

[86] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE.

[87] Klein, G. and Murray, D. (2008). Improving the agility of keyframe-based slam. In *European Conference on Computer Vision*, pages 802–815. Springer.

[88] Kouskouridas, R., Tejani, A., Doumanoglou, A., Tang, D., and Kim, T.-K. (2016). Latent-class hough forests for 6 dof object pose estimation. *arXiv preprint arXiv:1602.01464*.

[89] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[90] Krull, A., Brachmann, E., Michel, F., Ying Yang, M., Gumhold, S., and Rother, C. (2015). Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 954–962.

[91] Krull, A., Michel, F., Brachmann, E., Gumhold, S., Ihrke, S., and Rother, C. (2014). 6-dof model based tracking via object coordinate regression. In *Asian Conference on Computer Vision*, pages 384–399. Springer.

[92] Kwon, J. and Lee, K. M. (2010). Monocular slam with locally planar landmarks via geometric rao-blackwellized particle filtering on lie groups. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1522–1529. IEEE.

[93] Kwong, J. N. S. and Gong, S. (1999). Learning support vector machines for a multi-view face model. In *BMVC*, pages 1–10. Citeseer.

[94] Laskar, Z., Melekhov, I., Kalia, S., and Kannala, J. (2017). Camera relocalization by computing pairwise relative poses using convolutional neural network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 929–938.

[95] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

[96] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[97] Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE transactions on pattern analysis and machine intelligence*, 28(9):1465–1479.

[98] Lepetit, V., Fua, P., et al. (2005). Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision*, 1(1):1–89.

[99] Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155.

[100] Li, R., Wang, S., Long, Z., and Gu, D. (2018a). Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291. IEEE.

[101] Li, S. Z., Fu, Q., Gu, L., Scholkopf, B., Cheng, Y., and Zhang, H. (2001). Kernel machine based learning for multi-view face detection and pose estimation. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 674–679. IEEE.

[102] Li, X., Ylioinas, J., and Kannala, J. (2018b). Full-frame scene coordinate regression for image-based localization. In *RSS*.

[103] Li, X., Ylioinas, J., Verbeek, J., and Kannala, J. (2018c). Scene coordinate regression with angle-based reprojection loss for camera relocalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0.

[104] Li, Y., Gong, S., Sherrah, J., and Liddell, H. (2004). Support vector machine based multi-view face detection and recognition. *Image and Vision Computing*, 22(5):413–427.

[105] Li, Y., Snavely, N., Huttenlocher, D., and Fua, P. (2012). Worldwide pose estimation using 3d point clouds. In *European conference on computer vision*, pages 15–29. Springer.

[106] Li, Y., Snavely, N., and Huttenlocher, D. P. (2010). Location recognition using prioritized feature matching. In *European Conference on Computer Vision*, pages 791–804. Springer.

[107] Lieberknecht, S., Huber, A., Ilic, S., and Benhimane, S. (2011). Rgb-d camera-based parallel tracking and meshing. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 147–155. IEEE.

[108] Liu, L., Li, H., and Dai, Y. (2017). Efficient global 2d-3d matching for camera localization in a large-scale 3d map. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2372–2381.

[109] Liu, T., Moore, A. W., Yang, K., and Gray, A. G. (2005). An investigation of practical approximate nearest neighbor algorithms. In *Advances in neural information processing systems*, pages 825–832.

[110] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[111] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

[112] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[113] Malyavej, V., Torteeka, P., Wongkharn, S., and Wiangtong, T. (2009). Pose estimation of unmanned ground vehicle based on dead-reckoning/gps sensor fusion by unscented kalman filter. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, volume 1, pages 395–398. IEEE.

[114] Massa, F., Marlet, R., and Aubry, M. (2016). Crafting a multi-task cnn for viewpoint estimation. *BMVC*.

[115] Massiceti, D., Krull, A., Brachmann, E., Rother, C., and Torr, P. H. (2017). Random forests versus neural networks—what's best for camera localization? In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5118–5125. IEEE.

[116] Melekhov, I., Ylioinas, J., Kannala, J., and Rahtu, E. (2017). Image-based localization using hourglass networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 879–886.

[117] Meng, L., Chen, J., Tung, F., Little, J. J., and de Silva, C. W. (2016). Exploiting random rgb and sparse features for camera pose estimation. In *BMVC*.

[118] Meng, L., Chen, J., Tung, F., Little J., J., Valentin, J., and Silva, C. (2017). Backtracking regression forests for accurate camera relocalization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017)*.

[119] Meng, L., Tung, F., Little, J. J., Valentin, J., and de Silva, C. W. (2018). Exploiting points and lines in regression forests for rgb-d camera relocalization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6827–6834. IEEE.

[120] Michel, F., Krull, A., Brachmann, E., Yang, M. Y., Gumhold, S., and Rother, C. (2015). Pose estimation of kinematic chain instances via object coordinate regression. In *Proc. British Machine Vision Conf*, pages 181–1.

[121] Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. (1995). Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, volume 2351, pages 282–293. International Society for Optics and Photonics.

[122] Moulon, P., Monasse, P., and Marlet, R. (2013). Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3248–3255.

[123] Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2.

[124] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[125] Mur-Artal, R. and Tardós, J. D. (2014). Fast relocalisation and loop closing in keyframe-based slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 846–853. IEEE.

[126] Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

[127] Murphy-Chutorian, E., Doshi, A., and Trivedi, M. M. (2007). Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 709–714. IEEE.

[128] Naseer, T. and Burgard, W. (2017). Deep regression for monocular camera-based 6-dof global localization in outdoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1525–1530. IEEE.

[129] Newcombe, R. A., Fox, D., and Seitz, S. M. (2015). Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352.

[130] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE.

[131] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011b). Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE.

[132] Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. Ieee.

[133] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175.

[134] Peasley, B. and Birchfield, S. (2015). Rgbd point cloud alignment using lucas–kanade data association and automatic error metric selection. *IEEE Transactions on Robotics*, 31(6):1548–1554.

[135] Perronnin, F., Liu, Y., Sánchez, J., and Poirier, H. (2010). Large-scale image retrieval with compressed fisher vectors. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3384–3391. IEEE.

[136] Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., and Koch, R. (2004). Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232.

[137] Pupilli, M. and Calway, A. (2005). Real-time camera tracking using a particle filter. In *BMVC*.

[138] Quan, L. and Lan, Z. (1999). Linear n-point camera pose determination. *IEEE Transactions on pattern analysis and machine intelligence*, 21(8):774–780.

[139] Radwan, N., Valada, A., and Burgard, W. (2018). Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, 3(4):4407–4414.

[140] Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer.

[141] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

[142] Riazuelo, L., Montano, L., and Montiel, J. (2017). Semantic visual slam in populated environments. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–7. IEEE.

[143] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer.

[144] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. R. (2011). Orb: An efficient alternative to sift or surf. In *ICCV*, page 2. Citeseer.

[145] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[146] Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *3dim*, volume 1, pages 145–152.

[147] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

[148] Sattler, T., Havlena, M., Radenovic, F., Schindler, K., and Pollefeys, M. (2015). Hyperpoints and fine vocabularies for large-scale location recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2102–2110.

[149] Sattler, T., Leibe, B., and Kobbelt, L. (2011). Fast image-based localization using direct 2d-to-3d matching. In *2011 International Conference on Computer Vision*, pages 667–674. IEEE.

[150] Sattler, T., Leibe, B., and Kobbelt, L. (2012). Improving image-based localization by active correspondence search. In *European conference on computer vision*, pages 752–765. Springer.

[151] Sattler, T., Leibe, B., and Kobbelt, L. (2017). Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1744–1756.

[152] Schmalstieg, D. and Hollerer, T. (2016). *Augmented reality: principles and practice*. Addison-Wesley Professional.

[153] Schwarz, M., Schulz, H., and Behnke, S. (2015). Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1329–1335. IEEE.

[154] Seemann, E., Nickel, K., and Stiefelhagen, R. (2004). Head pose estimation using stereo vision for human-robot interaction. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 626–631. IEEE.

[155] Sharif Razavian, A., Sullivan, J., Maki, A., and Carlsson, S. (2015). A baseline for visual instance retrieval with deep convolutional networks. In *International Conference on Learning Representations, May 7-9, 2015, San Diego, CA*. ICLR.

[156] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013a). Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937.

[157] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013b). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

[158] Silpa-Anan, C. and Hartley, R. (2008). Optimised kd-trees for fast image descriptor matching. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[159] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

[160] Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE.

[161] Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM.

[162] Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.

[163] Sturm, P. F. and Maybank, S. J. (1999). On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 432–437. IEEE.

[164] Su, H., Qi, C. R., Li, Y., and Guibas, L. J. (2015). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694.

[165] Sun, Y., Liu, M., and Meng, M. Q.-H. (2017). Improving rgb-d slam in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems*, 89:110–122.

[166] Svärm, L., Enqvist, O., Kahl, F., and Oskarsson, M. (2017). City-scale localization for cameras with known vertical direction. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1455–1461.

[167] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[168] Taira, H., Okutomi, M., Sattler, T., Cimpoi, M., Pollefeys, M., Sivic, J., Pajdla, T., and Torii, A. (2018). Inloc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209.

[169] Tan, W., Liu, H., Dong, Z., Zhang, G., and Bao, H. (2013). Robust monocular slam in dynamic environments. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 209–218. IEEE.

[170] Tateno, K., Tombari, F., Laina, I., and Navab, N. (2017). Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[171] Torii, A., Arandjelovic, R., Sivic, J., Okutomi, M., and Pajdla, T. (2015). 24/7 place recognition by view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1808–1817.

[172] Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660.

[173] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer.

[174] Tsai, R. Y. and Lenz, R. K. (1988). Real time versatile robotics hand/eye calibration using 3d machine vision. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 554–561. IEEE.

[175] Valada, A., Radwan, N., and Burgard, W. (2018). Deep auxiliary learning for visual localization and odometry. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6939–6946. IEEE.

[176] Valentin, J., Nießner, M., Shotton, J., Fitzgibbon, A., Izadi, S., and Torr, P. H. (2015). Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4400–4408.

[177] Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., and Cremers, D. (2017). Image-based localization using lstms for structured feature correlation. In *The IEEE International Conference on Computer Vision (ICCV)*.

[178] Wangsiripitak, S. and Murray, D. W. (2009). Avoiding moving outliers in visual slam by tracking moving objects. In *ICRA*, volume 2, page 7.

[179] Wasenmüller, O., Meyer, M., and Stricker, D. (2016). Corbs: Comprehensive rgb-d benchmark for slam using kinect v2. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–7. IEEE.

[180] Weyand, T., Kostrikov, I., and Philbin, J. (2016). Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer.

[181] Whelan, K. F., Kaess, M., Fallon, M. F., Johannsson, H., Leonard, J. J., and McDonald, J. (2012). Kintinuous: Spatially extended kinectfusion. In *AAAI 2012*.

[182] Whelan, T., Leutenegger, S., Salas-Moreno, R. F., Glocker, B., and Davison, A. J. (2015). Elasticfusion: Dense slam without a pose graph. In *Robotics: science and systems*, volume 11.

[183] Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, page 0278364916669237.

[184] Wilson, K. and Snavely, N. (2014). Robust global translations with 1dsfm. In *European Conference on Computer Vision*, pages 61–75. Springer.

[185] Wohlhart, P. and Lepetit, V. (2015). Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3109–3118.

[186] Wu, C. (2013). Towards linear-time incremental structure from motion. In *3DTV-Conference, 2013 International Conference on*, pages 127–134. IEEE.

[187] Wu, J., Ma, L., and Hu, X. (2017). Delving deeper into convolutional neural networks for camera relocalization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5644–5651. IEEE.

[188] You, S. and Neumann, U. (2001). Fusion of vision and gyro tracking for robust augmented reality registration. In *Proceedings IEEE Virtual Reality 2001*, pages 71–78. IEEE.

[189] Yun, Y., Changrampadi, M. H., and Gu, I. Y. (2014). Head pose classification by multi-class adaboost with fusion of rgb and depth images. In *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, pages 174–177. IEEE.

[190] Zeisl, B., Sattler, T., and Pollefeys, M. (2015). Camera pose voting for large-scale image-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2704–2712.

[191] Zhang, P., Gu, J., Milios, E. E., and Huynh, P. (2005). Navigation with imu/gps/digital compass with unscented kalman filter. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 3, pages 1497–1502. IEEE.

[192] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22.

[193] Zhou, G., Bescos, B., Dymczyk, M., Pfeiffer, M., Neira, J., and Siegwart, R. (2018). Dynamic objects segmentation for visual localization in urban environments. *arXiv preprint arXiv:1807.02996*.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

CentraleSupélec

Title : Hybrid Machine Learning and Geometric Approaches for Single RGB Camera Relocalization

Keywords: Camera relocalization; Deep learning; Random forest; Hybrid method; Scene coordinate.

Abstract:
In the last few years, image-based camera relocalization becomes an important issue of computer vision applied to augmented reality, robotics as well as autonomous vehicles.
Camera relocalization refers to the problematic of the camera pose estimation including both 3D translation and 3D rotation.
In localization systems, camera relocalization component is necessary to retrieve camera pose after tracking lost, rather than restarting the localization from scratch. However, the classical existing camera relocalization methods store a large set of keypoints or keyframes to relocalize camera based geometric information. Consequently, memory usage as well as processing time rise with respect to the size of the models. Accordingly, machine learning approaches have been developed to tackle these constraints. Nevertheless, the limitations of machine learning approaches lie in their time-consuming training process, moderate accuracy and lack of confidence score in the estimation of each pose. Recently, hybrid methods increase considerably the accuracy. Yet, they still take more time to optimize camera pose from thousands of correspondences. Moreover, all these machine learning based methods still fail to challenge dynamic scenes with moving objects.
Given those pros and cons, this thesis aims at improving the performance of camera relocalization in terms of both runtime and accuracy as well as handling challenges of camera relocalization in dynamic environments.
We present camera pose estimation based on combining multi-patch pose regression to overcome the uncertainty of end-to-end deep learning methods. To balance between accuracy and computational time of camera relocalization from a single RGB image, we propose a sparse feature hybrid methods. A better prediction in the machine learning part of our methods leads to a rapid inference of camera pose in the geometric part.
To tackle the challenge of dynamic environments, we propose an adaptive regression forest algorithm that adapts itself in real time to predictive model. It evolves by part over time without requirement of re-training the whole model from scratch. When applying this algorithm to our real-time and accurate camera relocalization, we can cope with dynamic environments, especially moving objects.
The experiments proves the efficiency of our proposed methods. Our method achieves results as accurate as the best state-of-the-art methods on the rigid scenes dataset. Moreover, we also obtain high accuracy even on the dynamic scenes dataset.