Robust 3D Object Pose Estimation and Tracking from Monocular Images in Industrial Environments

THÈSE N^{O} 7282 (2016)

PRÉSENTÉE LE 18 NOVEMBRE 2016

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE VISION PAR ORDINATEUR
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Alberto CRIVELLARO

acceptée sur proposition du jury:

Prof. P. Dillenbourg, président du jury
Prof. P. Fua, Prof. V. Lepetit, directeurs de thèse
Prof. R. Cipolla, rapporteur
Prof. C. Rother, rapporteur
Dr R. Boulic, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2016

Τί δύσκολον· Τὸ ἑαυτὸν γνῶναι.
Thales, 625-546 BC

Abstract

Recent advances in Computer Vision are changing our way of living and enabling new applications for both leisure and professional use, ranging from games based on Augmented Reality to automated diagnostic tools based on the analysis of medical imagery. Regrettably, in many industrial domains the spread of state-of-the-art technologies is made challenging by the abundance of nuisances that corrupt existing techniques beyond the required dependability.

This is especially true for object localization and tracking, that is, the problem of detecting the presence of objects on images and videos and estimating their pose. This is a critical task for applications such as Augmented Reality (AR), robotic autonomous navigation, robotic object grasping, or production quality control; unfortunately, the reliability of existing techniques is harmed by the visual features encountered in many industrial environments, such as the abundance of specular and poorly textured objects, cluttered scenes, or artificial and in-homogeneous lighting.

In this thesis, we propose two methods for robustly estimating the pose of a rigid object under the challenging conditions typical of industrial environments. Both methods rely on monocular images to handle metallic environments, on which depth cameras would fail; both are conceived with a limited computational and memory footprint, so that they are suitable for real-time applications such as AR. We test our methods on datasets issued from real user case scenarios, exhibiting challenging conditions.

The first method is based on a global image alignment framework and a robust dense descriptor. Its global approach makes it robust in presence of local artifacts such as specularities appearing on metallic objects, ambiguous patterns like screws or wires, and poorly textured objects. Employing a global approach avoids the need of reliably detecting and matching local features across images, that become ill-conditioned tasks in the considered environments; on the other hand, current methods based on dense image alignment usually rely on luminous intensities for comparing the pixels, which is not robust in presence of challenging illumination artifacts. We show how the use of a dense descriptor computed as a non-linear function of luminous intensities, that we refer to as "Descriptor Fields", greatly enhances performances at a minimal computational overhead. Among others, we show the effectiveness of our Descriptor Fields over several optimization schemes and distance metrics. Their low computational complexity and their ease of implementation make Descriptor Fields suitable for replacing intensities in a wide number of state-of-the-art techniques based on dense image alignment.

Relying on a global approach is appropriate for overcoming local artifacts, but it can be un-effective when the target object undergoes extreme occlusions in cluttered environments. For this reason, we propose a second approach based on the detection of discriminative object parts. At the core of our approach is a novel representation for the 3D pose of the parts, that allows us to predict the 3D pose of the object even when only a single part is visible; when several parts are visible, we can easily combine them to compute a better pose of the object. The 3D pose we obtain is usually very accurate, even when only few parts are visible. We show how to use this representation in a robust 3D tracking framework. In addition to extensive comparisons with the state-of-the-art, we demonstrate our method on a practical Augmented Reality application for maintenance assistance in the ATLAS particle detector at CERN.

Key words: Computer Vision, 3D Detection, 3D Tracking, Rigid Pose Estimation, Augmented Reality

Résumé

Les récentes avancées dans le domaine de la Vision Assistée par Ordinateur sont en train de changer notre façon de vivre ; elles rendent disponibles des nouvelles applications dans la sphère professionnelle aussi bien que privée, à partir des jeux basés sur la Réalité Augmentée jusqu'aux outils de diagnostic basés sur l'analyse d'imagerie médicale.

Malheureusement, dans de nombreux domaines industriels la diffusion des dernières technologies est freinée par l'abondance de nuisances qui dégradent les performances des méthodes actuelles au delà du degré de fiabilité requis.

Cela est vrai aussi, en particulier, pour la localisation et le suivi objets, c'est-à-dire le problème de détecter la présence d'objets sur des images et des vidéos et d'estimer leur pose. Il s'agit d'une étape cruciale dans des nombreuses applications, telles que la Réalité Augmentée (AR), la navigation autonome et la saisie d'objets par des robots, ou le contrôle de qualité sur des produits. Malheureusement, l'efficacité des approches existantes est limitée par des nombreuses caractéristiques typiques des environnements industriels, dont l'abondance d'objets spéculaires ou non-texturés, de scènes encombrées, d'une illumination artificielle et non-homogène.

Dans cette thèse, nous proposons deux méthodes pour estimer de façon robuste la pose d'un objet rigide en temps réel dans les conditions extrêmes typiques des environnements industriels.

Les deux méthodes utilisent des images monoculaires pour être robustes en présence d'objet métalliques où des senseurs de profondeur ne marcheraient pas ; les deux demandent une quantité limitée de ressources de calcul, ce qui les rend utilisables pour des applications en temps réel comme la Réalité Augmentée. Nous testons nos méthodes sur des bases de test représentatives des conditions réelles.

La première méthode est basée sur une technique d'alignement d'images et un descripteur robuste. Son approche globale la rend robuste en présence d'artefacts tels que des spécularités qui apparaissent sur des objets métalliques, détails ambigus comme des vis ou des câbles, et des objets non texturés.

En utilisant une approche globale nous évitons de détecter explicitement et mettre en correspondance des zones d'intérêt locales dans les images, une tâche extrêmement mal conditionnée dans les environnements considérés. Les méthodes courantes basées sur l'alignement dense d'images utilisent l'intensité pour comparer les pixels, ce qui le rend fragiles en présence d'artéfacts issues de l'illumination. Nous montrons comment l'utilisation d'un descripteur dense, calculé avec une transformation non-linéaire des intensités, que nous appelons "Descriptor Fields" , améliore sensiblement les performances avec un faible surcoût computationnel. Notamment, nous montrons l'efficacité de nos Descriptor Fields avec différents schémas d'optimisation et distances. Leur faible coût de calcul et leur facilité d'implémentation les rendent adaptés pour remplacer les intensités dans des nombreuses méthodes courantes basées sur l'alignement d'images dense.

Utiliser une approche globale est approprié pour être robuste en présence d'artefacts locaux, mais il peut s'avérer peu efficace quand l'objet est masqué par des très vastes occlusions. Pour cette raison, nous proposons une deuxième approche basée sur la détection de parties de l'objet. Notre approche repose sur une nouvelle représentation pour la pose en 3D des parties, qui nous permet de prédire la pose 3D de l'objet aussi quand une seule partie est visible ; quand plusieures parties sont détectées sur la même image, les estimations pour chaque parties sont facilement combinées pour prédire la pose de l'objet de façon plus précise. Normalement, la pose calculée est très précise, même quand seulement peu de parties sont visibles. Nous montrons comment utiliser cette représentation dans un système robuste pour le suivi d'objets. En plus de comparaisons extensives avec l'état de l'Art, nous présentons l'application de notre méthode pour un vrai cas test, un outil de Réalité Augmentée pour l'assistance pendant des interventions techniques dans le détecteur de particules ATLAS au CERN.

Mots clefs : Vision Assistée par Ordinateur, Détection 3D , Suivi d'Objets, Estimation de Pose d'Objets Rigides, Réalité Augmentée

Acknowledgements

First of all, I would like to thank my supervisor, Prof. Pascal Fua, for giving me the opportunity to work at CVLab, and for the guidance he provided to me, not only as a creative researcher, but also as a passionate teacher and an effective manager. I would also like to express my gratitude to Prof. Pierre Dillenbourg, Dr. Ronan Boulic, Prof. Roberto Cipolla, and Prof. Carsten Rother, for accepting to evaluate this thesis and for their insightful advice.

My deepest gratitude goes to my co-supervisor, prof. Vincent Lepetit. With his enthusiasm, his enormous commitment, his creativity, his exceptional human and technical skills, he taught me lessons that go far beyond Computer Vision, and has been a constant source of inspiration for my work.

I am also very grateful to Josiane: during all these years, her perfect sense of organization was only second to her patience coping with me and my clumsiness. I wish her all the best for her new life, and good luck to Ariane for her up-coming work at CVLab.

During the time of my application, I randomly picked a name in the list of the future colleagues and wrote to him for information about *"this new job in Switzerland I could apply for"*. Since that day, Roberto has been a guidance and source of inspiration for research, computer science, the mountains and original life styles.

My special thanks goes to him and my colleagues at EPFL and CVLab.

To Amos, Marco and Jean Louis, for all the fun moments spent together. Working on different topics and having different interests did not prevent us from sharing good memory. It lead us to passionate discussions about topics ranging from techniques for growing vegetables to methods for alleviating hangovers and best gears for alpine skiing.

To Yannick, with whom I shared the most tragic and epic moments of the EDUSAFE project, ranging from extenuating organizational meetings to adventurous climbing expeditions. Thanks to our great team work, we could survive both. To Kwang, my favorite Linux guru, and Xinchao, for their support and their advice (including proofreading these pages). To Tomasz, Carlos, Dat, Amaury and the other CVLab colleagues for the interesting discussions and their friendly advice. To Olga Beltramello for her endless commitment in coordinating the EDUSAFE project, and to all the EDUSAFE ESRs and ERs for our collaboration and the nice moments we spent together.

I am especially grateful to my favorite music group, Paperboots, aka Maira, Mati and Piotr, for their great music and also for being great flat mates. To Eleonora and Francesco, who already

shared two cities and two universities with me; I hope Lausanne won't be the last one! Last but not least, to Lorenzo, Kri, Giada, Michele, Johnatan, Zoé, Juan, Audrey, Nadia and the other friends *Lausannois*: the smiles and the bits of optimism they transmitted to me in all the good moments spent together contributed to the work presented in this thesis.

Finally, my deepest gratitude goes to my family, Rossella, Mino and Andrea, for their unconditional love, and to my wife Geraldine, for her patience, her support, her constant encouragement and for sharing with me all the good and bad moments, *for better or for worse*. They are all role models to me.

Lausanne, 25 October 2016

Contents

Abstract (English/Français/Deutsch) **i**

Acknowledgements **v**

List of figures **xiii**

List of tables **xvii**

1 3D Tracking for Industrial Applications **1**

1.1 3D Tracking in Industrial Environments: The "Daily Setup" Bias 2

1.2 3D Tracking Applications in Industrial Environments 2

1.2.1 Augmented Reality at CERN: the EDUSAFE Project 4

1.2.2 A Robotic Vision Case Study. Seam Tracking for Industrial TIG Pipe Welding 5

1.3 Contributions 8

1.4 Outline 9

2 3D Tracking : State of the Art **11**

2.1 Related Work 12

2.1.1 Edge-based Methods 12

2.1.2 Keypoints-based Methods 13

2.1.3 Region-based Methods 13

2.1.4 Dense Image Alignment Methods 14

2.1.5 Depth-based Methods 15

2.1.6 Learning-based Methods 15

2.1.7 Deep Learning-based Methods 17

2.1.8 Part-based Methods 17

2.1.9 Large Scale Pose Estimation Methods 18

2.1.10 SLAM 19

2.2 Addressing Challenges in Industrial Environments 20

2.2.1 Non-textured Objects; Ambiguous Patterns 20

2.2.2 Specularities and Illumination Artifacts 20

2.2.3 Occlusions and Clutter 20

2.3 Conclusion 21

3 Mathematical Framework **23**

3.1 Main Notations 23

3.2 Perspective Camera Model 24

3.3 Approximate Camera Models 27

3.4 Distortion Models 29

3.5 Exponential Map Parametrization for 3D Rotations 30

3.6 Conclusion 31

4 Dense Methods for Image Alignment and their Application to 3D Tracking 33

4.1 Dense Image Alignment . . . 34

4.2 Optimization Framework . . . 35

4.3 First Order Methods . . . 35

4.3.1 Forward Additive Algorithm . . . 36

4.3.2 Forward Compositional Algorithm . . . 37

4.3.3 Inverse Compositional Algorithm . . . 39

4.3.4 Inverse Additive Algorithm . . . 41

4.4 A Second-order Method: ESM . . . 43

4.5 Choice of the Appropriate Algorithm. Additive vs Compositional Approach . . . 46

4.6 A Warp for 3D Tracking . . . 48

4.7 Extensions and Related Methods . . . 51

4.8 Conclusion . . . 55

5 Descriptor Fields for 3D Tracking 57

5.1 3D Tracking via Dense Image Alignment . . . 58

5.1.1 Optimization Framework . . . 58

5.1.2 Multi-scale Optimization . . . 59

5.2 Descriptor Fields . . . 60

5.2.1 A 1D Example . . . 61

5.2.2 An Example with Real Images . . . 62

5.3 Experimental Results . . . 63

5.3.1 Datasets . . . 63

5.3.2 Evaluation Framework 64
5.3.3 Evaluation 65
5.3.4 Evaluation of the Distance Function 67
5.3.5 Rotation Invariance 69
5.4 Applications and Further Developments 70
5.5 Conclusion 71

6 Robust 3D Tracking using Stable Parts 77

6.1 Overview of the Method 80
6.2 Part Detection 81
6.3 Part Pose Estimation 84
6.3.1 Representation of the Part Pose 84
6.3.2 Prediction of the Reprojections of the Control Points 85
6.4 Object Pose Estimation 86
6.4.1 Using a Single Gaussian Pose Prior 87
6.4.2 Outlier Rejection for the Detected Parts 88
6.4.3 Using a Mixture-of-Gaussians for the Pose Prior 89
6.4.4 Identifying the Best Pose Estimate 89
6.5 Tracking Frames across a Video Sequence and Pose Filtering 89
6.5.1 Extended Kalman Filter for 3D Tracking 90
6.6 Experimental Results 93
6.6.1 Evaluation Protocol 93
6.6.2 Datasets 93

6.6.3 Part Detection 94

6.6.4 Validation of the Part Pose Representation 96

6.6.5 Virtual Points Configuration 98

6.6.6 Comparison against the State-of-the-Art 100

6.6.7 Training Details 101

6.6.8 Results 101

6.6.9 Runtimes 103

6.7 Applications and Further Developments 103

6.7.1 Articulated Objects 104

6.7.2 Object Tracking and SLAM 105

6.8 Conclusion 106

7 Conclusion **107**

7.1 Future Work 108

A Appendix A: Additional Results for Dense Image Alignment Methods **111**

A.1 An Example of Warp: 2D Rigid Deformation 111

A.2 Equivalence of First-order Methods 113

A.2.1 Equivalence of FA and FC Algorithms 113

A.2.2 Equivalence of FC and IC Algorithms 114

A.3 A Theorem about Groups of Differentiable Warps 115

A.4 Relaxing Hypothesis of ESM 116

A.5 Alignment of Multi-Channel Images 117

A.6 Comparative Tables of Dense Alignment Methods 118

B Appendix B: 3D Tracking with Dense Image Alignment and Different Internal Matrices 121

Bibliography 131

Curriculum Vitae 133

List of Figures

1.1 Examples of the challenging visual conditions encountered in industrial environments 3

1.2 Examples of Applications of 3D Tracking in Industrial Environments 4

1.3 The EDUSAFE Augmented Reality prototype in action. 4

1.4 The Personal Safety Device developed for the EDUSAFE project. 5

1.5 The pipe welding machine in action during an indoor test. 6

1.6 The automatic vision-aided system developed for a welding machine. 7

2.1 Examples of state-of-the art methods: edge-based tracking, Keypoint-based tracking, template matching 14

2.2 Examples of state-of-the art methods: part-based tracking, large scale pose estimation, and SLAM 19

3.1 Perspective projection model 25

3.2 The full perspective projection model and its approximations. 27

4.1 The image alignment problem. 34

4.2 The Forward Additive algorithm. 36

4.3 Updated warp in the Forward Compositional algorithm. 38

4.4 Updated warp provided by the Inverse Compositional algorithm. 39

4.5 Schematic representation of dense image alignment algorithms. 49

4.6 3D warp between a template T with pose $\mathbf{p}_T$, and an image I, with pose $\mathbf{p}_T + \mathbf{p}$. 50

4.7 Inverse warp for 3D tracking. 50

5.1 Robust 3D tracking with Descriptor Fields in a non-Lambertian environment. . . 58

5.2 Iterative alignment of two 1D signals. 62

5.3 Sparsifying action of Descriptor Fields on a 1D example 63

5.4 Different descriptors on a specular surface and corresponding objective function for a translation warp 72

5.5 Examples from the ATLAS dataset 73

5.6 Error thresholds employed for our evaluations 73

5.7 Comparisons between intensities and Descriptor Fields on our Experimental Setup dataset. 73

5.8 Comparisons between intensities and Descriptor Fields on the ATLAS dataset. . . 74

5.9 Some applications of image alignment and Descriptor Fields 74

5.10 Comparison between different distance functions for image alignment. 75

6.1 Our part-based method in action during a demonstrative technical intervention at CERN, Geneva. 78

6.2 Our representation of the 3D pose of an object part. 79

6.3 Main steps of our part-based algorithm. 80

6.4 Part detection pipeline. 83

6.5 Architecture of $\text{CNN}^{\text{part-det}}$ for part detection. The last layer outputs the likelihoods of the patch to correspond to each part or to the background. 83

6.6 Architecture of the CNN $\text{CNN}^{\text{cp-pred-}p}$ predicting the projections of the control points. 86

6.7 Visualisation of the pose prior for an electric box: Projections of the box by each of the 9 Gaussians centers $\overline{\mathbf{p}}_m$. 87

6.8 Qualitative results of our part-based tracking framework on our challenging datasets. 95

6.9 Training images and control points we used for the BOX, the CAN and the DOOR datasets. 95

6.10 Results of the experiment described in Section 6.6.3: detection error Cumulative Distribution Functions (CDF) for the BOX dataset for different detectors. Top row: Video #1. Bottom row: Video #2. 97

6.11 The rotation and translation error Cumulative Distribution Functions (CDF) on the BOX dataset, Video #1 for the parametrizations of the part poses presented in Section 6.6.4. Our pose representation entails a substantial performance gain. . . 99

6.12 Rotation and translation error Cumulative Distribution Functions (CDF) for the configurations of virtual points shown in Figure 6.13 for the CAN dataset-Video #1. 99

6.13 Different configurations of control points tested on the CAN dataset. 100

6.14 The rotation and translation error Cumulative Distribution Functions (CDF) on the BOX dataset. 101

6.15 The rotation and translation error Cumulative Distribution Functions (CDF) on the CAN dataset. 102

6.16 The rotation and translation error Cumulative Distribution Functions (CDF) on the DOOR dataset. 102

6.17 Tracking of articulated objects with our part-based framework. 105

6.18 Integration of our object pipeline and a ORB-SLAM, as described in Section 6.7.2. 106

B.1 Warp between a template T with internal calibration matrix $\mathbf{K}_T$ and pose $\mathbf{p}_T$, and an image I, with pose $\mathbf{p}_T + \mathbf{p}$ and internal calibration matrix $\mathbf{K}_{IM}$. 122

List of Tables

3.1 Main notations employed in this thesis. 24

5.1 Experimental results of comparisons between Descriptor Fields and state-of-the-art. 66

6.1 Main notations employed in Chapter 6 81

6.2 Detection error results for the BOX Dataset. We report the AUC scores for the detection error relative to each part, as described in Section 6.6.3. 96

6.3 Experimental results for our part-based framework. 100

A.1 Computational complexity and assumptions on the family of warps for the algorithms described in Chapter 4. 119

A.2 Formulas for the update of the warp estimate for the algorithms described in this chapter. 120

1

3D Tracking for Industrial Applications

Computer Vision is disclosing new, unforeseen possibilities in many domains of our life. New applications are proposed every day in a wide number of domains, ranging from gaming to medical analysis, from marketing to online retail.

The industrial domain, too, massively benefits from Computer Vision and its methods, because of some undeniable advantages over competitor technologies: vision-based system are more cost-effective, easier to install and simpler to maintain; moreover, they allow to accurately execute some critical tasks, such as production control in manufacturing, at rates that would be impossible for human operators.

Unfortunately, it is still rare for industrial applications to fully exploit the potential of state-of-the art methods. On the one hand, this is due to the typical "conservatism" of industrial domain: the required degree of dependability of the employed technologies is usually much higher for industry than for other domains, since system upgrades, replacements and production interruptions are expensive. On the other hand, in the case of vision-based technologies there is also another kind of difficulties to overcome: as we shall describe in Section 1.1, industrial environments are usually characterized by peculiar visual features, that make it difficult to apply methods conceived and demonstrated for daily setups. This is especially true for 2D and 3D tracking methods, that are among the techniques industrial applications may heavily benefit.

In this work, we propose efficient and robust methods for image based 3D pose estimation and tracking of rigid objects, suited to applications in industrial contexts, such as Augmented Reality (AR), robotic vision, quality control. Our goal is to contribute to bridge the gap between these contexts and other environments where 3D tracking techniques have gained a high level of robustness and reliability.

Our research has been mainly carried out within the EDUSAFE European research project, described in Section 1.2.1, that aims at developing Augmented Reality based technology for assistance to technical interventions in extreme environments. Of course, Augmented Reality is not the only field of application of the techniques presented in this work: other applications are described in Section 1.2, while a practical test case example, developed for a welding machine guidance system, is described in Section 1.2.2.

1.1 3D Tracking in Industrial Environments: The "Daily Setup" Bias

Vision-based 3D tracking can be defined as the problem of estimating the 3D pose of an object with respect to a camera, based on the acquired images. The problem has been extensively studied and many methods are proposed in the literature; a more detailed definition and a review of the state-of-the-art are given in Chapter 2. Nonetheless, it is interesting to notice how, probably driven by the great number of applications available, many of the most popular methods are applied to common, daily setups, such as indoor scenes with controlled lighting conditions, matte, discriminative objects, possibly seen under moderate occlusions.

It is clear that reliable 3D tracking solutions for industrial environments like those shown in Figure 1.1 would be of great benefit for a wide number of applications, as those presented in Section 1.2. Unfortunately, most of the existing solutions are not suited to such environments, because of the massive presence of challenging conditions, such as:

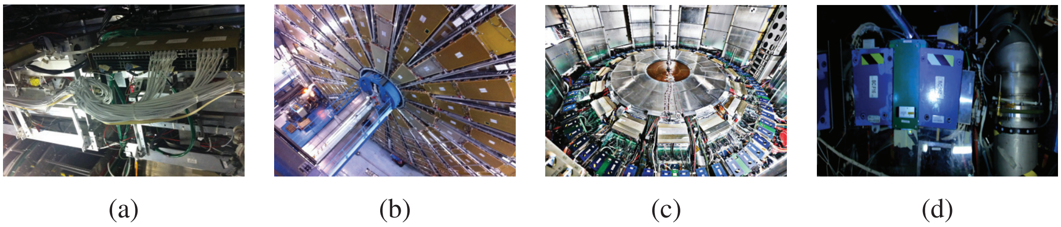
- non-textured objects;
- non-Lambertian surfaces, like metal and glass;
- drastic illumination conditions;
- ambiguous, repetitive patterns (screws, grids, cables, connectors, etc.);
- heavily cluttered scenes;
- large occlusions.

As confirmed by the experimental results presented in Chapters 5 and 6, many methods achieving exceptional performances in daily setups are prone to fail in industrial setups, because of the sources of nuisance mentioned above.

The methods presented in this work aim at alleviating the effect of these nuisances and to provide robust 3D tracking methods suited for industrial applications, even though, of course, they may be profitably employed in a general case.

1.2 3D Tracking Applications in Industrial Environments

Automated systems based on visual sensing are undergoing a growing interest in industrial applications, thanks to their low price and their flexibility. 3D tracking plays an increasingly important



(a) (b) (c) (d)

Figure 1.1 – Examples of the challenging visual conditions encountered in industrial environments: the ATLAS particle detector at CERN, Switzerland. Common sources of nuisance are (a) repetitive patterns, (b)-(d) drastic illumination changes (hand held torches are used for lighting dark places) (c) non-Lambertian surfaces.

role, enabling new applications and opportunities. One of the earliest domains of applications consists in production quality control, where visual systems speed up the verification process achieving very high precision rates; their early employment is partly justified by the fact that the systems usually operate in strictly controlled conditions; moreover, many control tasks only require simple reasoning, such as assessing the presence/absence of a component, or the alignment of 2 pieces; nowadays, the vision-based production quality assessment is widely employed in fields ranging from electronic circuits [1] to fruits and vegetables [2].

Another well-established field of applications where 3D tacking has a prominent role consists in robots for object picking, such as the one shown in Figure 1.2-(a); recent methods employing weakly supervised learning approaches [3] aim at working for objects with generic shapes and materials. A strictly related task is object sorting, for example in waste treatment factories where robots automatically classify, pick and separate objects on a conveyor, as shown in Figure 1.2-(c).

Autonomous navigation and obstacle avoidance is a very active field of research, and autonomous unmanned vehicles are increasingly used for military and civil applications, moving in air, on ground and under water [4].

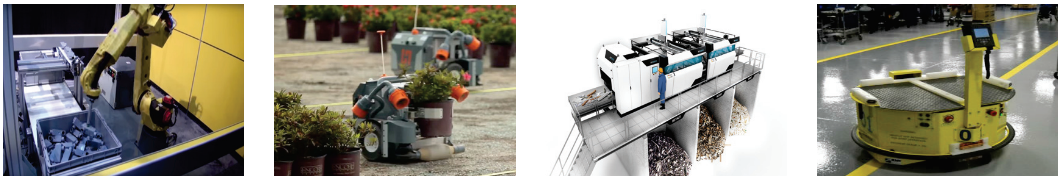
Augmented Reality has been actively investigated for several purposes, ranging from maintenance assistance [5], to personnel training, personnel guidance in warehouses, and quality control: the first commercial products are appearing, for example, for assisting workers in assembly operations on airplane production lines [1], or in logistic warehouses for vision-aided picking [2].

Currently, new possibilities for Augmented Reality-based applications are being disclosed by the recent introduction of devices such as the Microsoft Hololens [6]. Such devices are able to provide accurate localization and 3D reconstruction by combining visual and inertial tracking, and also to perform simple shape reasoning, such as identifying a flat surface where a virtual screen can be overlapped. Methods for 3D object detection and tracking such as those presented in this work provide the ability of detecting known shapes and estimating their position in the reconstructed environments: we hope that this will empower new applications and further expand the capabilities of modern Augmented Reality devices.

---

[1] http://www.airbusgroup.com/int/en/news-media/press-releases/Airbus-Group/Financial_Communication/2016/04/Airbus-Group-Unit-Testia-to-Supply-To-Spirit-AeroSystems.html

[2] http://www.dhl.com/en/press/releases/releases_2015/logistics/dhl_successfully_tests_augmented_reality_application_in_warehouse.html



(a) (b) (c) (d)

Figure 1.2 – Examples of industrial 3D tracking applications: (a) a robotic arm picking metallic pieces [a]; (b) a self-navigating robot displacing plant vases over a warehouse [b]; (c) an automated waste sorting system [c]; (d) an autonomous robot for goods transportation.[d].

---

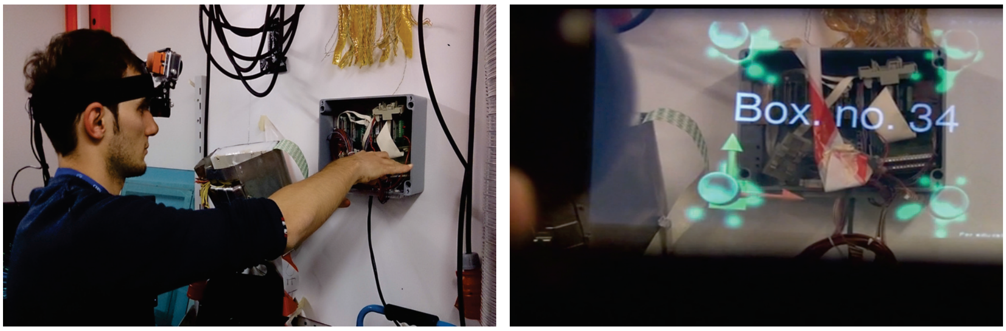
[a] http://goo.gl/Fr4opa

[b] http://www.public.harvestai.com/

[c] http://zenrobotics.com/

[d] http://www.adamrobot.com/en-ca/page/about

1.2.1 Augmented Reality at CERN: the EDUSAFE Project



(a) (b)

Figure 1.3 – The EDUSAFE Augmented Reality prototype in action for assisting technical interventions. Left: a camera placed over the user's head streams images to a server for pose estimation; the pose is transmitted back to an head-mounted see-through display (HMD) for rendering. Right: an example of augmented content seen through the HMD.

The EDUSAFE Marie Curie ITN project (http://edusafe.web.cern.ch/) is a European research project coordinated by the CERN and involving 15 European institutions, belonging to both the academic and the industrial domain; its goal is to design Augmented Reality-based solutions for maintenance assistance in harsh industrial scenarios.

At CERN, technical interventions are often carried out in extreme conditions: technicians and engineers are called to perform complex tasks in dangerous areas, where the high risk of exposure to radioactive and bio-hazardous agents limits the intervention time and is a major cause of stress for operators. The goal of EDUSAFE Project is to investigate Augmented Reality as a way of reducing the time of intervention and the operators' stress, by providing them instructions and environmental data in visual form through an head-mounted display (HMD). Moreover, application of AR-enabled systems to personnel training is also being investigated: the training activity is

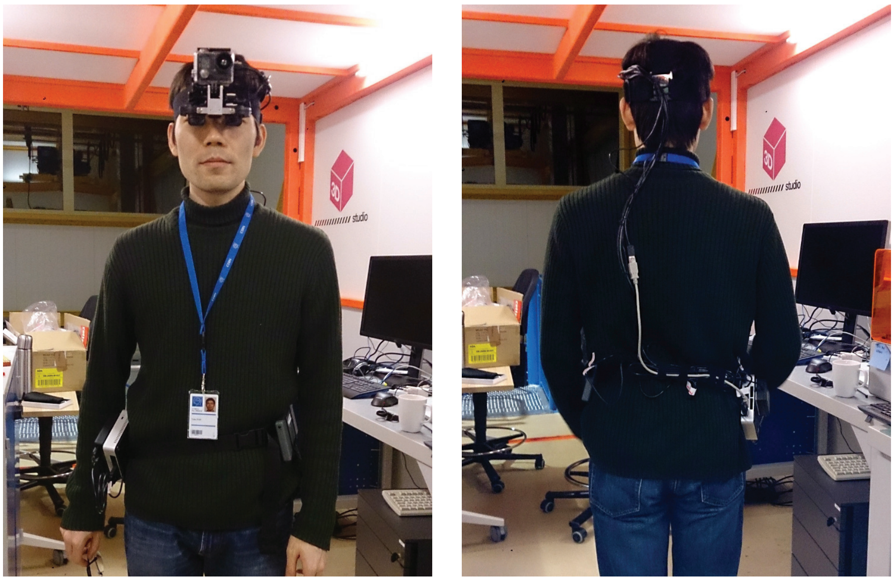

(a) (b)

Figure 1.4 – The Personal Safety Device developed for the EDUSAFE project. (a) front: the compact camera and the Head-Mounted optical see through display are visible. The headset is also equipped with accelerometers and gyroscopes for more robust tracking. (b) back: a portable computing unit and a battery are fixed to the user's belt. Images courtesy of m. Yuta Itoh.

currently carried out by senior technicians that could be profitably employed for other, critical tasks.

The prototype shown in Figure 1.4 has been designed, built and presented at the EDUSAFE final conference on June 20th, 2016. It shows the user instructions for a technical intervention on a generic electric box as the one shown in Figure 1.3.

The system employs a remote server and a wearable, portable unit (PU) carried by the user. Images captured by a camera mounted on the operator's helmet are streamed to the server: there, the pose of the object of interest—an electric box in the user case shown in Figure 1.4—is computed and transmitted back to the user's HMD, where authoring content is rendered on the head-mounted display. Object tracking is performed with the pipeline described in Chapter 6 and fused with information coming by accelerometers placed on the headset. The setup is shown in Figure 1.3: the setup allows the user to keep his/her hands free during the whole intervention.

1.2.2 A Robotic Vision Case Study. Seam Tracking for Industrial TIG Pipe Welding

During a 2-months internship at S&H, Milan, Italy, we developed a tracking system for an industrial pipe welding machine. The welding machine shown in Figure 1.5 is employed in the construction of oil and gas pipelines. Consecutive pieces of pipe are lathed and put side-by-side, so that a V-shaped profile is created at the interface between the pipes. The Tugsten Inert Gas (TIG) welding machine is guided around the interface and the welding torch fills the profile with fused metal. The welding torch position must be adjusted according to the seam width, depth and kept above the center of the seam, which is not always aligned with the machine rail. Currently, the torch guidance is manually carried out by experts: a painful task, considering that the welding torch must be observed during the whole operation, as shown in Figure 1.5, and that usually such welding operations are performed in extreme environments such as deserts, mud, or ice lands.

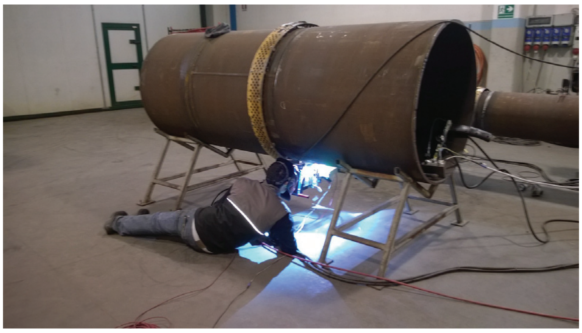
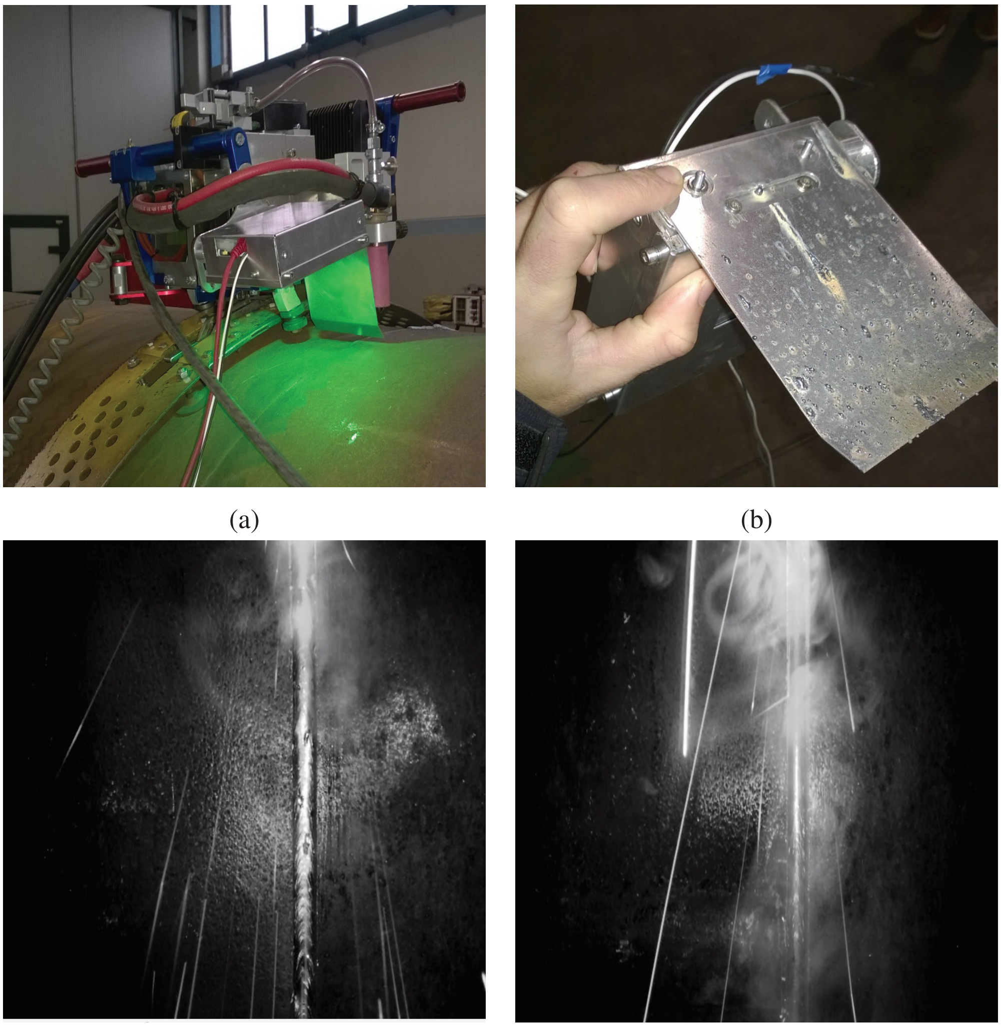


Figure 1.5 – The pipe welding machine in action during an indoor test. Guidance is provided by an expert, that closely monitors the welding torch while it turns around the pipe and manually adjusts the trajectory of the welding torch.

We developed the visual tracking system shown in Figure 1.6, (a), (b) for automatic welding torch guidance. A calibrated camera has been installed on to the welding machine, at a fixed distance from the pipe surface. The seam is tracked on the acquired images employing a generalized Hough transform that tracks sets of parallel lines detected on the image, and the guidance systems outputs the displacement of the machine from the seam center line in mm (horizontal displacement and scale). In order to maximize accuracy, the guidance system must be placed as close as possible to the welding torch; unfortunately, this causes artifacts created by sparks, smoke and projected welding residuals appear on the images, as shown in Figure 1.6, (c), (d), seriously harming the tracking performances. The system is equipped with an illumination system, a band-pass filter in front of the camera, and a protective shield, shown in Figure 1.6, (a), (b). The prototype was tested on videos recorded during real welding sessions, tracking a 7mm-width seam with sub-millimiter accuracy at at 5Hz on an embedded device.

In this scenario, given the extremely simple shape of the tracked object, a standard technique can successfully perform a 3D tracking task; nonetheless, the test case is representative of the challenging problems that 3D tracking must face when operating in industrial environments: occlusions and clutter coming from different sources, limited computational resources, the presence of non-Lambertian surfaces, and heavy illumination changes caused by the unstable light produced by the welding torch. Even with additional hardware protections, such as a protecting shield and supplementary lighting and filters, the nuisance of these factors can only be partially alleviated.

Our experience confirmed that it is of crucial importance to design robust algorithms, able to operate in realistic conditions, in order to let Computer Vision enable new applications in industrial domains.



(a) (b)

(d) (d)

Figure 1.6 – The automatic vision-aided system developed for a welding machine. Top: (a) close-up of the guidance system; (b) the shielding after a welding run : residuals of fused material are present on the protection shield. Bottom: (c), (d) examples of acquired images : the the welding seam is occluded by sparks, smoke and incandescent residuals. An integrated illumination system, a band-pass optical filter and a protective shield have been integrated to the system for greater robustness.

1.3 Contributions

In this thesis, we propose two new methods for robust and accurate 3D tracking, able to operate in the difficult conditions mentioned above. Our goal is to bridge the gap between 3D tracking performances of state-of-methods in daily setups and in industrial environments. More in particular, the contributions of this thesis are:

- A thorough analysis of dense alignment methods and their employability for 3D tracking, presented in Chapter 4. Following the footsteps of the excellent survey presented in [7], we analyze recently introduced methods in the same framework (such as the Efficient Second Order Method [8]) and discuss their employability for 3D tracking applications;
- The introduction of a novel dense descriptor, that we refer to as "Descriptor Fields", originally presented in [9], for robust image alignment with non-textured and non-Lambertian objects under heavy light changes;
- A 3D tracking approach based on the detection of stable parts of the object, that we introduced in [10], robust in presence of difficult light conditions, clutter and severe occlusions, described in Chapter 6;
- 2 datasets for evaluation of 3D tracking methods, exhibiting the challenging conditions encountered in industrial environments described above. The published datasets are part of the “International Workshop on Recovering 6D Object Pose” Challenges presented at ICCV 2015 and ECCV 2016.

This thesis covers the following peer-reviewed accepted or pending publications and demos:

- Alberto Crivellaro and Vincent Lepetit. Robust 3D tracking with descriptor fields. Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, USA, 2014.
- Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, Vincent Lepetit. A Novel Representation of Parts for Accurate 3D Object Detection and Tracking in Monocular Images. International Conference on Computer Vision (ICCV), Santiago, Chile, 2015.
- Dat Tien Ngo, Sanghyuk Park, Anne Jorstad, Alberto Crivellaro, Chang Yoo, Pascal Fua. Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture. International Conference on Computer Vision (ICCV), Santiago, Chile, 2015.
- Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, Vincent Lepetit. Robust 3D Object Tracking from Monocular Images using Stable Parts. Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2016.
- **DEMO:** Alberto Crivellaro, Yannick Verdie, Kwang Moo Yi, Pascal Fua and Vincent Lepetit. Tracking Texture-less, Shiny Objects with Descriptor Fields. International Symposium on Mixed and Augmented Reality (ISMAR), Munich, 2014.

- **DEMO:** Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua and Vincent Lepetit. 3D Object Tracking from Monocular Images using Stable Parts. Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, 2016.

1.4 Outline

This thesis is organized as follows.

- The next chapter provides an extensive overview of the state of the art in the field of 3D rigid object tracking.
- In Chapter 3 we introduce the main notations employed in the thesis and describe the mathematical background relevant to our work, that is, the perspective projection model and its approximations, the most common image distortion models, and the exponential map parametrization for 3D rotations.
- A description of the main dense alignment paradigms and their application to 3D tracking, along with their recent extensions, is presented in Chapter 4.
- In Chapter 5 we propose a 3D tracking framework based on dense image alignment and a novel dense image descriptor, the Descriptor Fields.
- A complementary approach based on the detection of stable parts of the object, suited for tracking severely occluded objects, is described in Chapter 6. For both the methods presented in Chapters 5 and 6, extensive evaluations against state-of-the-art, implementation details and applications are described in the respective chapters.
- Finally, Chapter 7 provides final remarks and discusses future research directions.

2

3D Tracking : State of the Art

After introducing the main industrial applications of 3D tracking in the previous chapter, in this chapter we give a more rigorous definition of 3D tracking and present an overview of the relevant related work.

We refer to 3D tracking as the problem of retrieving the pose of a rigid object, based on one or more images captured by a camera; the pose is defined as the rigid transform between two Euclidean reference systems, one integral with the object and the other integral with the camera.

The **pose** is described as the 3D rigid transform mapping the camera reference system to a reference system integral with the object. The 2 opposite problems of retrieving the camera pose with respect to the object and the object pose with respect to the camera are symmetric: the exact same methods can be employed for solving both variants, passing from one to the other just requires a change of coordinates. The rigid transform has 6 degrees of freedom, 3 for the rotation and 3 for the translation.

The focus of this work is on **rigid objects**: tracking of non-rigid or articulated objects such as deformable surfaces and human bodies may require ad-hoc formulations and explicit modeling of the deformations the target objects can undergo. Some extensions of the proposed methods with application to deformable surfaces and articulated objects are investigated respectively in Sections 5.4 and 6.7.1. When the target object consists in the whole scene, the 3D tracking is also referred to as *camera localization*.

The notion of **camera** is quite broad. Traditionally, 3D tracking techniques have been developed employing inexpensive, low-resolution RGB or gray-scale cameras. The recent introduction of cheap, effective hardware for depth sensing has lead to the spread of methods exploiting based on RGB-D images. The choice of the employed camera should depend on the required application: while in general depth sensors are undergoing a growing interest for their accessibility and their robustness in many daily scenarios, their employability is still limited for outdoor scenes and for objects with non-Lambertian surfaces (such as metallic objects). Color information is a very strong cue for object recognition, but it is also subject to radical degradation in presence of strong illumination variations. As this work mainly focuses on industrial scenarios, we employ gray-scale images, since they can be used in the most general situations and allow for a reduced computational cost.

3D tracking methods usually employ some kind of prior information about the object, about its geometry (e.g. CAD model) and/or about its appearance (textured 3D models, templates, keypoints mapped on the object surface, etc.). The pose estimation is usually performed across sequences of subsequent frames such as videos, hence the term **tracking**. For all the methods presented in this thesis, each frame is treated independently and temporal consistency can be enforced *a posteriori* or employed for making the estimation faster and more accurate. The first method aligns incoming images with registered key-frames, while the second follows the so-called **tracking-by-detection** paradigm, where the object is independently detected on each frame. Other methods exist, based on the so-called *temporal tracking paradigm*, aiming at retrieving the incremental displacement of an object between subsequent frames of a video sequence. These methods are usually faster than tracking-by-detection methods, but suffer from drawbacks such as drift and the need of initialization and re-initialization whenever the tracking is lost.

2.1 Related Work

The literature on 3D tracking is vast; many different approaches have been proposed and the state-of-the-art is rapidly evolving, also thanks to the recent introduction of low-cost depth sensors and to recent advances in deep learning techniques. Nonetheless, vision-based tracking remains an open problem, because of the numerous nuisances that may intervene: these may be caused by physical features of the target object, such as specular materials, non-textured surfaces, configurable or articulated objects, and by features of the surrounding environment, such as occlusions, scenes with clutter or ambiguous patterns, poor illumination conditions or varying lighting.

Other relevant challenges concern the scalability for simultaneous detection of multiple objects, reduction of computational and memory footprint, lack of offline data for training, and category-based tracking, that is, focusing on a generic instance of a given category (e.g. a car, a chair, a horse) rather than on a single physical object. While some state-of-the-art methods allow to cope with some of the mentioned difficulties, no existing technique allows to simultaneously overcome all of them. In this section we present the most relevant work in 3D tracking, discussing the main advantages and disadvantages of the different methods.

2.1.1 Edge-based Methods

One of the earliest research direction for 3D tracking relies on edges and image contours. Edge-based methods usually represent a 3D object as a set of control points regularly sampled along a wire-frame 3D model of the object [11, 12, 13, 14]. The control points are projected onto the image using a prior pose estimation; then, for each control point, a 1D search is executed in the direction perpendicular to the predicted edge, to find the strongest image gradient close to it, which is assumed to be the new position of the edge, as shown in Figure 2.1, (a) (d). The new pose estimate is calculated by minimising the distance from the control points to the actual image edge found, and tracked over time using temporal filtering, such as a Kalman filter [11].

The main pitfall of edge-based methods is the ambiguity of the primitives employed for tracking, since contour information is much more ambiguous than other local features such as keypoints. This makes edge-based methods particularly sensitive to spurious results and local minima. Possible techniques for alleviating the nuisance of ambiguous edge matching include the use of a RANSAC matching scheme [15], employing robust estimators [13, 16] or using a particle filter for the simultaneous evaluation of multiple hypotheses [14]. However, edges and contours are relatively fragile in practice, and sensitive to large occlusions, clutter, and light changes. For example, in the environment depicted in Figure 5.1, the object contours are perturbed by their reflections on the metallic surface and the contours of the specularities in the background.

2.1.2 Keypoints-based Methods

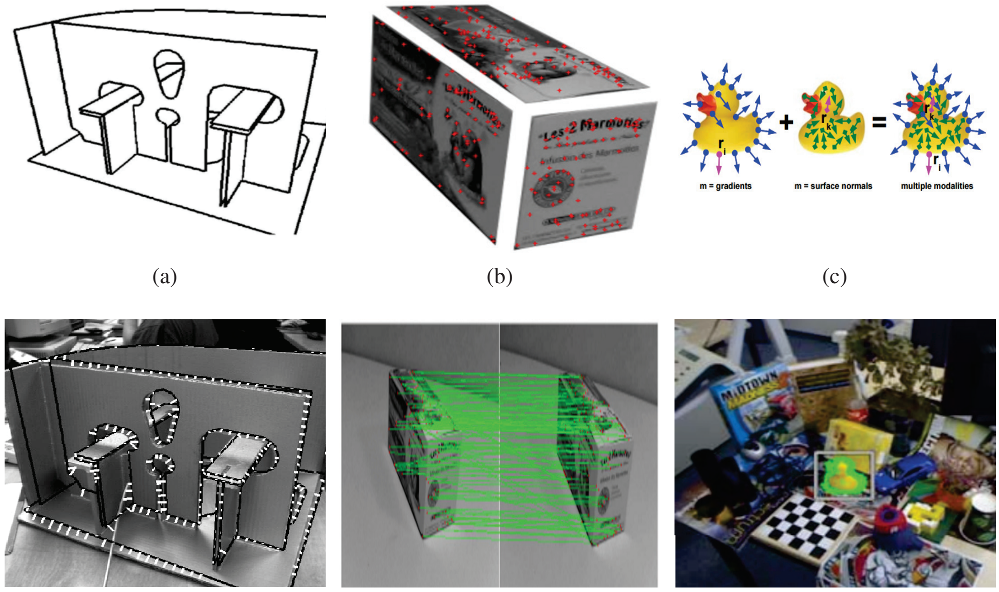
More recently, keypoint-based methods became popular [17, 18, 19]: keypoints can be extracted and matched more reliably than contours, since they can be efficiently characterized by the surrounding texture by mean of local, invariant descriptors [20, 21, 22, 23, 24, 25]. These methods usually represent the object as a set of local features detected on a textured model, such as the popular SIFT keypoints [26]. At run time, keypoints are detected on the images and matched to those of the pre-computed object models; detection and pose estimation are executed based on the retrieved correspondences.

The main limitation of keypoint-based methods is that they can only be employed if the objects are textured enough; moreover, as for other local features, reliably matching keypoints across images becomes problematic in presence of clutter and repetitive, locally ambiguous patterns, such as screws, grids or bundles of wires.

Some works combine keypoints with edges [27, 28]; however, as discussed above, extracting and matching edges remains delicate. Since keypoints can be reliably matched under heavy viewpoint changes, as opposed to narrow baseline methods as optical flow or dense image alignment, [29] tries to combine the best of both worlds: sparse, keypoints-based pose estimation is computed along with a dense pose estimation, based on frame-to-frame optical flow and the optical flow computed employing rendered images of the model. Simple forward/backward consistency check is used to select either the sparse and or the dense result. However, their setup requires a stereo configuration, which limits the applicability of a 3D tracker.

2.1.3 Region-based Methods

Besides keypoints, silhouettes and region based methods have also been proposed. In [31, 32], 3D tracking problem is considered as joint 2D segmentation and 3D pose estimation problem, and the method looks for the pose that best segments the target object from the background. Contours and edges are used in [33] with multiple hypotheses to provide robust pose estimation. Partial



(a) (b) (c)

(d) (e) (f)

Figure 2.1 – **Edge-based tracking:** (a) the pose of a wire-frame 3D model is estimated on an incoming frame (d) based on image edges (from [13]). **Keypoint-based tracking:** (b) a set of keypoints detected on a textured object surface are matched with those detected on incoming frames (e) [a]. **Template matching:** LINEMOD templates (c), build from RGB-D images, are employed for detecting non-textured objects in cluttered environments (f) (from [30]).

---

[a] From: http://visp-doc.inria.fr/doxygen/visp-daily/tutorial-detection-object.html

occlusions, however, are difficult to handle with such approaches. In Chapter 6 we compared our part-based method with [32] on sequences with highly occluded objects, our tests confirm that relying on image regions is not robust for the scenarios considered in this thesis.

2.1.4 Dense Image Alignment Methods

With the growing computational power of modern devices, dense image alignment approaches [34, 35, 36, 7, 37, 38, 39] have become very attractive. These methods look for the pose of an input image by aligning the pixels of this image with those of a registered template: the alignment is typically performed by iteratively minimizing some distance function, such as the sum of squared differences, of the location intensities.

Although computationally more expensive than methods based on local features, they can exploit most of the image information without being limited to contour or keypoints features. Therefore, they can properly handle poorly textured objects, and they are more robust with respect to local artifacts induced, for example, by specularities, or ambiguous patterns. Since we developed a 3D tracking framework based on dense alignment, we will discuss in detail this kind of approaches in Chapter 4. Our own framework will be detailed in Chapter 5.

2.1.5 Depth-based Methods

The development of inexpensive 3D sensors such as the Kinect has recently sparkled different approaches to 3D object detection. Depth data are appealing for several reasons: for example, the pose estimation problem can be cast as the problem of aligning two 3D point clouds, for which several well-established methods exist [40]. Moreover, occlusion reasoning becomes easier than on monocular images, since occlusions necessarily lie in front of the object [41].

[42, 43] use votes from pairs of 3D points and their normals to detect 3D objects. The popular LINEMOD tracker [44, 30] uses surface normals extracted from depth images and edge orientations from RGB images as template features for dealing with poorly textured objects. Using a fast matching scheme the incoming image is matched against thousands of such template features corresponding to different viewpoints of the object to robustly detect its presence. Despite its robustness to clutter and light changes, according to our experimental results, this approach is sensitive to occlusions, a key-requirement in our context. [45] shows that substantial performances gain in terms of accuracy and computational efficiency can be achieved by assigning different weights to different regions of the LINEMOD templates, learnt with a discriminative approach; moreover, rather than testing each image against all the templates, cluster of templates are built, and cascade classifiers are trained to check if an image must be checked against the templates of each cluster. Nonetheless, these improvements only partially alleviate the sensitivity to occlusions of LINEMOD tracker.

Most recent depth-based methods leverage the power of machine learning for better performances, and are discussed below.

2.1.6 Learning-based Methods

Spreading of learning-based methods revolutionized many areas of Computer Vision, including 3D tracking. Several approaches showed that it is possible to learn a regression model relying images to 3D poses. [46] describes an online learning framework for real-time estimation of 3D poses, but it is limited to planar targets. [47] uses a decision tree applied to RGB-D images. [48] learns a regression model based on random forests relying the change of appearance between a template and an image with the change of the object pose. The method has been demonstrated both for tracking of planar surfaces based on RGB images and for 3D object tracking based on depth images. Each pose parameter is estimated independently by a different forest; moreover, different forests must be trained for different points of view uniformly sampled on a sphere around the object. At run time, the forests for the viewing point closest to the previously predicted pose are selected for the incoming frame. [49] improves this method for temporal tracking of 3D objects based on depth images; more in particular, a huge number of viewpoints is sampled on a geodesic sphere around the object, and a very simple regressor made by a single tree is learnt for each viewpoint and for each pose parameter. At testing time, multiple trees from the closest predicted viewpoint are employed for predicting the object motion. Although this method achieves impressive computational and memory efficiency and fair performances in presence of moderate occlusions, it has only been demonstrated on depth images. Moreover, their temporal tracking paradigm needs initialization for first frame of the sequence and every time the tracking is lost, as opposed to our part-based approach described in Chapter 6, which is based on a tracking-by-detection paradigm.

In [50] co-training with hough forests are used for multi-object detection. Co-training has the advantage of avoiding the need for background/negative training data; however, at testing time it requires multiple passes over the trained forest to predict the location and pose of the object. Moreover, since negative data are not employed for training, it is unclear how this method can perform in the presence of similar looking clutter in the background. The employed features, consisting in pairwise comparisons of gradient-based template maps, are replaced in [51] by features learnt by a sparse auto-encoder; moreover, authors of [51] show how the next best view can be predicted within their framework for a refining the pose estimation.

[52] proposed a learning-based framework processing point clouds, that, as opposed to raw depth images, are scale invariant and can be easily obtained from online fusion algorithms [53] that reduce noise and fill missing data. After edgelet features are detected over 3D shapes, feature vectors are obtained by computing their dominant orientations over a grid of voxels, and employed for training a soft label Random Forest classifier. The output of the classifier is a set of soft-assigned pose labels, plus a label predicting the probability of the presence/absence of the object for each voxel. Resilience to occlusions is achieved employing feature-aware features (special values are assigned to the feature vector components corresponding to occluded voxels), while the discriminative power of the detector in presence of clutter and similar distractor objects is enhanced with an iterative training scheme that enlarges the margin between object and non-object classes.

In [54] objects are detected employing an intermediate representation in the form of a dense 3D object coordinate labelling, as previously done for related tasks as human pose estimation [55] and camera localization [56]. A random forest is trained for predicting for each pixel its probability of belonging to a given object, as well as the 3D object coordinate labels. An energy function comparing the acquired depth image, the object coordinates and the object probability predicted by the random forests with, respectively, depth images, object coordinates and object silhouettes rendered under different poses is minimized using a robust optimization scheme for obtaining the final pose and location of the object on the image. The original deterministic energy formulation is replaced by a score predicted by a Convolutional Neural Network in [57], while a real-time tracking framework based on this approach and a temporal particle filter in the pose space is proposed in [58].

The intermediate object coordinate representation is shown to work for both textured and textureless objects, in presence of light changes and in presence of moderate occlusions. Recently, the method has been successfully improved exploiting auto-context random forests to work exclusively based on monocular images in [59]. The approach of [59] is closely related to our part-based method presented in Chapter 6, since it predicts 3D poses based exclusively on monocular images and the object coordinates can be somehow interpreted as densely sampled parts; the two methods should be considered complementary, the former more adapted to small, poorly textured objects such as the ones of the LINEMOD dataset [44], the latter for highly occluded objects with a small number of discriminative parts, such as the ones of the datasets described in Section 6.6.2.

2.1.7 Deep Learning-based Methods

Methods based on deep learned architectures are currently disclosing new, unprecedented capabilities for many different Computer Vision tasks, including 3D tracking.

Deep models are employed in [60] for real-time large scale localization and 3D pose estimation in open spaces starting from monocular images. The 6 degrees of freedom of the pose are directly predicted by a Convolutional Neural Network (CNN), employing quaternion parametrization for rotations. Similarly as the approach proposed in Chapter 6, the pose is obtained solving a regression problem with CNNs; on the other hand, predicting the pose based on the whole image is appropriate for scene localization, while, for object tracking, our method only exploits detected parts of the object for being resilient to large occlusions.

Discriminative deep descriptors are learnt in [61] for object detection and pose estimation. The descriptors are predicted by a CNN trained enforcing that the distance between descriptors is large when the descriptors represent different objects, and directly related to the distance between the poses when the descriptors are from the same object. A similar idea is proposed [62], where a deep siamese network is trained, taking as input pairs of images and enforcing that images having dissimilar poses should be mapped into distant feature representations. In both works, object detection and pose estimation are performed on nearest neighbor search over a set of templates in the feature space, similarly as done in LINEMOD [44]. This strategy is claimed to work better than direct pose regression in [62], due to the limits of existing pose representations. Our pose representation introduced in Chapter 6 provides an elegant solution, allowing, among others, for direct, efficient pose regression.

An approach combining template matching and deep learning for object recognition is proposed in [63]. This work shows that the performances of a standard CNN for object recognition can be boosted by employing prior knowledge about the object. More in particular, a so-called template layer is added after the convolutional network layers, sparsifying their output by performing element-wise multiplication with pre-computed object template masks. The sparsified features are treated by a classifier whose output is the probability of the presence of the object and a soft-assigned pose label. Therefore, the pose is computed from pre-computed quantized poses employing the soft-assigned pose labels, for example taking the pose with the largest label or computing a weighed sum over the best scored poses. The sparsifying effect of the hard-coded template masks enforces learning of better structured features and sensibly boosts accuracy. At the best of our knowledge, this method has only been demonstrated on depth-based input.

2.1.8 Part-based Methods

Representing objects as a collection of parts has been exploited since the early days of modern Computer Vision, with the introduction of aspect graphs models [64]. Recently, the notion of parts naturally arose in works focusing on category level object recognition, such as [65, 66], where characterizing common parts of object categories like cars or bikes is beneficial for recognizing class instances from very different viewpoints. Moreover, a coarse viewpoint estimation can be computed as a-side result. [67, 68, 69] focus on 3D viewpoint classification at a category level, building object parts from clusters of local features and learning about the 3D spatial relations among parts.

Many works were motivated by the success of the Deformable Part Models [70] developed for 2D detection, which were exploited for coarse viewpoint estimation in [71] and successfully extended to 3D, for example in [72] and [73]. [74] also performs 3D tracking through part-based particle filtering. [66] learns part-based appearance models of object classes and the 3D geometric relationship between parts using synthetic 3D models; at testing time, the detections of parts are verified and checked against the 3D geometric information, also providing a coarse viewpoint estimation. [75] uses contours as parts. In [76], 3D shared parts are learned employing both synthetic images rendered from CAD models and real images for fine pose estimation.

In general, these works only provide coarse viewpoint estimation, since they focus on object categories; moreover, they are not robust to occlusions of some of the parts, especially because the 2D location of the part is solely considered to constrain the object pose. Some of them assume homographies and planar parts [77, 46, 74]. Finally, they usually require huge computational resources, making them un-suited for time-critical applications. Our part-based 3D tracking framework described in Chapter 6 overcomes these limits; more in particular, it is able to directly infer 3D poses also from a single part, without any assumption on the shape of the parts.

2.1.9 Large Scale Pose Estimation Methods

A recent class of methods exploits deep learning for solving large-scale, category-based recognition problems. Coarse 3D poses are predicted by a CNN classifier in [78] for indoor objects such as chairs and beds from 3D normal maps, based on the output of the fine-grained object segmentation of [79]. An approach for detection and fine pose estimation of objects trained on a wide number of CAD models retrieved on the internet is described in [80].

In [81], a fine-grained CNN classifier is trained employing millions of synthetic images created rendering 3D models to simultaneously predict the category and the pose of an object. A similar approach is proposed in [82], that additionally predicts the location of category-specific keypoints on the images.

A key-issue for this kind of methods is the need of huge amounts of annotated training data; generating synthetic data from CAD models is a scalable, effective approach, but it requires to bridge the gap between the limited realism quality of the rendered images and the real images. Some methods, such as [76], address this problem training on both mixed real and synthetic images; others focus on rendering as realistic as possible images [83, 81]; others, such as [84], propose to directly learn domain adaptation functions for comparing features extracted for real and for synthetic images.

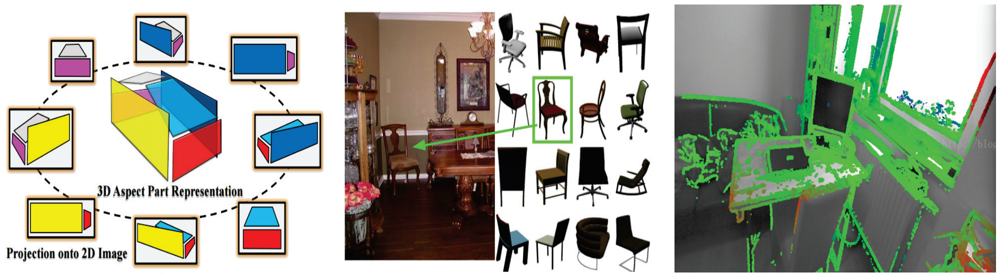
Unfortunately, at the moment, the accuracy of the predicted poses and the computational efficiency are not suited yet for applications such as Augmented Reality; moreover, these approaches typically require massive amounts of training data, which are often unavailable in our setups: therefore, their application to the scenarios considered in this thesis would be impractical at best.

2.1.10 SLAM

Finally, a very active and related field is SLAM (Simultaneous Localization and Mapping) [85, 39, 86, 87]. In this approach, 2 threads run concurently, one computing a sparse or semi-dense map of the surrounding environment consisting of 3D points, and the other tracking the camera motion in the map across a video sequence. The map is updated online exploiting new observations on the tracked frames. The camera motion and the 3D map may be iteratively refined with bundle adjustment strategies.

Modern methods follow two main paradigms, either feature-based SLAM or direct SLAM. Feature-based systems such as the famous PTAM [85] and the more recent ORB-SLAM [87], are built upon the detection and matching of sparse features on the image; as a consequence, the resulting 3D maps consist in sparse 3D points. These methods carry all the advantages and the limitations of tracking based on sparse features: resilience to occlusions, invariance to wide viewpoints changes, but reduced reliability for poorly textured environments. On the other hand, direct SLAM methods, such as DTAM [39] or LSD-SLAM [86], align subsequent frames and build maps employing dense or semi-dense image alignment methods, so that the resulting maps consist of semi-dense point-clouds, as shown in Figure 2.2. They are more adapted to deal with poorly textured surfaces.

A very interesting feature of SLAM systems is that they do not require any prior 3D knowledge, but on the other hand they only provide a relative pose, defined up to a scale; this is not suitable for many Augmented Reality applications. In Section 6.7.2 we give an example of how an object detection pipeline and a SLAM system can be combined for practical applications.



(a) (b) (c)

Figure 2.2 – (a) Part-based tracking: a car is modeled as a set of aspect parts and their reprojections under different viewpoints, from [74]. (b) Large scale detection and pose estimation from a large dataset of 3D models, from [80]. (c) The semi-dense 3D map reconstructed with the SLAM method [86], shown as colored points projected to the incoming frame.

2.2 Addressing Challenges in Industrial Environments

As pointed out above, the state of the art is in 3D tracking is broad, and no established paradigm exists, yet, able to undertake all the challenging conditions encountered in the considered environments, resumed in Section 1.1. So, different techniques are suitable for addressing different challenges.

2.2.1 Non-textured Objects; Ambiguous Patterns

When dealing with poorly textured objects or locally ambiguous patterns such as grids, bundles of wires, and so on, methods based on local features become unreliable. Typical solutions consist in holistic approaches, such as those based on dense image alignment or on template matching. When applicable, exploiting depth information is a valuable option in those cases where color information becomes ambiguous.

2.2.2 Specularities and Illumination Artifacts

Several works attempt to make dense image alignment more robust in presence of challenging illumination effects: [88] extends a dense image alignment approach, adding terms in the objective function in order to explicitly estimate some illumination parameters; this has the obvious drawback of increasing the search space and the optimization complexity. In [89], the tracked surface, such as a CD cover, is split into patches and normalize the patches independently. Although this significantly improves the robustness, it is not clear how to split an arbitrary surface, especially in 3D. Relying on illumination invariant features, such as the Descriptor Fields introduced in Chapter 5 achieves good results at reasonable burden.

Rather than treating specular artifacts as nuisance, other kinds of methods, exploit them to improve the accuracy of the registration [90, 91]; interestingly, such approaches achieve a great accuracy, but typically work only in controlled environments.

Learning-based methods, typically achieve illumination invariance by providing rich learning datasets showing the target objects under as varied illumination conditions as possible. The generalization capabilities of the employed methods becomes crucial for achieving illumination invariance, since the amount of learning data is limited and generating accurate synthetic data under different lighting conditions is challenging.

2.2.3 Occlusions and Clutter

Resilience to occlusions is a key-requirement of tracking methods, since occlusions are likely to occur in a wide number of scenarios. Occlusions on well textured object can be successfully handled by methods based on local features, such as keypoints, discussed above; some methods attempt at explicit occlusion modeling [92, 93], but besides adding considerable computational complexity, these approaches are usually limited to specific scenarios, such as objects standing on a plane, hand-held objects or cars on the road. More recent approaches discussed below achieve robustness to occlusions by learning occlusion-aware features [52] or employing mixed local-global object representations [54]. Our part-based approach aims at being intrinsically robust to occlusions by relying on a minimal part of the object appearance.

2.3 Conclusion

In this chapter we discussed the most representative works of state-of-the-art, along with their main advantages and limitations. Despite the great variety of proposed techniques, some trends can be identified in the current research directions; among them:

- The growing power of computational devices allows traditional frameworks such as dense image alignment or template matching to be exploited for 3D tracking.
- The appearance of new sensing technologies, such as depth sensing cameras, is pushing forward the capabilities of 3D tracking technologies. On the one hand, the current limits of depth sensors, such as their limited accuracy and resolution, their failure in presence of specular surfaces and in outdoor environments, or the lack of such tools on commodity mobile devices, will arguably be overcome in next years; on the other hand, it still makes sense to focus on monocular 3D tracking, not only because nowadays it is employable in a wider range of situations, but also because extending methods working with monocular images to depth data is usually much easier than the other way round.
- Learning and, more recently, deep learning-based techniques have arisen as powerful and effective tools for solving not only large-scale problems involving massive amounts of data but also geometric problems such as 3D object tracking. Nonetheless, as we show in Chapter 6, care should be taken in order to exploit the full potential of these tools, not only in terms of training procedures and architecture optimization but also in terms of model formulation.
- Most probably, at least for some time the only general-purpose, flexible, universal vision machine available will be the human brain; but it is a fact that interactions among techniques for solving different classical computer vision tasks, such as object category recognition, instance detection, pose estimation, SLAM, model-based 3D tracking, non-rigid tracking, and so on, are going to become more and more strictly overlapped; therefore, any new proposed approach should account for potential generalizations and interactions with other methods and tasks. More considerations about that will be presented in Chapter 7.

In the next chapter, we will describe the mathematical framework in which our 3D monocular tracking problem is formulated, before giving an overview of dense image alignment methods in Chapter 4 and describing our proposed frameworks and their applications in Chapters 5 and 6.

3

Mathematical Framework

After describing the motivations and the main applications of this work in Chapter 1 and a relevant selection of state-of-the-art work in Chapter 2, in this chapter we introduce the mathematical background of our work.

Since its first rigorous formulation during Italian Renaissance in XIV century, perspective projection has been extensively studied and employed in a wide range of domains, and it is currently used for modeling the image formation process in modern cameras in almost the totality of Computer Vision algorithms. We give an overview of the image formation model employed for all the methods proposed in this work, based on perspective geometry, in Section 3.2.

Despite its great accuracy in describing the mapping from real world objects to their representations on images, the perspective camera model is difficult to handle because of its non-linear nature; for this reason, several 3D tracking methods employ affine approximations, described in Section 3.3. Such approximations greatly simplify tracking problems, but unfortunately, as discussed in Section 3.3, they can only be employed in particular situations, that do not correspond to the scenarios considered in this work. In chapter 6, we will discuss the properties of different pose representations under different projection models.

Another fundamental and delicate aspect about 3D tracking is how to parametrize 3D poses, especially rotations. Several parametrizations exists, with different pros and cons: in Section 3.5 we describe the parametrization we employ for all the methods described in this thesis, the exponential map representation, briefly outlining its advantages over alternative representations.

In the next section we outline the main notations employed in this thesis.

3.1 Main Notations

The main notations employed in the remainder of this work are summarized in Table 3.1. In general, scalars are represented by plain characters, while matrices and vectors are boldface. Two related vectors, one belonging to the 3D world and the other to the image plane, are referred to with the same letter, respectively upper and lower case (e.g. an image pixel $\mathbf{x}$ representing a 3D point $\mathbf{X}$). Images are generally described as bi-variate functions mapping a bi-dimensional pixel location $\mathbf{x}$ to its intensity value $I(\mathbf{x})$. Since we mainly deal with grayscale images, we consider all images are made by a single channel, unless explicitly specified, such as in Appendix A.5.

Notations employed within a single chapter are resumed at the beginning of the chapter.

symbol	domain	meaning
$\mathbf{X}$	$\mathbb{R}^3$	3D point
$\mathbf{x}$	$\mathbb{R}^2$	pixel
$\mathbf{R}$	$\mathbb{R}^{3\times 3}$	Rotation matrix
$\mathbf{t}$	$\mathbb{R}^3$	Translation vector
$\mathbf{K}$	$\mathbb{R}^{3\times 3}$	internal calibration matrix
$\mathbf{C}$	$\mathbb{R}^3$	camera center
$\mathbf{I}$	$\mathbb{R}^{3\times 3}$	3 by 3 identity matrix
$\mathcal{P}(\cdot)$	$\mathbb{R}^3 \to \mathbb{R}^2$	perspective projection
$I(\cdot)$	$\mathbb{R}^2 \to \mathbb{R}$	Mono-channel image
$(\cdot)^\top$	–	Transposed matrix or vector
$(\cdot)^{-1}$	–	Inverse of square matrix
$\mathcal{O}(\cdot)$	$\mathbb{R}$	Computational complexity

Table 3.1 – Main notations employed in this thesis.

3.2 Perspective Camera Model

Mathematically, image formation can be described as a non-linear mapping from the Euclidean 3D space to the 2D image plane. Since the goal of 3D tracking is to infer information about the object pose based on its representation on an image, the mathematical model describing the physical image formation process is of crucial importance: all methods will attempt, at some point, to estimate some of the parameters of this model, or to invert it.

In this section we present the perspective projection model, the most widely employed image formation model. The camera is modeled as a pinhole camera, and the image formation process as a perspective projection. Most cameras differ from a pinhole camera for many aspects, in particular for the use of lenses, so that the perspective camera model is just an approximation of the physical image formation process. Some additional effects are taken into account by employing distortion models on top of the perspective projection, as described in Section 3.4, while other physical phenomena (such as chromatic aberration) are usually neglected in 3D tracking domain.

For sake of simplicity, in this thesis we only give an operative description of all the presented projection models; we refer the interested reader to [94] for a thorough description in a rigorous mathematical framework with details on their analytical properties.

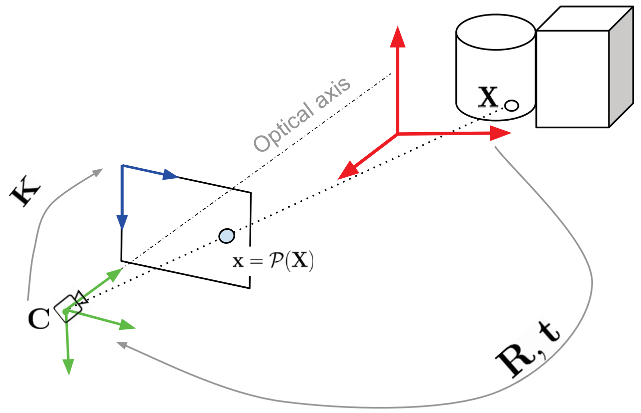


Figure 3.1 – Perspective projection model: a rigid transformation maps a 3D point $\mathbf{X}$ in the object reference system (red) in the camera reference system (green). The internal camera matrix $\mathbf{K}$ encodes the parameters of the projection from the 3D camera reference system to the 2D image plane (blue). The center of projection $\mathbf{C}$ coincides with the origin of the camera reference system; the optical axis is defined as the straight line perpendicular to the image plane passing through $\mathbf{C}$.

As shown in Figure 3.1, a perspective projection is defined by a plane, the image plane, and a 3D point $\mathbf{C}$, the center of projection. A generic 3D point $\mathbf{X}$ is mapped to a point on the image plane, given by the intersection between the image plane itself and the straight line relying $\mathbf{X}$ with the center of projection $\mathbf{C}$. The line passing by $\mathbf{C}$ and perpendicular to the image plane is usually referred to as the *optical axis*.

More in particular, the mapping $\mathcal{P}$ between a point $\mathbf{X} \in \mathbb{R}^3$ in the 3D world reference system and its representation $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{x} = \mathcal{P}(\mathbf{X})$ on an image can be expressed as the composition of 3 main steps:

- **3D Rigid Motion**: the first step is a change of coordinates of the 3D point $\mathbf{X}$ from the world reference system to another reference system, called the camera reference system, with the rigid motion

  $$\mathbf{X}_{cam} = \mathbf{R}\mathbf{X} + \mathbf{t}, \tag{3.1}$$

  where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ a translation vector; $\mathbf{R}$ and $\mathbf{t}$ are often referred to as the camera pose; the origin of the camera reference system has its origin located in the center of projection, and the 3D coordinates of the center of projection in the world reference system can be easily computed by the relation : $\mathbf{0} = \mathbf{R}\mathbf{C} + \mathbf{t}$, so that $\mathbf{C} = -\mathbf{R}^{-1}\mathbf{t}$. Adopting a widely employed convention in Computer Vision, we suppose that the z-direction of the camera reference system corresponds to the optical axis, and that points with a positive z value lie in front of the camera.

- **Camera Matrix Multiplication**: the 3D point in the camera reference system is multiplied by the so-called *internal camera matrix* $\mathbf{K}$, that contains the camera internal parameters (or camera intrinsic parameters):

$$\tilde{\mathbf{x}} = [U, V, Z]^\top = \mathbf{K}\mathbf{X}_{cam}; \tag{3.2}$$

the internal camera matrix only depends on the camera physical and mechanical properties, and it is structured as follows:

$$\mathbf{K} = \begin{bmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}; \tag{3.3}$$

f_u and f_v are the scale factors from the world/camera reference systems to the image reference system, in the u and v- coordinates directions respectively; they are proportional to the focal length of the camera. Pixel $\mathbf{c} = [c_u,\ c_v]^\top$, also called the principal point, represents the intersection between the optical axis and the image plane in the image reference system. s is called the skew, and it's non zero only if the axis of the image reference system are non-perpendicular. All the coefficients of the internal camera matrix only depend on the camera physical properties, so they are usually estimated offline during a *calibration step* and supposed to be known when performing 3D tracking. For more details about internal calibration, see, for example, [95].

- **Projection**: Finally, the 2 coordinates of the pixel on the image are given by dividing the first 2 coordinates of $\tilde{x}$ by the third coordinate:

$$\mathbf{x} = \mathcal{P}(\mathbf{X}) = [u, v]^\top = [U/Z, V/Z]^\top. \tag{3.4}$$

Notice that the first step only depends on the camera pose in the 3D world reference system; the second only depends on the camera physical properties, and the third directly originates from the perspective projection model.

We point out that, due to Equation (3.4), inverting the projection $\mathcal{P}(\cdot)$ without further constraints is only possible up to a scale factor. Moreover, there are 2 distinct non-linearities arising in the perspective projection $\mathcal{P}(\cdot)$: the first is the division of Equation (3.4); the second one is concealed in Equation (3.1): the rotation matrix $\mathbf{R}$ has 9 coefficients, but only 3 degrees of freedom. So, when estimating the camera pose, it usually suited to employ some minimal parametrizations for rotations, such as the Euler angles or the exponential map representation described in Section3.5, instead of dealing directly with the 9 unknown matrix coefficients; unfortunately, the mapping from minimal rotation parametrizations to rotation matrices is non-linear.

The alternative camera models presented in the next section provide affine approximations of the full perspective model.

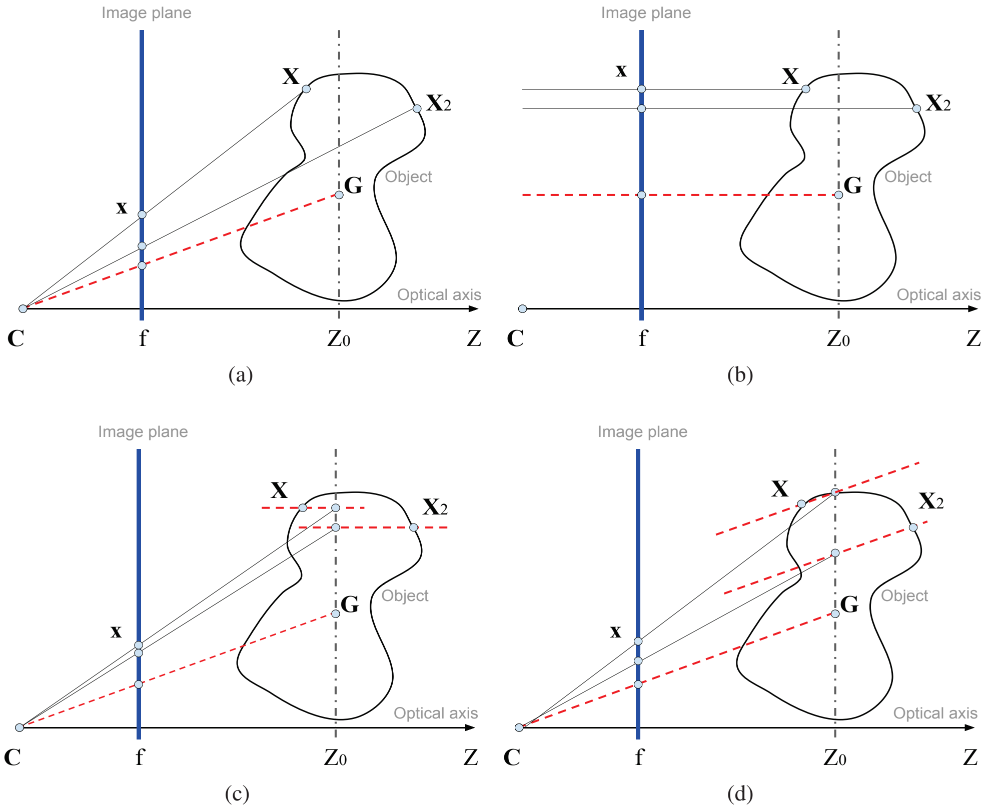


Figure 3.2 – The full perspective projection model (a) and its approximations. (b) Orthographic projection. (c) Weak perspective projection. (d) Para-perspective projection. A point $\mathbf{X}$ on an object is mapped into a pixel $\mathbf{x}$ on the image plane; the plane $Z = Z_0$ intersects the object center $\mathbf{G}$; the projection of $\mathbf{G}$ and another object point $\mathbf{X}_2$ are shown on the image plane, as well.

3.3 Approximate Camera Models

Affine perspective models have been introduced for approximating the full perspective projection model, simplifying the computations. Their main drawback is their limited domain of application: they can only be employed under some specific hypothesis, detailed below, that do not generally hold for the user cases considered in this work.

For a survey and in-depth analysis on several camera models see, for instance, [94]. We compare three affine camera projection models with the full perspective model introduced above; for sake of clarity, we consider a simplified framework where $\mathbf{R} = \mathbf{I}$ and $\mathbf{t} = \mathbf{0}$, so that $\tilde{\mathbf{X}} = \mathbf{X}$; moreover, we assume that the internal calibration matrix is of the following form:

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

In this case, depicted in Figure 3.2-(a), the image coordinates of the full perspective projection of a 3D point $\mathbf{X} = [X, Y, Z]^\top$ are given by:

$$u = f\frac{X}{Z}, \qquad v = f\frac{Y}{Z}. \tag{3.6}$$

Generalization to other cases is straightforward.

Orthographic Projection This approximation is the simplest projection model, in which the X and Y components of the 3D points are straightforwardly mapped to the u and v coordinates on the image plane:

$$u = X, \qquad v = Y. \tag{3.7}$$

This model does not even take into account the scaling objects undergo when projected on images.

Weak perspective Projection In this affine projection model, the image coordinates are given by

$$u = f\frac{X}{Z_0}, \qquad v = f\frac{Y}{Z_0} \tag{3.8}$$

where f/Z_0 is a fixed scaling factor. This projection model can be interpreted as an orthographic projection on an intermediate plane parallel to the image plane, the *reference plane*, followed by a scaling from the reference plane to the image plane. This model is usually employed for approximating the full perspective model for *objects lying far from the camera center* and *close to the optical axis*, taking Z_0 as the mean depth of the object points.

Para-perspective Projection This affine projection model can be interpreted, similarly as the weak perspective model, as a projection on an intermediate plane parallel to the image plane, the *reference plane*, followed by a scaling from the reference plane to the image plane. for the weak perspective model, the projection from the object to the reference plane is done along parallel lines, but in this case they are not parallel to the optical axis. The para-perspective projection model gives a good approximation of the full perspective model for objects that lie far from the camera but not necessarily close to the optical axis. In this case, after defining a a point $\mathbf{X}_0 = [X_0, Y_0, Z_0]^\top$, that can be thought as the center of the object points, the para-perspective projection of each object point $\mathbf{X} = [X, Y, Z]^\top$ would be:

$$u = \frac{f}{Z_0}\left(X - \frac{X_0}{Z_0}(Z - Z_0)\right) \qquad v = \frac{f}{Z_0}\left(Y - \frac{Y_0}{Z_0}(Z - Z_0)\right), \tag{3.9}$$

that is, lines of the linear projection on the reference plane would be parallel to the line passing by the camera center $\mathbf{C}$ and $\mathbf{X}_0$. It can be shown that the weak and the para-perspective model are finite order developments of the full perspective projection, respectively of zero-th and first order.

One interesting feature shared by all the described models is their affine structure: more in particular, taking an affine projection $\mathcal{P}_a(\cdot)$, an arbitrary 3D point $\mathbf{X}_{ref}$ and its representation on the image plane $\mathbf{x}_{ref} = \mathcal{P}_a(\mathbf{X}_{ref})$, then for any 3D point $\mathbf{X}$:

$$\mathcal{P}_a(\mathbf{X} + \mathbf{X}_{ref}) = \mathbf{x}_{ref} + \mathcal{P}_a(\mathbf{X}); \tag{3.10}$$

we refer to this property as translation invariance, and further discuss its importance in Chapter 6. Translation invariance, of course, does not hold for the full perspective model.

3.4 Distortion Models

As pointed out above, the perspective camera model describes the image formation process for a pinhole camera; more in particular, it is a *rectilinear projection*, where straight lines in the scene are mapped to straight lines on the image. The same holds for its approximations described in Section 3.3. The use of photographic lenses introduces a deviation from the rectilinear projection, called *distortion*. Wide angle lenses especially suffer from this phenomenon.

Distortion is usually modeled as a 2D image deformation taking place after the image formation by perspective projection; the distortion parameters are usually estimated during the camera calibration step, so that, at run time, the distortion on acquired images is compensated and the perspective projection model is used on the un-distorted image. Let $\mathbf{x}_{dist} = [u_{dist}, v_{dist}]^\top$ be an observed, distorted pixel, and $\check{\mathbf{x}}_{dist} = [\check{u}_{dist}, \check{v}_{dist}]^\top$ its *normalized representation*, so that :

$$\begin{aligned} u_{dist} &= c_u + f_u \check{u}_{dist} \\ v_{dist} &= c_v + f_v \check{v}_{dist} \end{aligned} \tag{3.11}$$

(3.12)

where $c_u,\ f_u,\ c_v,\ f_v$, are coefficient of the internal calibration matrix of Equation (3.3). Let $\mathbf{x} = [u, v]^\top$ and $\check{\mathbf{x}} = [\check{u}, \check{v}]^\top$ be the corresponding un-distorted pixels. The distortion is usually separated in 2 different component, radial and tangential distortion,such that:

$$\check{\mathbf{x}}_{dist} = \check{\mathbf{x}} + \partial\mathbf{x}_{radial} + \partial\mathbf{x}_{tangential}. \tag{3.13}$$

The radial distortion is usually modeled as:

$$\partial\mathbf{x}_{radial} = \Big(1 + k_1 r^2 + k_2 r^4 + \dots\Big)\check{\mathbf{x}}, \tag{3.14}$$

where $r = ||\check{\mathbf{x}}|| = \sqrt{\check{u}^2 + \check{v}^2}$. The tangential distortion can be expressed as:

$$\partial\mathbf{x}_{tangential} = \begin{bmatrix} 2p_1\check{u}\check{v} + p_2(r^2 + 2\check{u}^2) \\ p_1(r^2 + 2\check{v}^2) + 2p_2\check{u}\check{v} \end{bmatrix}. \tag{3.15}$$

As pointed out above, the distortion parameters $k_1,\ k_2,\ p_1,\ p_2$ are usually estimated during the internal calibration phase.

3.5 Exponential Map Parametrization for 3D Rotations

Throughout this work, we employ the same minimal representation for 3D rotations, given by the so-called *exponential map* representation. Given an orthonormal basis of $\mathbb{R}^3$, the set of 3D rotations $SO(3)$ is represented by all the square matrices $\mathbf{R} \in \mathbb{R}^{3\times 3}$ such that:

$$\mathbf{R}\mathbf{R}^\top = \mathbf{I}, \qquad \det(\mathbf{R}) = 1, \tag{3.16}$$

where $\mathbf{I}$ is the 3×3 identity matrix and $\det(\cdot)$ denotes the matrix determinant. $SO(3)$ forms a group under the operation of composition.

Alternatively, a rotation of the 3D point $\mathbf{X}$ around axis represented by the one-norm vector $\mathbf{v} \in \mathbb{R}^3$ by an angle of θ radians can be represented by the 3D vector $\theta\mathbf{v}$. This representation, referred to as *axis-angle representation*, has the advantage of being a minimal parametrization, since it only has 3 degrees of freedom as opposed to the 9 coefficients of a rotation matrix or the 4 components of a unit quaternion. On the other hand, the matrix representation is the most suited one for applying a rotation to a 3D vector. Other representations can be used for representing 3D rotations, such as unit quaternions and euler angles; we refer to the excellent survey [96] for a thorough comparison of the different representations, while we describe more in detail the *axis-angle representation* here, since it is our parametrization of choice.

The exponential map is the non-linear mapping from the axis-angle representation to the rotation matrices; more in particular, given a a rotation vector $\theta\mathbf{v} = \theta[v_x, v_y, v_z]^\top$ such that $||\mathbf{v}|| = 1$, the corresponding rotation matrix is given by:

$$\mathbf{R} = \begin{bmatrix} 2(v_x^2-1)s^2+1 & 2v_xv_ys^2-2v_zcs & 2v_xv_zs^2+2v_ycs \\ 2v_xv_ys^2+2v_zcs & 2(v_y^2-1)s^2+1 & 2v_yv_zs^2-2v_xcs \\ 2v_xv_zs^2-2v_ycs & 2v_yv_zs^2+2v_xcs & 2(v_z^2-1)s^2+1 \end{bmatrix}, \tag{3.17}$$

where $s = \sin(\theta/2)$ and $c = \cos(\theta/2)$. The null rotation ($\theta = 0$) is mapped to $\mathbf{R} = \mathbf{I}$. The inverse mapping, also referred to as log map, allows to pass from the matrix representation to the axis-angle representation:

$$\theta = \text{acos}\left(\frac{\text{trace}(\mathbf{R}) - 1}{2}\right) \tag{3.18}$$

$$\mathbf{v} = \frac{1}{2\sin(\theta)} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}, \tag{3.19}$$

where trace$(\cdot)$ denotes the sum of the diagonal elements of a matrix and R_{ij} is the element of the $i-$th row ang $j-$th column of $\mathbf{R}$. With a slight and widely employed abuse of terminology, in the remainder of this work we will refer interchangeably to the "exponential map representation" of a 3D rotation and to the axis-angle representation. The exponential map representation has been mainly employed for all the work presented in this thesis, thanks to its several advantages over alternative representations:

- it remains free from gimbal lock for rotations of magnitude up to 2π; this makes it suitable for iterative optimization algorithms, provided that the steps at each iteration are small enough.
- As observed above, the exponential map uses a minimal parametrization for SO(3); this makes optimization more efficient and does not require to enforce explicit constraints on the parameters, as it must be done, for instance, when using unit quaternions.

3.6 Conclusion

In this chapter we described the main mathematical tools employed for all the methods proposed in this thesis. More in particular, we adopt a full perspective model despite its non-linear nature, since approximated projection models can not be employed in the scenarios we consider. As for the representation of 3D rotations, several parametrizations are currently employed, such as quaternions, Euler angles and exponential map: arguably, the latter is the most suitable for our purposes, thanks to its nice analytical properties. In the next chapter, we will give a thorough description of dense image alignment methods, the framework employed by the method described in Chapter 5.

4

Dense Methods for Image Alignment and their Application to 3D Tracking

The first method for 3D tracking proposed in this thesis is based on a global image alignment framework and on a robust dense descriptor. In this chapter we present an overview of the main paradigms for global image alignment, while our robust descriptor and quantitative evaluations of our method will be detailed in the next chapter.

Recently, methods for image alignment based on global optimization problems have undergone a regain of interest for their accuracy and robustness, thanks to the growing power of modern computing devices, with applications such as image stitching, pose estimation, optical flow, object tracking, face coding, and others. In this chapter we present a survey on the most common classes of iterative optimization methods proposed since the seminal work of Lucas and Kanade [34]. A first overview of the main optimization schemes was presented in a unifying framework in the excellent survey from Baker and Matthews [7]. Following its footsteps, we present here an updated analysis of the most relevant dense image alignment methods, providing a synthetic description of the existing algorithms, highlighting both theoretical aspects and implementation issues. Some relevant updates are presented with respect to [7], more in particular:

- The Efficient Second-order Method [97], which was absent from [7], is described in the same framework, in Section 4.4.
- Other recent extensions and related methods for dense image alignment, appeared since the publication of [7], are described in Section 4.7.
- We describe a complex, non-linear warp allowing the application of dense image alignment methods to 3D tracking of non-planar targets in Section 4.6, discussing the applicability and implementation issues of each method for this particular application.

All the presented methods allow to estimate a generic, parametrized 2D warp mapping a reference image usually referred to as the *template* to a deformed image. Commonly employed warps include

2D translations, rotations, affine deformations and homographies.

4.1 Dense Image Alignment

Let T, I be 2 images, seen here as functions returning the value of the luminous intensity for a given pixel location $\mathbf{x}$:

$$T, I : \mathbb{R}^2 \rightarrow \mathbb{R}; \quad \mathbf{x} \mapsto T(\mathbf{x}), I(\mathbf{x}). \tag{4.1}$$

T will denote a “reference image”, or “template”; I will denote an input image.

Let $\mathcal{F}$ be a family of warps, parametrized by n parameters stored in a vector $\mathbf{p}$:

$$\mathbf{W} : \mathbb{R}^2 \times \mathbb{R}^n \rightarrow \mathbb{R}^2, \quad (\mathbf{x}, \mathbf{p}) \mapsto \mathbf{W}(\mathbf{x}, \mathbf{p}). \tag{4.2}$$

Furthermore, we introduce an explicit notation for the warped image $I_{\mathbf{p}}$:

$$I_{\mathbf{p}}(\mathbf{x}) = I(\mathbf{W}(\mathbf{x}, \mathbf{p})) \quad \forall \mathbf{x} \in \mathcal{D}, \tag{4.3}$$

where $\mathcal{D} \subset \mathbb{R}^2$ the domain of the reference image T.

As shown in Figure 4.1, image alignment consists in estimating the parameters $\mathbf{p}_I$ of the warp mapping the reference image T (*template*) over an input image I, such that $T(\mathbf{x}) = I_{\mathbf{p}_I}(\mathbf{x}) = I(\mathbf{W}(\mathbf{x}, \mathbf{p}_I))$.

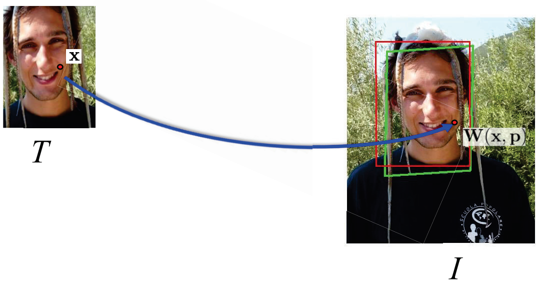


Figure 4.1 – The image alignment problem.

This problem can be modelled through the following optimization problem:

$$\mathbf{p}_I = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \Big(I(\mathbf{W}(\mathbf{x}, \mathbf{p})) - T(\mathbf{x}) \Big)^2, \tag{4.4}$$

where the sum is extended over all, or a dense subset of, the pixels of the template. Even for very simple warps, the optimization problem (4.4) is highly non-linear because of the presence of the functions $T(\cdot), I(\cdot)$, and it is usually solved by iterative methods. On the other hand, this approach does not need to detect nor match any local image features, which makes dense methods particularly suited for aligning images with low textures or repetitive patterns. Examples of frequently employed families of warps are 2D and 3D translations, in-plane and out-of-plane rotations, affine warps, and homographies.

For sake of clarity, in this chapter we only consider mono-channel images, e.g. grayscale images; straightforward generalization of the presented methods to multi-channel images, such as RGB or RGB-D images, is described in Appendix A.5.

4.2 Optimization Framework

All the iterative algorithms presented in this chapter share the same basic structure. At iteration c, given the current estimate of the parameters $\mathbf{p}_c$, they:

1. seek for an increment $\delta\mathbf{p}$ of the current parameters, that minimizes a non-linear objective function $F(\delta\mathbf{p})$ somehow related to Equation (4.4);
2. approximate the non-linear objective function using a finite order development, and obtain a closed-form formula for $\delta\mathbf{p}$;
3. update the current estimate of the parameters using the computed value of $\delta\mathbf{p}$.

The above steps are iterated until some stopping criterion is met, such as $||\delta\mathbf{p}|| < \epsilon_{tol}$ for a threshold ϵ_{tol} selected by the user. The methods differ because of the kind of objective function employed (*additive* or *compositional*), because of the direction of the warping (*forward* or *inverse*), and because of the kind of approximation employed (a second order development for the Efficient Second-order Method, a first-order one for the other methods).

Depending on the different choices, the methods can be applied to different classes of warps, and have different computational costs and convergence properties. This is discussed in the next sections.

Comparative tables of all the methods described in this chapter are provided in Appendix A.6.

4.3 First Order Methods

The methods presented in this section employ a first-order development of several variants of the objective function on Equation (4.4), as opposed to the Efficient Second-order Method described in Section 4.4.

4.3.1 Forward Additive Algorithm

The most widespread dense image alignment method is the well-known Lucas-Kanade algorithm proposed in [34]. At iteration c, given a current estimate of the parameters $\mathbf{p}_c$, we seek for an increment of the parameters $\delta\mathbf{p}$ that minimizes the approximation of the objective function:

$$F_{FA}(\delta\mathbf{p}) = \sum_{\mathbf{x}} (I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c + \delta\mathbf{p})) - T(\mathbf{x}))^2. \tag{4.5}$$

According to the classification proposed in [7], this is a *forward* method, because it seeks for an update of the warp of the image, and it is *additive*, because the update is summed to the current estimate of the parameters. The outline of this Forward Additive algorithm is schematically represented in Figure 4.2.

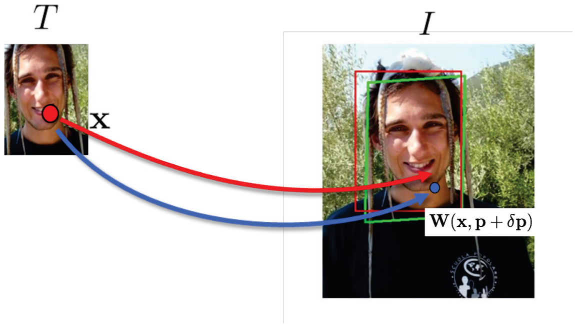


Figure 4.2 – The Forward Additive algorithm. The current warp $\mathbf{W}(\mathbf{x}, \mathbf{p})$ for a pixel $\mathbf{x}$ is represented as a red arrow, the updated warp as a blue arrow.

Approximate Solution: Equation (4.5) is a non-linear function with respect to $\delta\mathbf{p}$, so we approximate it by computing a first-order Taylor expansion of $I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c + \delta\mathbf{p}))$ with respect to the second argument of $\mathbf{W}$ around $\mathbf{p}_c$:

$$F_{FA}(\delta\mathbf{p}) \approx \sum_{\mathbf{x}} (I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) + \mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c)\delta\mathbf{p} - T(\mathbf{x}))^2; \tag{4.6}$$

$\mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c)$ is the $n \times 1$ Jacobian matrix of F_{FA}:

$$\mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c) = \nabla I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) \left.\frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c}, \tag{4.7}$$

where ∇I is the gradient of I; the second term of Equation (4.6) is a quadratic form with respect to $\delta\mathbf{p}$; by setting its derivative equal to zero, it is possible to obtain the following closed-form solution for $\delta\mathbf{p}$:

$$\delta\mathbf{p} = \mathbf{H}(\mathbf{p}_c)^{-1} \sum_{\mathbf{x}} \mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c)^\top (T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c))), \tag{4.8}$$

where $\mathbf{H}(\mathbf{p}_c) = \sum_{\mathbf{x}} \mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c)^\top \mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c)$.

Parameters Update: After computing an increment $\delta\mathbf{p}$ using Equation (4.8), the current estimate of the parameters is updated with the following additive rule:

$$\mathbf{p}_{c+1} = \mathbf{p}_c + \delta\mathbf{p}. \tag{4.9}$$

Assumption on the Set of Warps: The only requirement for the warps $\mathbf{W}(\mathbf{x}, \mathbf{p}) \in \mathcal{F}$ is to be differentiable with respect to $\mathbf{p}$.

Computational Complexity: The main steps of the FA algorithms are summarized in Algorithm 1. The computational complexity of each step is reported as a function of the number of parameters n and the number of pixels of the template N.

Algorithm 1 Forward Additional algorithm

```
while $||\delta\mathbf{p}|| > \epsilon$ do
    Compute $\mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c)$ for all $\mathbf{x} \in \mathcal{D}$ with Equation (4.7)      $\mathcal{O}(nN)$
    Compute $\mathbf{H}(\mathbf{p}_c) = \sum_{\mathbf{x}} \mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c)^\top \mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c)$      $\mathcal{O}(n^2N)$
    Compute $\delta\mathbf{p}$ with Equation (4.8)      $\mathcal{O}(nN + n^3)$
    Update the parameters using Equation (4.9)      $\mathcal{O}(n)$
end while
```

One iteration of the FA algorithm has a computational complexity of:

$$\mathcal{O}(n^2N + n^3) \tag{4.10}$$

Note that, for usual applications, $n \leq 10$, and $N \in [10^3, 10^5]$.

4.3.2 Forward Compositional Algorithm

An alternative algorithm is the Forward Compositional algorithm (FC), proposed in [98]. At iteration c, given the current estimate of the parameters $\mathbf{p}_c$, we seek for an increment of the parameters $\delta\mathbf{p}$ minimizing

$$F_{FC}(\delta\mathbf{p}) = \sum_{\mathbf{x}} (I(\mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p}_c)) - T(\mathbf{x}))^2; \tag{4.11}$$

employing a *compositional* approach instead of the additional approach of Equation (4.5) can lead to better computational performances than the FA algorithm, while, under some assumptions, the convergence properties of the two methods are equivalent, as discussed in Appendix A.2. The update of the warp provided by the FC algorithm is represented in Figure 4.3

Approximate Solution: Assuming that $\mathbf{W}(\mathbf{x}, \mathbf{0}) = \mathbf{x}$, a first-order Taylor expansion of $I(\mathbf{W}(\mathbf{W}(\mathbf{x}, \mathbf{p}), \mathbf{p}_c))$ around $\mathbf{p} = \mathbf{0}$ yields:

$$F_{FC}(\delta\mathbf{p}) \approx \sum_{\mathbf{x}} (I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) + \mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)\delta\mathbf{p} - T(\mathbf{x}))^2, \tag{4.12}$$

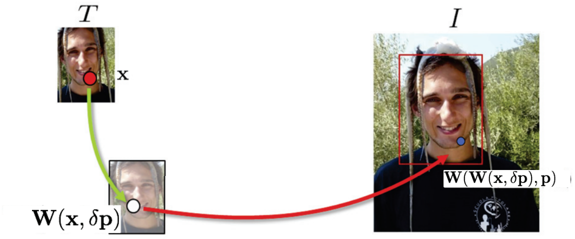

T
$\mathbf{x}$
$\mathbf{W}(\mathbf{x}, \delta\mathbf{p})$
I
$\mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p})$

Figure 4.3 – Updated warp in the Forward Compositional algorithm. The warp increment $\mathbf{W}(\mathbf{x}, \delta\mathbf{p})$ is shown as a green arrow.

where

$$\mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c) = \nabla I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) \left. \frac{\partial \mathbf{W}(\mathbf{y}, \mathbf{p}_c)}{\partial \mathbf{y}} \right|_{\mathbf{y}=\mathbf{x}} \left. \frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{0}}, \tag{4.13}$$

where the last term on the right does not depends on $\mathbf{p}_c$ and can be pre-computed.

As done in Section 4.3.1 for the FA algorithm, by deriving the quadratic form of Equation (4.12) with respect to $\delta\mathbf{p}$ and setting the derivative equal to zero yields:

$$\delta\mathbf{p} = \mathbf{H}(\mathbf{p}_c)^{-1} \sum_{\mathbf{x}} \mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)^\top (T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c))), \tag{4.14}$$

where $\mathbf{H}(\mathbf{p}_c) = \sum_{\mathbf{x}} \mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)^\top \mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)$.

Parameters Update: Once an approximate solution $\delta\mathbf{p}$ for the minimization problem (4.11) has been found with Equation (4.14), the *warp* is updated with the compositional rule:

$$\mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1}) = \mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p}_c). \tag{4.15}$$

If an explicit expression for $\mathbf{p}_{c+1}$ from Equation (4.15) cannot be found, then it is possible to estimate it, for instance, by computing $\mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1})$ for a subset of the pixels and then fitting a regression model to the correspondences $\{\mathbf{x} \leftrightarrow \mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1})\}$.

Assumptions on the Set of Warps: In order to compute the updated warp at each iteration, the composition of 2 admissible warps must be an admissible warp, that is, $\mathcal{F}$ must be closed with respect to composition. Moreover, in order to compute $\mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)$, all warps $\mathbf{W} \in \mathcal{F}$ must be differentiable. Finally, we require that the identity is an admissible warp, so that $\mathbf{W}(\mathbf{x}, \mathbf{0}) = \mathbf{x}$ (after a re-parametrization if needed, as for the warp described in Section 4.6). That is, $\mathcal{F}$ should form a semi-group of differentiable warps.

Computational Complexity: The main steps of the FC algorithms are resumed in Algorithm 2, along with the computational complexity of each step.

Algorithm 2 Forward Compositional algorithm

```
Pre-compute $\left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{0}}$, for all $\mathbf{x} \in \mathcal{D}$                $\mathcal{O}(nN)$
while $||\delta\mathbf{p}|| > \epsilon$ do
    Compute $\mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)$ for all $\mathbf{x} \in \mathcal{D}$ with Equation (4.13)                $\mathcal{O}(nN)$
    Compute $\mathbf{H}(\mathbf{p}_c) = \sum_{\mathbf{x}} \mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)^\top \mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)$    $\mathcal{O}(n^2N)$
    Compute $\mathbf{H}(\mathbf{p}_c)^{-1}$                                        $\mathcal{O}(n^3)$
    Compute $\delta\mathbf{p}$ with Equation (4.14)                                $\mathcal{O}(nN + n^2)$
    Update the parameters using Equation (4.9)                     $\mathcal{O}(n^2)$
end while
```

Despite the fact that some quantities can be pre-computed, one iteration of the FC algorithm has the same computational complexity as the FA algorithm:

$$\mathcal{O}(n^2N + n^3) \tag{4.16}$$

Actually, the complexity of the update of the warp depends on the family of warps, in Algorithm 2 the computational complexity for affine warps ($\mathcal{O}(n^2)$) is reported; for other kinds of warps the computational cost of this step can change, but it usually it does not affect the global complexity estimation for one iteration of the algorithm.

4.3.3 Inverse Compositional Algorithm

A major drawback of the forward algorithms is their heavy computational cost, since the matrix H has to be re-computed at each iteration. A more efficient iterative method, the Inverse Compositional algorithm (IC), has been proposed in [99]. Given the current estimate of the parameters $\mathbf{p}_c$, the IC algorithm seeks an increment $\delta\mathbf{p}$ that minimizes

$$F_{IC}(\delta\mathbf{p}) = \sum_{\mathbf{x}} (T(\mathbf{W}(\mathbf{x}, \delta\mathbf{p})) - I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)))^2; \tag{4.17}$$

note that this objective function is very similar to that of the FC algorithm (4.11), but the roles of the image and the template are switched. Rather than seeking for an incremental warp that makes the warped image more similar to the template, the template is warped to make it more similar to the current warped image. This trick allows to decrease the computational cost. The update of the warp in the IC algorithm is shown in Figure 4.4

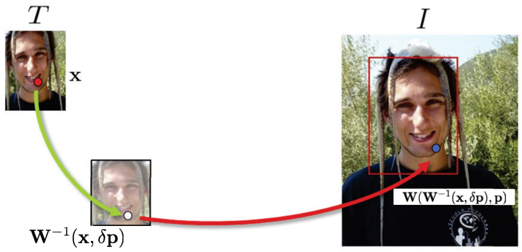

T
$\mathbf{x}$
$\mathbf{W}^{-1}(\mathbf{x}, \delta\mathbf{p})$
I
$\mathbf{W}(\mathbf{W}^{-1}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p})$

Figure 4.4 – Updated warp provided by the Inverse Compositional algorithm. The inverse of the warp increment $\mathbf{W}(\mathbf{x}, \delta\mathbf{p})$ is shown as a green arrow.

Approximate Solution: As in the previous sections, to find an approximated objective function, we perform a first order expansion of $T(\mathbf{W}(\mathbf{x}, \mathbf{p}))$ around $\mathbf{p} = \mathbf{0}$. Assuming that $\mathbf{W}(\mathbf{x}, \mathbf{0}) = \mathbf{x}$ and setting the derivative of the resulting quadratic form equal to zero yields:

$$\delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \mathbf{J}_{IC}(\mathbf{x})^\top [I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) - T(\mathbf{x})], \tag{4.18}$$

where $\mathbf{H} = \sum_{\mathbf{x}} \mathbf{J}_{IC}(\mathbf{x})^\top \mathbf{J}_{IC}(\mathbf{x})$, and:

$$\mathbf{J}_{IC}(\mathbf{x}) = \nabla T(\mathbf{x}) \left. \frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{0}}. \tag{4.19}$$

Note that $\mathbf{J}_{IC}(\mathbf{x})$ and $\mathbf{H}$ do NOT depend on the current parameters estimate and can be pre-computed once and for all at the beginning of the optimization.

Parameters Update: After computing $\delta \mathbf{p}$ with Equation (4.18), the current warp is updated so that:

$$\mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1}) = \mathbf{W}(\mathbf{W}(\mathbf{x}, \delta \mathbf{p})^{-1}, \mathbf{p}_c). \tag{4.20}$$

As for the FC algorithm, if an expression for $\mathbf{p}_{c+1}$ can not be explicitly computed from Equation (4.15), it is possible to compute it by fitting a regression model to the correspondences $\{\mathbf{x} \leftrightarrow \mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1})\}$.

Assumptions on the Set of Warps: As for FC, we assume that the warps $\mathbf{W} \in \mathcal{F}$ are differentiable, that $\mathcal{F}$ is closed with respect to the composition and that the identity is an admissible warp. Moreover, the warps should be invertible and $\mathcal{F}$ should be closed under the inversion. That is, $\mathcal{F}$ must form a group.

Computational Complexity: The main steps of the IC algorithms and their computational complexity are reported in Algorithm 3. As for the FC algorithm, the cost of the update of the warp depends on the family of warps, but generally it does not affect the global complexity estimation. Thanks to the fact that $\mathbf{H}^{-1}$ and $\mathbf{J}_{IC}(\mathbf{x})$ can be pre-computed, the computational complexity for

Algorithm 3 Inverse Compositional algorithm

```
Pre-compute $\mathbf{J}_{IC}(\mathbf{x})$ with Equation (4.19) for all $\mathbf{x} \in \mathcal{D}$        $\mathcal{O}(nN)$
Pre-compute $\mathbf{H} = \sum_{\mathbf{x}} \mathbf{J}_{IC}(\mathbf{x})^\top \mathbf{J}_{IC}(\mathbf{x})$        $\mathcal{O}(n^2 N)$
Pre-compute $\mathbf{H}^{-1}$        $\mathcal{O}(n^3)$
while $||\delta \mathbf{p}|| > \epsilon$ do
    Compute $\delta \mathbf{p}$ with Equation (4.18)        $\mathcal{O}(nN + n^2)$
    Update the parameters using Equation (4.20)        $\mathcal{O}(n^2)$
end while
```

one iteration of the IC algorithm is lower than that of the forward algorithms:

$$\mathcal{O}(nN + n^2). \tag{4.21}$$

Notice that pre-computing $\mathbf{H}^{-1}$ is numerically less stable than solving a linear system involving $\mathbf{H}$ at each iteration, but this does not represent a problem in most part of practical applications.

4.3.4 Inverse Additive Algorithm

An Inverse Additive algorithm (IA) has been proposed in [100], which has as a low computational complexity as the IC algorithm while employing an additive update of the parameters. Unfortunately, as we will see, this algorithm can only be applied to a very restricted set of warps. Given the current estimate of the parameters $\mathbf{p}_c$, the IA algorithm finds an increment $\delta\mathbf{p}$ that approximatively minimizes the same objective function as the Forward Additive algorithm:

$$F_{IA}(\delta\mathbf{p}) = F_{FA}(\delta\mathbf{p}) = \sum_{\mathbf{x}}(I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c + \delta\mathbf{p})) - T(\mathbf{x}))^2. \tag{4.22}$$

Approximate Solution: We start from the first order expansion of $I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c + \delta\mathbf{p}))$ around $\mathbf{p}_c$ of Equation (4.6):

$$F_{IA}(\delta\mathbf{p}) \approx \sum_{\mathbf{x}}(I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) + \mathbf{J}_{IA}(\mathbf{x}, \mathbf{p}_c)\delta\mathbf{p} - T(\mathbf{x}))^2, \tag{4.23}$$

where:

$$\mathbf{J}_{IA}(\mathbf{x}, \mathbf{p}_c) = \mathbf{J}_{FA}(\mathbf{x}, \mathbf{p}_c) = \nabla I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) \left.\frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c}. \tag{4.24}$$

Assuming that the current parameters estimate is approximately correct, that is, $I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) \approx T(\mathbf{x})$, the following approximation holds:

$$\frac{\partial I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c))}{\partial \mathbf{x}} = \nabla I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) \frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p}_c)}{\partial \mathbf{x}} \approx \nabla T. \tag{4.25}$$

Injecting this approximation in Equation (4.24), yields:

$$\mathbf{J}_{IA}(\mathbf{x}, \mathbf{p}_c) = \nabla T(\mathbf{x}) \left(\frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p}_c)}{\partial \mathbf{x}}\right)^{-1} \left.\frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c}. \tag{4.26}$$

Minimizing the quadratic form (4.23), we find the closed-form solution for $\delta\mathbf{p}$:

$$\delta\mathbf{p} = \mathbf{H}(\mathbf{p}_c)^{-1} \sum_{\mathbf{x}} \mathbf{J}_{IA}(\mathbf{x}, \mathbf{p}_c)^\top (I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) - T(\mathbf{x})), \tag{4.27}$$

where $\mathbf{H}(\mathbf{p}_c) = \sum_{\mathbf{x}} \mathbf{J}_{IA}(\mathbf{x}, \mathbf{p}_c)^\top \mathbf{J}_{IA}(\mathbf{x}, \mathbf{p}_c)$. For keeping notations coherent, in the above formula we replaced $(I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) - T(\mathbf{x}))$ with $(T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c))$. This entails a change of sign of $\delta\mathbf{p}$, so that, when updating the parameters estimate, the value of $\delta\mathbf{p}$ will be subtracted from the current parameters estimate rather than added. To compute $\delta\mathbf{p}$ efficiently, we assume that:

$$\left(\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{p}_c)}{\partial\mathbf{x}}\right)^{-1}\left.\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{p})}{\partial\mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c}=\Gamma(\mathbf{x})\Sigma(\mathbf{p}_c) \tag{4.28}$$

where $\Gamma(\mathbf{x})$ is a $2\times k$ matrix that is only function of the pixels coordinates on the template and $\Sigma(\mathbf{p}_c)$ is a $k\times n$ matrix depending on the current estimate of the parameters, for some integer $k>0$. Then, matrix $\mathbf{H}(\mathbf{p}_c)$ can be re-written as:

$$\mathbf{H}(\mathbf{p}_c)=\Sigma(\mathbf{p}_c)^\top\mathbf{H}_*\Sigma(\mathbf{p}_c), \tag{4.29}$$

where:

$$\mathbf{H}_*=\sum_{\mathbf{x}}(\nabla T(\mathbf{x})\Gamma(\mathbf{x}))^\top(\nabla T(\mathbf{x})\Gamma(\mathbf{x})). \tag{4.30}$$

For simplicity's sake, we assume here that $k=n$ and that $\Sigma(\mathbf{p}_c)$ is invertible (see [100] for the general case $k\neq n$); then, the inverse of $\mathbf{H}(\mathbf{p}_c)$ becomes:

$$\mathbf{H}^{-1}(\mathbf{p}_c)=\Sigma(\mathbf{p}_c)^{-1}\mathbf{H}_*^{-1}\Sigma(\mathbf{p}_c)^{-T}; \tag{4.31}$$

and the expression in Equation (4.27) reduces to:

$$\delta\mathbf{p}=\Sigma(\mathbf{p}_c)^{-1}\mathbf{H}_*^{-1}\sum_{\mathbf{x}}(\nabla T(\mathbf{x})\Gamma(\mathbf{x}))^\top(I(\mathbf{W}(\mathbf{x},\mathbf{p}_c)-T(\mathbf{x})). \tag{4.32}$$

This formula yields an update of the parameters with a lower computational complexity than the forward algorithms, but under the very restrictive assumption that the composition of Equation (4.28) can be explicitly computed.

Parameters Update: After computing $\delta\mathbf{p}$ with Equation (4.32), the current estimate of the warp is updated with the additive rule:

$$\mathbf{p}_{c+1}=\mathbf{p}_c-\delta\mathbf{p}, \tag{4.33}$$

where the minus is due to the change of sign introduced in Equation (4.27).

Assumptions on the Set of Warps: We assume that the warps $\mathbf{W}\in\mathcal{F}$ are differentiable; moreover, in order to find an expression for $\mathbf{H}_*$ that does not depend on $\mathbf{p}_c$, one has to explicitly find the decomposition of Equation (4.28). Moreover, in the general case, not only it is difficult to find an explicit decomposition, but it is not even obvious that it exists. This makes the IA algorithm usable with only a very limited set of warps in practical cases. In practice, authors of [100] show that IA algorithm can be employed with 2D translations, 2D affine warps and "a small number of esoteric non-linear warps" [7].

Computational Complexity: The main steps of the IA algorithms are reported in Algorithm 4. The computational complexity of each step is reported as a function of the number of parameters n and the number of pixels of the template N, supposing for sake of simplicity that $k = n$.

Algorithm 4 Inverse Additional algorithm

```
Pre-compute $(\nabla T(\mathbf{x})\Gamma(\mathbf{x}))$ for all the pixels of the template    $\mathcal{O}(nN)$
Pre-compute $\mathbf{H}_*$ using Equation (4.30)                                            $\mathcal{O}(nN)$
Pre-compute $\mathbf{H}_*^{-1}$                                                              $\mathcal{O}(n^3)$
while $||\delta\mathbf{p}|| > \epsilon$ do
    Evaluate $\Sigma\mathbf{p}_c{}^{-1}$                                                     $\mathcal{O}(n^2)$
    Compute $\delta\mathbf{p}$ with Equation (4.27)                                           $\mathcal{O}(nN + n^2)$
    Update the parameters using Equation (4.33)                                              $\mathcal{O}(n)$
end while
```

Assuming that evaluating the matrix $\Sigma(\mathbf{p})^{-1}$ has a computational complexity of $\mathcal{O}(n^2)$, then the computational complexity of an iteration of the IA algorithm is:

$$\mathcal{O}(nN + n^2). \tag{4.34}$$

4.4 A Second-order Method: ESM

All the iterative methods described above are based on a first-order development of the terms appearing in the objective function. A priori, a second-order development should yield a more accurate approximation, but computing the second order derivatives is computationally too expensive for most applications. The Efficient Second-order Method (ESM) proposed in [97] allows to iteratively minimize a second-order approximation of the objective function using only first-order derivatives.

We start from same objective function as in the FC algorithm:

$$F_{ESM}(\delta\mathbf{p}) = F_{FC}(\delta\mathbf{p}) = \sum_{\mathbf{x}} (I(\mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p}_c)) - T(\mathbf{x}))^2; \tag{4.35}$$

Approximate Solution: We perform the following *second-order* development around $\mathbf{p} = \mathbf{0}$:

$$\sum_{\mathbf{x}} (I(\mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p}_c)) - T(\mathbf{x}))^2 \approx \sum_{\mathbf{x}} (I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) + \mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)\delta\mathbf{p} + \frac{1}{2}\delta\mathbf{p}^\top \mathbf{M}\delta\mathbf{p} - T(\mathbf{x}))^2, \tag{4.36}$$

where:

$$\mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c) = \frac{\partial}{\partial \mathbf{q}}\Big(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c))\Big)\Big|_{\mathbf{q}=\mathbf{0}} \tag{4.37}$$

$$\mathbf{M} = \mathbf{M}(\mathbf{x},\mathbf{p}_c) = \frac{\partial^2}{\partial \mathbf{q}^2}\Big(I(\mathbf{W}((\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c))\Big)\Big|_{\mathbf{q}=\mathbf{0}}. \tag{4.38}$$

Now, let us assume that the following property holds for all the warps $\mathbf{W} \in \mathcal{F}$:

$$\begin{gathered}\exists \epsilon > 0 \text{ such that } \forall \delta\mathbf{p} \in \mathbb{R}^n, ||\delta\mathbf{p}|| < \epsilon, \text{ then:}\\ \mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}) = \mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}) \qquad \forall \mathbf{p} \in \mathbb{R}^n.\end{gathered} \tag{4.39}$$

Moreover, we assume here that $I(\mathbf{W}(\mathbf{x},\mathbf{p}_c)) \approx T(\mathbf{x})$ and that $\delta\mathbf{p}$ is the (unknown) parameters increment such that $I(\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c)) = T(\mathbf{x})$.

Under these assumptions, the template jacobian $\frac{\partial}{\partial \mathbf{q}}\Big(T(\mathbf{W}(\mathbf{x},\mathbf{q}))\Big)\Big|_{\mathbf{q}=\mathbf{0}} = \mathbf{J}_{IC}(\mathbf{x})$ can be approximated with a first-order development of the image jacobian $\mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c)$ around $\mathbf{p} = \mathbf{p}_c$, yielding:

$$\mathbf{J}_{IC}(\mathbf{x}) \approx \mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c) + \mathbf{M}\delta\mathbf{p}. \tag{4.40}$$

In fact we have:

$$\begin{aligned}
T(\mathbf{x}) &= I(\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c)) \quad \Leftrightarrow\\
T(\mathbf{x}) &= I(\mathbf{W}(\mathbf{x},\mathbf{p}_c+\delta\mathbf{p})) \quad \Leftrightarrow\\
T(\mathbf{W}(\mathbf{x},\mathbf{0})) &= I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{0}),\mathbf{p}_c+\delta\mathbf{p})) \quad \Leftrightarrow\\
\mathbf{J}_{IC}(\mathbf{x}) &= \frac{\partial}{\partial \mathbf{q}}\Big(T(\mathbf{W}(\mathbf{x},\mathbf{q}))\Big)\Big|_{\mathbf{q}=\mathbf{0}} = \frac{\partial}{\partial \mathbf{q}}\Big(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c+\delta\mathbf{p})))\Big|_{\mathbf{q}=\mathbf{0}} \quad \Leftrightarrow\\
\mathbf{J}_{IC}(\mathbf{x}) &\approx \frac{\partial}{\partial \mathbf{q}}\Big(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c))\Big)\Big|_{\mathbf{q}=\mathbf{0}} + \frac{\partial}{\partial \mathbf{p}}\frac{\partial}{\partial \mathbf{q}}\Big(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p})\Big)\Big|_{\mathbf{p}=\mathbf{p}_c,\ \mathbf{q}=\mathbf{0}}\delta\mathbf{p} \quad \Leftrightarrow\\
\mathbf{J}_{IC}(\mathbf{x}) &\approx \mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c) + \mathbf{M}\delta\mathbf{p},
\end{aligned}$$

From Equation (4.40) we compute the following approximation:

$$\mathbf{M}\delta\mathbf{p} \approx \mathbf{J}_{IC}(\mathbf{x}) - \mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c); \tag{4.41}$$

by employing it into the second-order development of the objective function of Equation (4.36), we obtain a second-order approximation of the objective function, which can be computed using only first-order derivatives of the warp:

$$\sum_{\mathbf{x}}(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c))-T(\mathbf{x}))^2 \approx \sum_{\mathbf{x}}(I(\mathbf{W}(\mathbf{x},\mathbf{p}_c))+\mathbf{J}_{ESM}(\mathbf{x},\mathbf{p}_c)\delta\mathbf{p}-T(\mathbf{x}))^2, \quad (4.42)$$

where:

$$\mathbf{J}_{ESM}(\mathbf{x},\mathbf{p}_c) = \frac{\mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c)+\mathbf{J}_{IC}(\mathbf{x})}{2} = \left(\frac{\nabla I_{\mathbf{p}_c}(\mathbf{x})+\nabla T(\mathbf{x})}{2}\right)\left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{0}}. \quad (4.43)$$

The minimum of the quadratic form of Equation (4.42) is reached for:

$$\delta\mathbf{p} = \mathbf{H}^{-1}(\mathbf{p}_c)\sum_{\mathbf{x}}\mathbf{J}_{ESM}(\mathbf{x},\mathbf{p}_c)(I(\mathbf{W}(\mathbf{x},\mathbf{p}_c))-T(\mathbf{x})); \quad (4.44)$$

where $\mathbf{H}(\mathbf{p}_c) = \sum_{\mathbf{x}}\mathbf{J}_{ESM}(\mathbf{x},\mathbf{p}_c)^\top\mathbf{J}_{ESM}(\mathbf{x},\mathbf{p}_c)$.

Parameters Update: After computing a $\delta\mathbf{p}$ minimizing an approximation of $F_{ESM}(\delta\mathbf{p})$ based on a *second order development*, we can update the warp as in the FC algorithm, using Equation (4.15):

$$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1}) = \mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c). \quad (4.45)$$

Assumptions on the Set of Warps: The same assumptions as the FC algorithm should hold, more in particular that the warps $\mathbf{W} \in \mathcal{F}$ are differentiable, that the identity is an admissible warp and that $\mathcal{F}$ is closed with respect to the composition.

The assumption that the current estimate of the parameters is approximately exact and that $I(\mathbf{W}(\mathbf{x},\mathbf{p}_c)) \approx T(\mathbf{x})$ guarantees that the finite order development of Equation (4.40) is valid, and that the parameters increment is the same as the one in Equation (4.36).

Moreover, in order to write the ESM parameters update equations, the cumbersome assumption (4.39) has to hold. Notice that this assumption does not only depend on the family of warps, but also on the parametrization chosen. If assumption (4.39) does not hold but $\mathcal{F}$ is a group of differentiable warps, the formula provided by ESM is still correct up to the first-order, as shown in Appendix A.4.

Computational Complexity: The main steps of the ESM and their computational complexities are resumed in Algorithm 5.

So, the computational complexity of one iteration of the ESM is the same as the forward algorithms:

$$\mathcal{O}(n^2N+n^3), \quad (4.46)$$

Algorithm 5 Efficient Second-order Method

```
Pre-compute $\left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{0}}$, for all $\mathbf{x} \in \mathcal{D}$                  $\mathcal{O}(nN)$
Pre-compute $\nabla T$                                                                          $\mathcal{O}(N)$
while $||\delta\mathbf{p}|| > \epsilon$ do
    Compute $\mathbf{J}_{ESM}(\mathbf{x},\mathbf{p}_c)$ for all $\mathbf{x} \in \mathcal{D}$ with Equation (4.43)           $\mathcal{O}(nN)$
    Compute $\mathbf{H}(\mathbf{p}_c) = \sum_{\mathbf{x}} \mathbf{J}_{ESM}(\mathbf{x},\mathbf{p}_c)^\top \mathbf{J}_{ESM}(\mathbf{x},\mathbf{p}_c)$      $\mathcal{O}(n^2N)$
    Compute $\mathbf{H}(\mathbf{p}_c)^{-1}$                                                      $\mathcal{O}(n^3)$
    Compute $\delta\mathbf{p}$ with Equation (4.44)                                             $\mathcal{O}(nN + n^2)$
    Update the parameters using Equation (4.45)                                         $\mathcal{O}(N)$
end while
```

while, if the family of warps respects the assumptions specified above, ESM provides a more precise parameters update, thus converging in less iterations.

4.5 Choice of the Appropriate Algorithm. Additive vs Compositional Approach

The algorithms described above mainly differ for their computational cost, for the assumptions made on the set of warps and for the accuracy of the approximation of the objective function. While in general cases each algorithm computes a different parameters update, it can be shown that the 4 first-order algorithms are equivalent [7], in the following sense:

> at a given iteration c, the 4 first-order algorithms provide the same updated warp $\mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1})$, up to a first-order development in the second argument of the warp.

In Appendix A.2 we show the equivalence of FA, FC and IC (we don't treat the case of IA since it's limited interest for practical applications, the interested reader may refer to [7]). We observe that ESM is equivalent to the other algorithms in the sense specified above, since it provides the same update as FC up to the first order.

The choice of the correct algorithm for a practical problem depends, among others, on the following factors:

- assumptions on the set of warps $\mathcal{F}$;
- comparison of the computational complexity for one iteration;
- comparison of the computational complexity of the updated parameters estimate.

If Assumption (4.39) holds, then ESM should converge in less iterations at a computational cost comparable with that of the first-order forward algorithms. Unfortunately, this seldom happens, with the important exception of the family of homographies (with an opportune parametrization).

As for the first-order methods, since usually FC does not improve much the computational efficiency of FA, the choice is made between FA anc IC algorithms. The latter has a better computational complexity, but it requires to compute the inverse warp and to find an explicit update of the parameters starting from the updated warp, so the effective computational cost should be compared in practical cases.

Another important difference is that, while computing the matrix $\mathbf{J}$, FA uses the gradient of the image, while IC uses the gradient of the template. If for some reason, one between I and T is much more affected by noise than the other, one should choose the algorithm that allows for the less noisy computation of $\mathbf{J}$.

In Section 5.3.3 we will provide an example of how different methods perform on a real scenario for 3D tracking, and further discuss the optimal algorithm choice for this application.

Finally, we observe that, despite the fact that the additive and compositional algorithms yield to equivalent results, they reflect two different ways of interpreting the problem of image alignment, as depicted in Figure 4.5. In the first one, the "additive" point of view, a pixel $\mathbf{x}$ in the system of reference of the template is progressively warped on the image for finding a pixel with the same intensity. In the "compositional" point of view, at each iteration we compare 2 images $T(\mathbf{x})$ and $I_{\mathbf{p}}(\mathbf{x})$ in the same reference system, and iteratively look for an infinitesimal warp $\mathbf{W}(\mathbf{x}, \delta\mathbf{p})$ such that either $I_{\mathbf{p}}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}))$ is closer to $T(\mathbf{x})$ (in FC), or $T(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}))$ is closer to $I_{\mathbf{p}}(\mathbf{x})$ (in IC). This can be done by comparing the formulas of the compositional algorithms, FC and IC. The first term of the jacobian matrix of the FC algorithm reported in Equation (4.13) corresponds to the gradient of the warped image $I_{\mathbf{p}_c}$ defined in Equation (4.3)

$$\nabla I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c)) \frac{\partial \mathbf{W}(\mathbf{y}, \mathbf{p}_c)}{\partial \mathbf{y}}\bigg|_{\mathbf{y}=\mathbf{x}} = \nabla I_{\mathbf{p}_c}(\mathbf{x}). \tag{4.47}$$

So, $\mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)$ can alternatively be computed as:

$$\mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c) = \nabla I_{\mathbf{p}_c}(\mathbf{x}) \frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}}\bigg|_{\mathbf{p}=\mathbf{0}}. \tag{4.48}$$

This expression is very close to that of $\mathbf{J}_{IC}(\mathbf{x})$ reported in Equation (4.19):

$$\mathbf{J}_{IC}(\mathbf{x}) = \nabla T(\mathbf{x}) \frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}}\bigg|_{\mathbf{p}=\mathbf{0}}. \tag{4.49}$$

Moreover, comparing the formulas for the computation of $\delta\mathbf{p}$ for FC (Equation (4.14)):

$$\delta\mathbf{p} = \mathbf{H}(\mathbf{p}_c)^{-1} \sum_{\mathbf{x}} \mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c)^{\top} (T(\mathbf{x}) - I_{\mathbf{p}_c}(\mathbf{x})), \tag{4.50}$$

and for IC (Equation (4.18)):

$$\delta\mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \mathbf{J}_{IC}(\mathbf{x})^{\top} (I_{\mathbf{p}_c}(\mathbf{x}) - T(\mathbf{x})), \tag{4.51}$$

it is possible to see that at a given iteration, the computation of $\delta\mathbf{p}$ is performed exactly in the same way in the 2 compositional algorithms, except that the roles of $T(\mathbf{x})$ and $I_{\mathbf{p}_c}(\mathbf{x})$ are switched. This does not mean that the computed values of $\delta\mathbf{p}$ are the same for the 2 algorithms, they only provide equivalent warps, as shown in Appendix A.2.

In Appendix A.1 we report a simple example of rigid 2D warp for illustrating the differences among the different methods at implementation level. In the next Section, we describe more in detail a family of warps employable for 3D rigid pose estimation, as we will detail in Chapter 5.

4.6 A Warp for 3D Tracking

In this Section, we present the family of warps shown in Figure 4.6, that will be employed in the 3D tracking framework described in Chapter 5 to estimate the 3D pose of an image in a general, non-planar scene by aligning it against a template with known pose.

We employ the perspective projection described in Chapter 2 for mapping a point $\mathbf{X} \in \mathbb{R}^3$ in the 3D world reference system to its representation $\mathbf{x} \in \mathbb{R}^2$ on a picture: $\mathbf{x} = \mathcal{P}(\mathbf{X})$. In this context, the template T is given by an image showing a 3D scene with a known pose $\mathbf{p}_T$. The parameters $\mathbf{p} \in \mathbb{R}^6$ of the warp shown in Figure 4.6 describe the pose of camera for another image I showing the same 3D scene (the actual pose of I is given by $\mathbf{p}_T + \mathbf{p}$, so that $\mathbf{W}(\mathbf{x}, \mathbf{0}) = \mathbf{x}$).

We assume that all the internal parameters of both images are known. So, the pose of the image can be estimated applying the methods described above for aligning T and I through the warp of Figure 4.6. In order to compute the warp, the 3D structure of the scene (e.g. a 3D model) must be known.

The inverse warp is depicted in Figure 4.7.

The warp $\mathbf{W}(\mathbf{x}, \mathbf{p})$ of Figure 4.6 is not continuous with respect to the first argument in regions close to occluding contours, while, for a given pixel $\mathbf{x}$, it is differentiable with respect to $\mathbf{p}$, since

$$\frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}} = \frac{\partial \mathcal{P}(\mathbf{X}, \mathbf{p})}{\partial \mathbf{p}}. \tag{4.52}$$

The composition of warps is easily computable, but the warp is closed with respect to composition only for planar scenes, not in a general case. Moreover, $\mathbf{W}$ is only piece-wise differentiable with respect to the first argument, since its spatial derivative $\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{x}}$ is well-defined only for pixels far from occluding contours and representing locally smooth surfaces of the scene.

Moreover, although the warp is mathematically well-defined for all the pixels of T, a pixel $\mathbf{x}$ on T and its corresponding pixel $\mathbf{W}(\mathbf{x}, \mathbf{p}_I)$ on I may not represent the same 3D point $\mathbf{X} = \mathcal{P}^{-1}(\mathbf{x}, \mathbf{p}_T)$ if $\mathbf{X}$ is not visible in both images; in these cases, the inverse warp does not exist, either.

At iteration c, it is possible to check what are the pixels for which $\mathbf{W}^{-1}$ is well defined, by checking that $\mathbf{W}(\mathbf{W}(\mathbf{x}, \mathbf{p}_c), \mathbf{p}_c)^{-1} = \mathbf{x}$, but in practical cases this is un-necessary. If the poses of T and I are not too far, the direct and inverse warp are well defined for most part of the pixels, and the influence of badly warped pixels is limited.

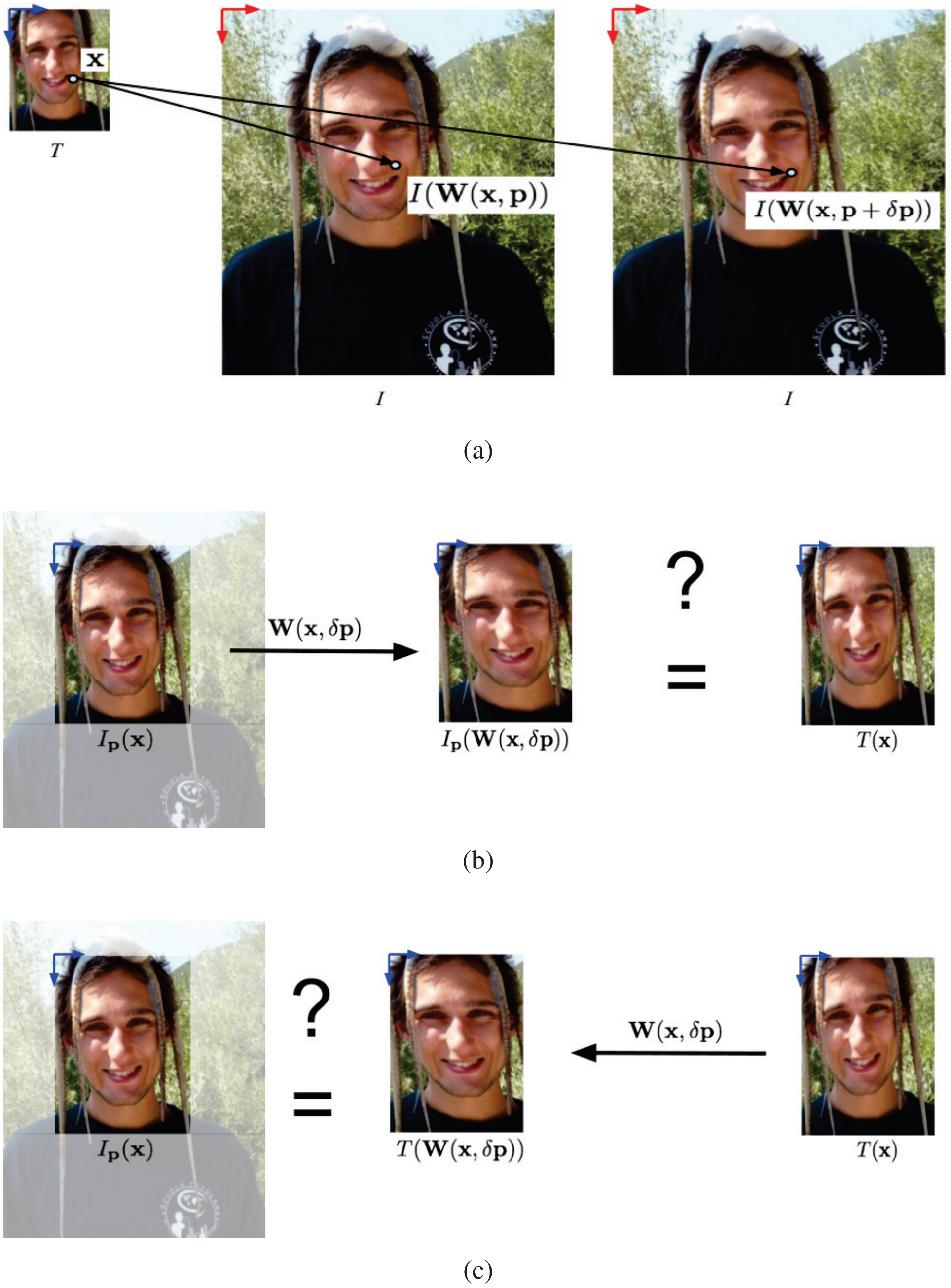

x
T
$I(\mathbf{W}(\mathbf{x},\mathbf{p}))$
$I(\mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}))$
I
I
(a)
$\mathbf{W}(\mathbf{x},\delta\mathbf{p})$
?
=
$I_\mathbf{p}(\mathbf{x})$
$I_\mathbf{p}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}))$
$T(\mathbf{x})$
(b)
?
=
$\mathbf{W}(\mathbf{x},\delta\mathbf{p})$
$I_\mathbf{p}(\mathbf{x})$
$T(\mathbf{W}(\mathbf{x},\delta\mathbf{p}))$
$T(\mathbf{x})$
(c)

Figure 4.5 – Schematic representation of dense image alignment algorithms. (a): FA algorithm. (b): FC algorithm (c) IC algorithm.

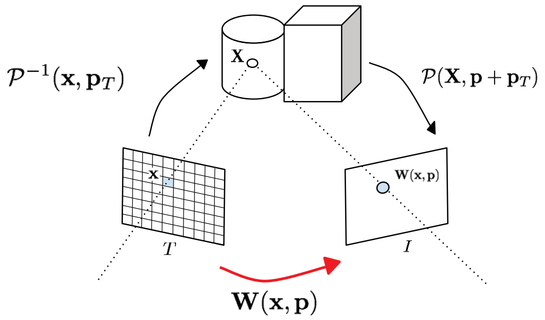

$\mathcal{P}^{-1}(\mathbf{x}, \mathbf{p}_T)$
$\mathbf{X}$
$\mathcal{P}(\mathbf{X}, \mathbf{p} + \mathbf{p}_T)$
$\mathbf{x}$
$\mathbf{W}(\mathbf{x}, \mathbf{p})$
T
I
$\mathbf{W}(\mathbf{x}, \mathbf{p})$

Figure 4.6 – 3D warp between a template T with pose $\mathbf{p}_T$, and an image I, with pose $\mathbf{p}_T + \mathbf{p}$.

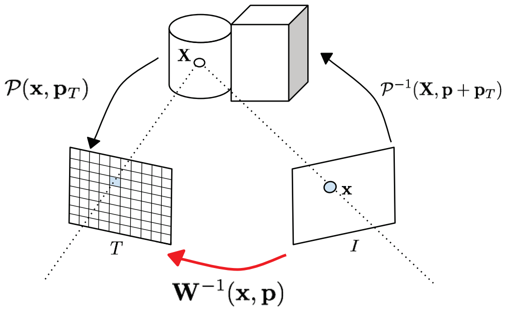

$\mathcal{P}(\mathbf{x}, \mathbf{p}_T)$
$\mathbf{X}$
$\mathcal{P}^{-1}(\mathbf{X}, \mathbf{p} + \mathbf{p}_T)$
$\mathbf{x}$
T
I
$\mathbf{W}^{-1}(\mathbf{x}, \mathbf{p})$

Figure 4.7 – Inverse warp for 3D tracking.

If the template and the image share the same internal calibration matrix then $\mathbf{W}(\mathbf{x}, \mathbf{0}) = \mathbf{x}$. The generalization to the case of different internal calibration matrices is treated in Appendix B.

For the compositional algorithms, at each iteration an explicit estimate of the updated parameters $\mathbf{p}_{c+1}$ is needed, that can not be analytically computed starting from the $\mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1})$. In practical cases, it is possible to compute the updated warp $\mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1})$ with Equation (4.15) or (4.20) for a subset of the pixels of the template; then, since $\mathbf{W}(\mathbf{x}, \mathbf{p}_{c+1}) = \mathcal{P}(\mathbf{X}, \mathbf{p}_T + \mathbf{p}_{c+1})$, an explicit estimate of $\mathbf{p}_{c+1}$ can be computed from the 3D-2D correspondences $\mathbf{X} \leftrightarrow \mathcal{P}(\mathbf{X}, \mathbf{p}_T + \mathbf{p}_{c+1})$ using, for instance, a PnP algorithm.

When computing the updated warp for the IC algorithm with Equation (4.20), the following simplification holds:

$$\begin{aligned}
&\mathbf{W}(\mathbf{W}^{-1}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p}_c) \\
=&\mathcal{P}(\mathcal{P}^{-1}(\mathcal{P}(\mathcal{P}^{-1}(\mathbf{x}, \mathbf{p}_T + \delta\mathbf{p}), \mathbf{p}_T), \mathbf{p}_T), \mathbf{p}_T + \mathbf{p}_c) \\
=&\mathcal{P}(\mathcal{P}^{-1}(\mathbf{x}, \mathbf{p}_T + \delta\mathbf{p}), \mathbf{p}_T + \mathbf{p}_c).
\end{aligned}$$

The IA algorithm is not employable with $\mathbf{W}$, since no decomposition of the form of Equation (4.28) is easily computable.

As for application of ESM algorithm, assumption of Equation (4.39) does not hold, we introduce an error in the second order terms, and ESM just provides a first-order method, as shown in Appendix A.4. Nonetheless, we experimentally observed that applying ESM to 3D tracking often entails a benefice: this can be explained by the fact that it relies on an average of the gradients of the template and the image in the jacobian matrix, so that the influence of noise in the images is reduced.

4.7 Extensions and Related Methods

The methods presented above have been extensively investigated and employed for many applications; their performances have been enhanced by some recent extensions and related methods. Although hundreds of variants exist, and an exhaustive overview would be out of the scope of this thesis, we present here a short description of the most relevant work, referring the interested reader to the references for further reading.

Learnt Descent Directions All the iterative methods described above compute the update of the parameters of the warp with the formula:

$$\delta\mathbf{p} = \alpha\,\mathbf{H}^{-1}\sum_{\mathbf{x}}\mathbf{J}(\mathbf{x})^{\top}\,(T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x},\mathbf{p}_c))), \tag{4.53}$$

where $\mathbf{H} = \sum_{\mathbf{x}}(\mathbf{J}(\mathbf{x})^{\top}\,\mathbf{J}(\mathbf{x}))$, and

$$\alpha = \begin{cases} 1 & \text{for forward algorithms (FA, FC, ESM)} \\ -1 & \text{for inverse algorithms (IA, IC).} \end{cases}$$

The expression of matrix $\mathbf{J}$ for each method is given in Table A.2. The update can be rewritten as:

$$\delta\mathbf{p} = \alpha\mathbf{D}\,\mathbf{r}(\mathbf{p}_c), \tag{4.54}$$

where $\mathbf{r}(\mathbf{p}_c)$ is a $N-$ dimensional column vector, whose component for pixel $\mathbf{x}$ is given by: $(T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x},\mathbf{p}_c)))$, and $\mathbf{D} \in \mathbb{R}^{n\times N}$ is a *descent matrix* that can either depend on the current parameters estimate $\mathbf{p}_c$, as in the FC algorithm, or be constant, for instance in the IC algorithm. In other words, at each iteration the methods presented above solve a regression problem, establishing a relationship between the appearance changes from the template to the image and the changes of the underlying geometric warp.

If the relationship was linear, a single iteration would be sufficient to solve the problem in closed form. Unfortunately, as observed above, even if the considered warp is linear, non-linearities arise because of the presence of the functions $T(\cdot),\ I(\cdot)$, whose gradients must be estimated by finite differences approaches, and iterations become necessary.

Motivated by this observation, some recent works [36, 101, 102, 103] propose to directly learn matrix $\mathbf{D}$ from data in an offline step, employing supervised learning techniques. [36, 101] propose to learn, respectively, a constant and a series of descent matrices using least squares optimization on training data; these methods are usually very effective in reducing the number of iterations required when the initial guess is close enough to the sought optimum, but tend to fail in presence of local optima.

For alleviating this problem, [102] proposes to split the parameters domain around the optimum into several *domains of homogeneous descent*, and learn a series of descent matrices for each domain of homogeneous descent. At testing time, when the choice of the appropriate matrix can not be made deterministically, heuristics can be employed, such as computing parameters updates employing all the matrices and selecting the one that yields the lower residual.

In all these approaches, the information about the employed warp is implicitly taken into account via the provided learning data. In many cases, this may be a sub-optimal strategy, since of the 2 elements contributing to matrix $\mathbf{D}$, that is, the image gradients and the warp jacobians, only the former are noisy and irregular, while the latter can usually be analytically computed. Therefore, the recent [103] proposes an extension of the IC algorithm, where the jacobian of Equation (4.19) is expressed as:

$$\mathbf{J}_{IC}(\mathbf{x}) = \gamma(\mathbf{x}) \frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}}\bigg|_{\mathbf{p}=\mathbf{0}}. \tag{4.55}$$

and the weights $\gamma(\mathbf{x})$ are learned employing supervised learning.

Other methods, instead of learning a regression matrix in Equation 4.54, directly employ non-linear regression models $\delta\mathbf{p} = \mathcal{R}(\mathbf{r}(\mathbf{p}_c))$, where $\mathcal{R}(\cdot)$ is a non-linear function: for instance, [48] employs random forests, showing a remarkable improvements of performances with respect to [36] for tracking of planar surfaces in presence of occlusions.

These learning-based algorithms are shown to achieve better performances for some applications such as face alignment; their major drawback with respect to deterministic methods is the need of an offline learning step, which may be impractical in some situations. More complex regression model can yield to better results; on the other hand, they may involve cumbersome learning steps, delicate parameters tuning, and increased computational complexity; the optimal selection of the algorithm depends on the particular application.

Effective Objective Function All the methods presented above minimize a non-linear least squares loss function of the form (4.4) consisting in a sum of squared residuals (Sum of Squared Differences, or SSD). Some alternatives forms of loss function have been proposed, such as Normalized Cross Correlation (NCC) [104] and Mutual Information (MI) [105, 106, 107, 108] . In [109], extensions of ICA and ESM for NCC and MI are proposed.

According to our experience, the use of NCC does not entail significant improvements over the use of SDD if images are normalized prior to alignment and if suited descriptors are employed instead of the image intensity. While MI has been shown to be the function of choice for multi-modal alignment, MI-based objective functions usually have very well-defined optima in correspondence of the true parameters, also in presence of extreme image nuisances given by light changes, occlusions, etc.; unfortunately, the basin of attraction of these optima is extremely narrow and image smoothing does not always enlarge it [110]; moreover, MI-based optimization is much more computationally intensive than SSD.

Another well-established practice for enhancing robustness consists in employing Mahalanobis distance or a learnt quadratic error function [111] instead of plain SSD. We experimented employing Mahalanobis distance in the approach described in Chapter 4, but no systematic improvement over SSD was obtained in our experimental results.

Employing robust estimators [112] is another common practice to reject outliers by penalizing too large residuals. Unfortunately, robust estimators are far from being a panacea; in addition to slowing down the convergence speed, in some case they may even harm the convergence. Consider the case of tracking poorly textured surfaces, one of the applications of interest for dense image alignment; given the current parameters estimate $\mathbf{p}$, a huge difference between the intensities $I(\mathbf{W}(\mathbf{x}, \mathbf{p}))$ and $T(\mathbf{x})$ could be due to different reasons: for instance, the image appearance may be corrupted, for instance by occlusions, even though $\mathbf{p}$ is a good estimate; or maybe the current estimate $\mathbf{p}$ is not accurate and needs further refinement. It's clear that in the first case the residual should be discarded, while in the second case it should be considered in the optimization, since it would drive the convergence towards the sought optimum much better than the noisy, low residuals corresponding to poorly textured regions. Robust estimators can not discriminate between these opposite situations, so their success highly depend on the considered case.

In [110], we compared different objective functions and descriptors for an image registration task in the domain of non-rigid shape tracking. A qualitative comparison for a 3D tracking task is given in Section 5.3.4.

Robust Dense Descriptors Another prominent research direction consists in replacing the image intensities with densely sampled descriptors [113]. The optimization problem of Equation (4.4) is then replaced by:

$$F(\mathbf{p}) = \sum_{\mathbf{x}} \left\| d(I, \mathbf{W}(\mathbf{x}, \mathbf{p})) - d(T, \mathbf{x}) \right\|^2, \tag{4.56}$$

where $d(I, \mathbf{x})$ is a function that returns a descriptor for location $\mathbf{x}$ in a generic image I.

While computing descriptors originally conceived for keypoint-based approaches, such as SIFT [20] or HOG [114], for densely sampled image locations is effective but intractable for real-time applications, some dense descriptors are suited for real time image alignment, as the Distribution Fields (DFs) [115], our Descriptor Fields [9], or a recent adaption of Bit-Planes proposed in [116]. Replacing the image intensities by other densely sampled descriptors is a good practice we recommend for most cases when employing dense image alignment: even simple, easily computed descriptors can entail a huge performance gain over image intensities at negligible implementation and computational efforts. We will extensively discuss and compare the effectiveness of different descriptors in Chapter 5.

Weighted Sums of Pixels It is interesting to notice that, in the classical formulations of iterative algorithms, the sum of Equation 4.4 is extended to all the pixels of the template, or to a dense, uniformly sampled subset; of course, not all the pixels are equally informative, so more appropriate choices exist.

In the so-called Extended Lucas-Kanade approach [117], the Lucas-Kanade algorithm is re-interpreted in a probabilistic framework for a 2D tracking application. Motivated by the observation that the template usually contains not only the tracking target, but also un-informative background pixels, the optimization is performed in an Expectation-Maximization framework. First, given a discriminative model of the target, obtained, for exampled, running a foreground/background segmentation on the template and computing histogram distributions for the foreground and the background, it is possible to compute a prior distribution for the pixels of the image to belong to the target based exclusively on their appearance, thus obtaining a probability image I_{target}; then, where at each iteration 2 steps are performed:

- Given the current parameters estimate $\mathbf{p}$, the joint probability distribution $P(h_T(\mathbf{x}), h_I(\mathbf{x}))$ is estimated, where

$$h_T(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \text{target on } T \\ 0 & \text{if } \mathbf{x} \in \text{background on } T \end{cases} \qquad h_I(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{W}(\mathbf{x},\mathbf{p}) \in \text{target on } I \\ 0 & \text{if } \mathbf{W}(\mathbf{x},\mathbf{p}) \in \text{background on } I \end{cases}$$

- Given the estimated joint probability of target/background segmentation, an equation similar to Equation (4.5) is optimized, where the sum is weighted by $P(h_T(\mathbf{x}) = 1, h_I(\mathbf{x}) = 1)$ and 2 terms are added in order to maximize the likelihood of warped image pixels to belong to the foreground, based on the pre-computed distribution I_{target}.

In [110] we proposed an alternative approach for 3D tracking of non-rigid surfaces. There, a so-called *relevancy score* $\tilde{\omega}(\mathbf{x})$ is computed for each pixel $\mathbf{x}$ for reducing the influence of occluded and lowly informative regions, and the optimization of Equation (4.4) is replaced by:

$$\mathbf{p}_I = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \tilde{\omega}(\mathbf{x}) \Big(I(\mathbf{W}(\mathbf{x},\mathbf{p})) - T(\mathbf{x}) \Big)^2 . \tag{4.57}$$

More details about this approach are described in Section 5.4.

On the one hand, it is clear that giving different weights to the pixels can be beneficial for the robustness of the tracking. On the other hand, finding the optimal weights and the optimal warp parameters is a chicken-egg problem; moreover, estimating the weights further increases the computational complexity of the method, so the right balance should be sought, between accurate, slow estimations and fast heuristics according to the application and to the computational resources available.

Taking Motion Blur into Account In [118] an extension of ESM is proposed, for more robust tracking in presence of motion blur. Instead of minimizing the objective function (4.35), the

ESM-BLUR algorithms considers a generalized image formation model, taking into account the shutter opening time. More in particular, the following equation is minimized:

$$F_{ESM-BLUR}(\delta\mathbf{p}) = \sum_{\mathbf{x}} \Big(\frac{1}{\Delta t} \int_{\tau \in \Delta t} I(\mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}(\tau)), \mathbf{p}_c)) d\tau - T(\mathbf{x}) \Big)^2, \tag{4.58}$$

whereΔt is the shutter opening time. By assuming a linear motion model for the parameters and taking a second order development as done for Equation (4.36), the same parameters update as Equation (4.44) is obtained, where $\mathbf{J}_{ESM}$ is replaced by the following:

$$\mathbf{J}_{ESM-BLUR}(\mathbf{x}, \mathbf{p}_c) = \frac{\mathbf{J}_{FC}(\mathbf{x}, \mathbf{p}_c) + a(\Delta t)\mathbf{J}_{IC}(\mathbf{x})}{2} \tag{4.59}$$

where $a(\Delta t) \in [1/2, 1]$ is a coefficient depending on the shutter opening time. Moreover, if the latter is unknown, it can be added to the warp parameters and estimated for each incoming frame. We refer the interested reader to [118] for further details.

4.8 Conclusion

In this chapter we presented a survey of the most common dense image alignment methods, and described how they can be employed for 3D tracking. In the next chapter we will introduce the Descriptor Fields, a dense descriptor proposed in [9], that can be employed instead of the image intensities in the optimization in a robust 3D tracking framework, effective even in presence of occlusions complex illumination artifacts.

5

Descriptor Fields for 3D Tracking

As described in Chapter 1, despite a long history of research in 3D tracking, it is still very challenging to reliably register poorly textured, specular objects, and this represents a clear obstacle to the development of Robotics and Augmented Reality applications in industrial environments, where such objects can typically be found.

In this regard, the dense image alignment techniques introduced in Chapter 4 provide an attractive framework,since they globally exploit most of the image information, even when local image features such as interest points or edges are ambiguous.

However, current approaches typically rely on image intensities, which is prone to fail in presence of non-Lambertian effects such as specularities, or when the objects do not exhibit convenient textures. Moreover, a multi-scale approach is usually required for robust alignment, where low-pass filters are applied to the signals to align. When employing image intensities, or a linear combination of them, low-pass filtering rapidly deteriorates information.

In this chapter, we introduce a more robust local descriptor in place of the pixel intensities, that we refer to as “Descriptor Fields”, that can be profitably employed in a 3D tracking framework.

As shown in Figure 5.1, our descriptor allows us to handle challenging imaging artifacts such as a strong lamp light moving in a highly specular, poorly textured environment. Our descriptors are computed from a small set of convolutional filters applied to the images, which makes it suitable for real-time applications. However, instead of relying on the simple linear transformation of the intensity signal issued by the convolutions, we apply a non-linear operation that separates the descriptors’ positive values from the negative ones. Our experimental results show that this operation is crucial for obtaining the best tracking performances.

This can be explained by the fact that, thanks to our non-linear operation, our Descriptor Fields remain discriminant even after low-pass filtering. As a result, large Gaussian kernels can be used to significantly broaden the region of convergence of the alignment optimization algorithms, which is an important factor for robustness.

As a result, Descriptor Fields sensibly enhance performances of global image alignment for 3D

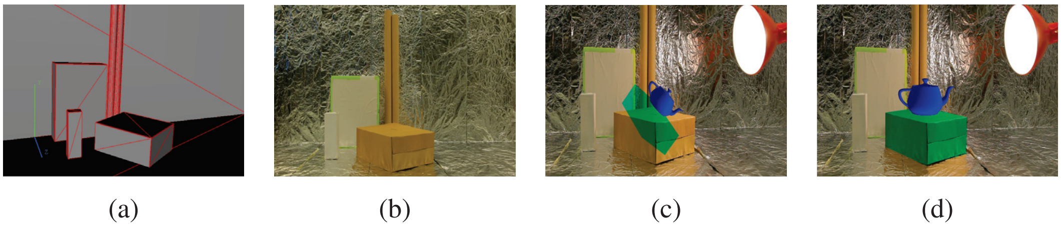

(a) (b) (c) (d)

Figure 5.1 – Given a partial 3D model of the environment such as the one shown in (a), we register the input images by aligning them with one reference view of the environment (b). The virtual teapot and the green virtual box correctly overlaid in the input image show that our approach registered image (d) correctly, despite the strong lamp changing the illumination and partially occluding the scene. By contrast, aligning the images based on the pixel intensities completely fails, as shown in (c).

tracking, as shown in Section 5.3, in presence of heavy light changes, occlusions and other local image artifacts; in Chapter 6 we will introduce a complementary approach, for tracking objects undergoing extreme occlusions in cluttered environments, for which employing a global approach may be problematic.

The matter covered in this chapter was originally published in [9], demonstrated at ISMAR [119] and applied to deformable surfaces tracking in [110].

5.1 3D Tracking via Dense Image Alignment

Our camera registration framework relies on the dense image alignment techniques presented in Chapter 4. In order to enhance convergence, we adopt a multi-scale scheme where optimization is run at coarser scales first, and then refined at finer scales.

5.1.1 Optimization Framework

We assume that we have a partial 3D model of the scene such as the one shown in Figure 5.1(a), and a small set of registered images $\mathcal{T} = \{T_i\}$ of this scene that we refer to as templates. [1] Given an input image I, we estimate the camera pose $\widehat{\mathbf{p}}$ for this image by aligning I with one of the templates T. Information about 3D geometry could as well be available in the form of RGB-D templates, or textured 3D models.

In the remainder of this chapter, we will consider a single template T. Depending on the application, different strategies may be suitable for finding an appropriate T given I: for example, one could select the template that maximizes the normalized cross-correlation with I, as in [85]; in [119] we implemented a real-time tracking approach for planar surfaces based on the warping of a single template, as described in Section 5.4. Alternatively, information about the previously registered

---

[1] We use the semi-automatic ImageModeler software to quickly register the templates in $\mathcal{T}$ and simultaneously obtain the 3D model.

image can be exploited for effective selection of the best template. Finally, when allowed by computational requirements, the optimization can also be performed employing different templates, retaining the one that yields to the smallest residuals.

Alignment is done based on the iterative image alignment framework introduced in Chapter 4 and the warp introduced in Section 4.6. This function backprojects image location $\mathbf{x}$ on the scene 3D model using $\mathbf{p}_T$, the pose for template T, to find its corresponding 3D location, and returns the 2D projection of this 3D location under pose $\mathbf{p} + \mathbf{p}_T$.

The pose parameters $\widehat{\mathbf{p}}$ that transfer the image locations in T to locations in I are estimated by minimizing the objective function introduced in Equation (4.56):

$$F(\mathbf{p}) = \sum_{\mathbf{x}} \Big\| d(I, \mathbf{W}(\mathbf{x}, \mathbf{p})) - d(T, \mathbf{x}) \Big\|^2, \tag{5.1}$$

where $d(J, \mathbf{x})$ is a function that returns a descriptor for location $\mathbf{x}$ in a generic image J; the final estimate is taken as:

$$\widehat{\mathbf{p}} = \underset{\mathbf{p}}{\operatorname{argmin}}\, F(\mathbf{p}). \tag{5.2}$$

The descriptor $d(J, \mathbf{x})$ can be either a scalar, as considered in Chapter 4, or a multi-valued vector. The implementation of the iterative minimization of Equation (5.1) for multi-valued descriptors is explained in Appendix A.5.

In previous dense alignment works, $d(I, \mathbf{x})$ is almost always taken to be $I(\mathbf{x})$, the intensity in image I at location $\mathbf{x}$. The Distribution Fields method [115] considers a function that returns a vector where all values are 0 but one, and which depends on the interval $I(\mathbf{x})$ belongs to. In Section 5.2 we will introduce an alternative dense descriptor for improved performances in presence of complex illumination changes.

The minimization problem of Equation 5.2 can be solved with any of the iterative methods presented in Chapter 4, such as the Lucas-Kanade (LK) algorithm [34, 7], the Inverse Compositional Algorithm (ICA) [7], and the Efficient Second Order Method (ESM) [8].

5.1.2 Multi-scale Optimization

In practice, a multi-scale approach is used to optimize Equation (5.1) by considering the intermediate objective function:

$$F(\mathbf{p}; \sigma) = \sum_{\mathbf{x}} \| D_\sigma(I, W(\mathbf{x}, \mathbf{p})) - D_\sigma(T, \mathbf{x}) \|^2, \tag{5.3}$$

where, for a generic image J, $D_\sigma(J, \mathbf{x})$ is a low-pass version of $d(J, \mathbf{x})$:

$$D_\sigma(J, \mathbf{x}) = G^\sigma * d(J, \mathbf{x}) \tag{5.4}$$

with G^σ a Gaussian kernel of standard deviation σ. The optimization scheme starts with a large value for σ, optimizes $F(\mathbf{p};\sigma)$ to obtain a first estimate $\widehat{\mathbf{p}}$ of the actual pose, decreases σ, optimizes $F(\mathbf{p};\sigma)$ again starting from $\widehat{\mathbf{p}}$, and iterates for a fixed number of iterations.

This multi-scale optimization scheme is important in practice as low-pass filtering increases the basin of convergence of Equation (5.3), but at the same time it degrades the localization of the minimum of the original function in Equation (5.1). In our implementation, the optimization is initialized with the camera pose for the template T. We use 4 scales, with σ initialized to a fixed parameter σ_{max} for the coarsest scale, and divided by 2 between each scale level.

The next section discusses how we compute the d function to improve the convergence when the images exhibit challenging artifacts.

5.2 Descriptor Fields

As mentioned in the previous section, a very common choice for the function $d(I, \mathbf{x})$, which appears in Equation (5.1) and on which image alignment is based, is simply

$$d(I, \mathbf{x}) = I(\mathbf{x}), \tag{5.5}$$

that is, the pixel intensity in image I at location $\mathbf{x}$. However, this option is very sensitive to complex light changes, especially in the absence of texture, as our evaluations presented in the next section will show.

To improve robustness, [115] proposed to use instead a vector of the form:

$$d(I, \mathbf{x}) = [\phi_{I_0 \le I(\mathbf{x}) < I_1}, \phi_{I_1 \le I(\mathbf{x}) < I_2}, \dots, \phi_{I_{K-1} \le I(\mathbf{x}) < I_K}]^\top \tag{5.6}$$

where

$$\phi_b = \begin{cases} 1 & \text{if } b \text{ is true,} \\ 0 & \text{otherwise,} \end{cases} \tag{5.7}$$

for a fixed number of quantization bins K. Thanks to this "explosion" of the image intensities, large Gaussian kernels can be applied as in Equation (5.4) in a multiscale approach to broaden the basin of attraction of the objective function, without blending the intensities together and loosing the image information. Unfortunately, this approach can only handle moderate changes in illumination, and failed on our test sequences.

While they have never been used for direct image alignment —to the best of our knowledge— it seems interesting to use "local jets" for the d function [120, 121, 122, 123]. Local jets are vectors often used as local descriptors and efficiently computed by convolving an image with a series of filters:

$$d(I, \mathbf{x}) = [(\mathbf{f}_1 * I)(\mathbf{x}), \dots, (\mathbf{f}_K * I)(\mathbf{x})]^\top, \tag{5.8}$$

where the $\mathbf{f}_i$ filters are typically Gaussian derivatives kernels. As shown in Figure 5.3, local jets sensibly enhance performances in presence of illumination artifacts; but they do not prevent the low-pass filtering process to degrade the signal information. In order to preserve information under large smoothing, we considered the following sparsifying function, which is at the core of our Descriptor Fields:

$$d(I, \mathbf{x}) = \Big[[(\mathbf{f}_1 * I)(\mathbf{x})]^+, [(\mathbf{f}_1 * I)(\mathbf{x})]^-, \dots, [(\mathbf{f}_K * I)(\mathbf{x})]^+, [(\mathbf{f}_K * I)(\mathbf{x})]^- \Big]^\top, \tag{5.9}$$

where the $[\cdot]^+$ and $[\cdot]^-$ operations respectively keep the positive and negative values of a signal:

$$[x]^+ = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \text{, and} \qquad [x]^- = [-x]^+. \tag{5.10}$$

The descriptor given by Equation (5.9) is a sparsified version of the one of Equation (5.8), since it has twice as many components, but half of its components for each pixel are zero. These operations are simple but fundamental, and we found the last descriptor given by Equation (5.9) to be much more effective than the first version of Equation (5.8), as exemplified in Figure 5.4: when strong Gaussian smoothing is applied by the multiscale optimization described in Section 5.1, the intensity signal flattens making it difficult to align across two images. The same phenomenon happens to the local jet of Equation (5.8), where positive and negative values eliminate each other during the low-pass filtering by a Gaussian kernel. By contrast, the descriptor of Equation (5.9) is much more resilient, and stays discriminant after strong Gaussian smoothing. This yields an objective function with a large basin of attraction and a well localized minimum, which is key for robustness of the alignment.

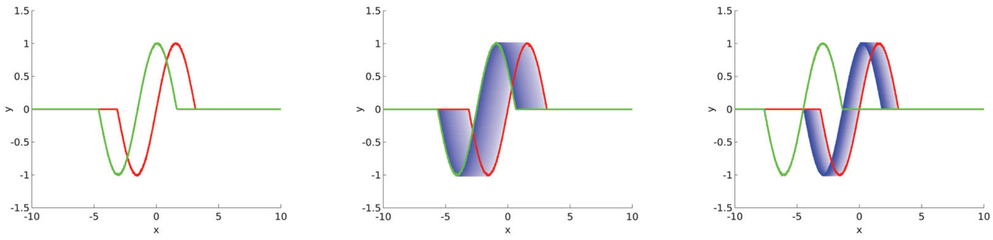
In the next section, we illustrate more in detail the joint effect of smoothing and the non-linear operation of Equation (5.10) for signal alignment on a 1D toy example; in Section 5.3 we report extensive quantitative results on real images.

5.2.1 A 1D Example

In this section we illustrate a simplified example of how the alignment of 2 1D signals benefits from the joint action of signal smoothing and the non-linear, sparsifying function of Equation (5.10).

Consider a 1D signal $y_1(x)$ and its noisy translated version $y_2(x) = y_1(x + p) + \chi(x)$ depicted in Figure 5.2 (a), where $\chi(x)$ is a white noise of standard deviation 0.025; samples of both signals at regular intervals are stored in the column vectors $\mathbf{y}_1$ and $\mathbf{y}_2$. It is possible to retrieve the true value of the translation parameter p applying a Gauss-Newton iterative scheme, equivalent to the Lucas-Kanade algorithm described in Section 4.3.1. After setting an initial guess $p_0 = 0$, at each iteration c we update our guess as $p_c = p_{c-1} + \delta p$, with :

$$\delta p = (\mathbf{J}_{FA}^\top \mathbf{J}_{FA})^{-1} \Big(\mathbf{J}_{FA}^\top (\mathbf{y}_2 - \mathbf{y}_1^{p_c}) \Big) \tag{5.11}$$



(a) (b) (c)

Figure 5.2 – Applying Gauss-Newton descent algorithm for retrieving the translation of a 1D signal. (a) a mono-dimensional signal $y_1(x)$ (red) and its noisy sampled translation $y_2(x)$ (green). (b) Gauss-Newton optimization with 100 iterations. Intermediate estimates of the translated signal are shown in blue, with darker nuances as the number of iterations increases. The optimization converges to the true value. (c) If the initial guess p_0 is too far from the true value of p, the optimization converges to a local optimum.

where the components of the column vector $\mathbf{y}_1^{p_c}$ are given by $y_1(x + p_c)$ and $\mathbf{J}_{FA}$ is the column vector containing the derivative of $\mathbf{y}_1^{p_c}$, obtained with finite differences.

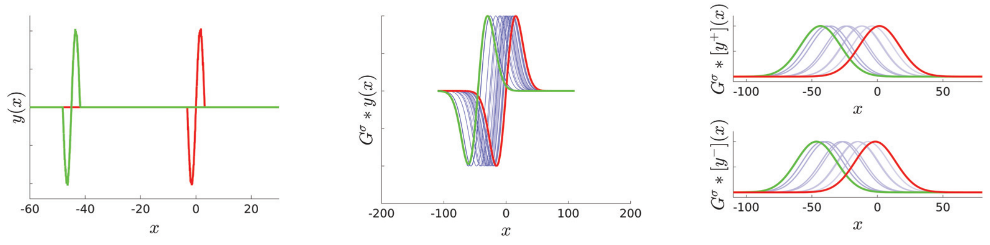
The optimization will converge to the sought optimum p if $p - p_0$ is small enough, otherwise it will converge to a local optimum, or simply diverge, as shown respectively in Figure 5.2, (b) and (c).

For retrieving large displacements, a common strategy is to smooth the signals, as described for images in Section 5.1.2 and as shown in Figure 5.3 (a),(b). Smoothing is beneficial for enlarging the basin of attraction of the algorithm, but its averaging effect also corrrupts useful information. Applying the non-linear, sparsifying function (5.9) to the signals before smoothing prevents information destruction, leading to much faster convergence, as shown in Figure 5.3 (c). Moreover, it is possible to apply larger smoothing and thus further enlarge the basin of convergence.

The 1D problem described in this section is a simplified example, but very analogous phenomena are observed when dealing with real images, as explained in the next section. In Section 5.3 we evaluate different descriptors obtained by several combinations of local jets and the non-linear function of Equation (5.9), together with different optimization algorithms, on several challenging video sequences.

5.2.2 An Example with Real Images

In Figure 5.4 we show the values of different d functions computed for images of a specular 3D scene, and the values of the corresponding objective functions for a translation warp (see Eqs. (5.1) and (5.3)). The values of different descriptors d have been sampled on 200 equi-spaced points along the reprojections of a 3D line lying on the background in the 2 images shown in Figure 5.4 (a), (b). In the third column we show the resulting objective function for a 1D translation warp. The expected location for the global minimum is marked with a red dot.



(a) (b) (c)

Figure 5.3 – Applying Gauss-Newton descent algorithm for retrieving large displacements between 2 noisy sampled signals $y_1(x)$ and $y_2(x)$. The true displacement is $p = 45$, while $p_0 = 0$. (a) The retrieval is impossible for the raw signals, the optimization gets stuck from the first iteration. (b) The optimization applied to signals smoothed with a 1D gaussian filter of standard deviation $\sigma = 20$ converges in about 50 iterations: the smoothing enlarged the basin of attraction of the algorithm. (c) The optimization applied to our Descriptor Fields smoothed with the same 1D gaussian filter converges in less than 10 iterations: not only the optimization is faster, but also the basin of attraction is further enlarged by applying the non-linear function (5.9)

.

Employing the raw intensity signals (c), (d), (e), the objective function exhibits local minima at wrong locations. Low-pass filtering the intensities (f), (g), has the effect of erasing the local minima from the objective function (h), but there is no minimum at the expected location, either. The descriptor obtained convolving the intensity signals with the first derivative of a Gaussian kernel (i), (j) is much more resilient to the local illumination artifacts, so that objective function computed with local jets (k) has a minimum at the expected place; on the other hand, it also exhibits many local minima. Low-pass filtering local jets (l), (m) makes the corresponding objective function (n) smoother, but the global minimum is displaced at the wrong location, and a local minimum appears. Finally, applying the non-linear $[.]^+$ operation to the local jets as shown in (o), (p), and smoothing them (r), (s) better preserves information, leading to the objective function (t), which is much better suited for numerical optimization.

5.3 Experimental Results

In this section, we first describe the datasets we used to evaluate our approach and the evaluation framework, and then present and discuss the results of the evaluation.

5.3.1 Datasets

We introduce a new dataset for the evaluation of 3D tracking algorithms on the challenging environments we consider; the dataset shows two different environments:

- **Experimental Setup Dataset**: Figure 5.1 illustrates the first environment. It is made from an experimental setup, where the background is covered with aluminum foil, and the foreground is made of non-textured boxes. A lamp is moved freely in the scene. Since the aluminum foil is very reflective, the images contain many specularities that move with the lamp. Also, the lamp occasionally occludes the scene. We used only one template and the 3D model made of 168 triangles shown in Figure 5.1 (a)-(b). We captured two video sequences. The first one is made of 394 frames, the camera remains still, and the lamp is moved around. The second video is made of 365 frames, and both the camera and the light source move.
- **ATLAS Dataset**: Figure 5.5 shows images from this second dataset. This dataset was captured in the LHC particle detector of ATLAS experiment at CERN. We captured a first video made of 209 frames with a fixed camera and a strong moving light source. The second video is longer, with 683 frames, and is much more challenging. The camera moves sometimes very fast, which results in motion blur. The light source generates very bright specularities in an extremely dark environment. This mimics the conditions of images captured by a camera mounted on the helmet of a worker in the ATLAS particle detector at CERN. We used the very simple 3D model showed in Figure 5.5 (a) made of only 12 triangles, and 24 templates.

Moreover, in order to give an example of how our approach behaves in a Lambertian environment, we report the results of the tests performed on a video sequence of 414 frames showing the popular STOP sign of the METAIO Dataset [124] printed on a sheet of paper, with limited motion blur and stable illumination conditions. For this sequence we employed the same workflow described above, retrieving the full 3D pose with a 3D model (made of 2 triangles) and 11 templates.

We tested PTAM, the state-of-the-art SLAM method of [85], on the two specular scenes. After several attempts, we managed to initialize the 3D tracking under ambient light; however, tracking was lost as soon as a lamp was switched on. [2] This attests the difficulty of our datasets, and shows that a feature point-based approach, such as PTAM, is not adapted.

To obtain the ground truth camera poses for our datasets, we had to register the images by manually matching 3D points on the scene models with their 2D reprojections in the images, and use these correspondences to estimate the camera poses with a PnP algorithm [125]. Our testing datasets, as well as some supplementary material, are available on the project page at http://cvlab.epfl.ch/research.

5.3.2 Evaluation Framework

We evaluated different possibilities for the d function in Equation (5.1) including the ones discussed in Section 5.2. In the following, we will refer to them as:

- Intensity: the simple case when $d(I, \mathbf{x}) = I(\mathbf{x})$;

---

[2] The reported results are also shown on the spotlight video of [9], available at https://www.youtube.com/watch?v=yw5hoImVuf8

- Magnitude of image gradient: in this case, d is taken as $d(I, \mathbf{x}) = \sqrt{(G_x * I)(\mathbf{x})^2 + (G_y * I)(\mathbf{x})^2}$, the magnitude of the image gradient at location $\mathbf{x}$. Like our descriptors, it is a non-linear function of the image intensities;

- 1^{st}-order local jet: the simple local jet $d(I, \mathbf{x}) = [(G_x * I)(\mathbf{x}), (G_y * I)(\mathbf{x})]^\top$ as given in Equation (5.8), where G_x and G_y are the first derivatives of the Gaussian kernel of standard deviation 1.0;

- 1^{st}- and 2^{nd}-order local jet: the simple local jet as given in Equation (5.8), with $\mathbf{f}_1 = G_x$, $\mathbf{f}_2 = G_y$, $\mathbf{f}_3 = G_{xx}$, $\mathbf{f}_4 = G_{xy}$, $\mathbf{f}_5 = G_{yy}$, the first and second derivatives of the Gaussian kernel of standard deviation 1.0.

- 1^{st}-order Descriptor Fields: in this case, d returns our descriptor as given in Equation (5.9), with $\mathbf{f}_1 = G_x$ and $\mathbf{f}_2 = G_y$, the first derivatives of the Gaussian kernel of standard deviation 1.0;

- 1^{st}- and 2^{nd}-order Descriptor Fields: in this case, d returns our descriptor as given in Equation (5.9), with $\mathbf{f}_1 = G_x$, $\mathbf{f}_2 = G_y$, $\mathbf{f}_3 = G_{xx}$, $\mathbf{f}_4 = G_{xy}$, $\mathbf{f}_5 = G_{yy}$, the first and second derivatives of the Gaussian kernel of standard deviation 1.0.

In all our experiments, the Distribution Fields method [115], as summarized in Equation (5.6), performed badly whatever the values for n: it successfully registered no more than 10% of the frames. This is due to the fact that local specularities heavily alter the distribution of pixels in the bins of intensity histograms, so that Distribution Fields are totally unsuitable in presence of strong local light changes, even if image intensities are normalized before computing the descriptors.

Before computing all the mentioned descriptors we first normalized the image intensities by subtracting their mean and dividing them by their standard deviation, as it improved the performances of all the methods.

Each of these descriptors was tested together with the Forward Additive (FA) algorithm described in Section 4.3.1, the Inverse Compositional Algorithm (IC) described in Section 4.3.3, and the Efficient Second Order Method (ESM) presented in Section 4.4. We optimized the parameters of each method to obtain the best performances.

5.3.3 Evaluation

Table 5.1 summarizes the results of our experiments. We report the percentage of frames that were correctly registered, together with the average number of iterations required for convergence. To decide if a frame was correctly registered or not, we compute a rotation error R_{err} and a translation error t_{err}. The rotation error is taken as the distance between the exponential maps for the estimated pose and for the ground truth, and the translation error as the distance between the camera centers for these two poses. If at least one of these errors is larger than a threshold, then we consider that the frame is not correctly registered. We use $\epsilon_{\text{rot}} = 0.07$ for the threshold on the rotation error, and $\epsilon_{\text{transl}} = 0.05$ for the threshold on the translation error. As shown in Figure 5.6, the values of these

Descriptor	Optimization Method	Lambertian Env. Video	Exp. Setup Video #1	Exp. Setup Video #2	ATLAS Video #1	ATLAS Video #2
Intensity	FA	88.7 (16.2)	25.6 (48.7)	10.7 (76.5)	40.7 (70.3)	21.7 (44.6)
	IC	16.0 (17.1)	42.1 (72.9)	22.2 (49.2)	88.6 (117.7)	19.3 (32.6)
	ESM	72.7 (43.9)	34.7 (46.8)	21.9 (46.2)	36.8 (62.1)	22.5 (40.8)
Magnitude of image gradient	FA	84.2 (22.)	52.0 (55.6)	81.0 (55.1)	99.5 (45.3)	33.6 (44.1)
	IC	18.4 (16.7)	83.9 (71.9)	73.9 (45.4)	96.6 (27.2)	29.5 (31.7)
	ESM	67.8 (31.3)	90.8 (30.5)	92.0 (43.1)	89.9 (33.3)	19.7 (28.4)
1st-order local jet	FA	85.2 (29.1)	75.6 (39.0)	52.3 (37.5)	100 (73.6)	31.5 (32.6)
	IC	28.3 (23.5)	73.0 (33.5)	50.1 (41.7)	100 (50.5)	23.4 (34.2)
	ESM	78.3 (35.0)	75.6 (27.8)	49.5 (25.8)	100 (67.7)	24.7 (35.8)
1st- and 2nd-order local jet	FA	91.2 (27.4)	67.8 (49.0)	46.8 (45.4)	100 (36.1)	31.5 (31.3)
	IC	57.7 (20.5)	71.0 (40.7)	46.3 (63.6)	98.5 (37.7)	22.9 (27.7)
	ESM	84.9 (26.0)	74.4 (34.2)	50.7 (33.6)	100 (27.9)	24.0 (21.3)
1st-order Descriptor Fields	FA	89.3 (25.4)	**85.0 (49.0)**	87.9 (86.5)	100 (47.6)	**39.4 (33.8)**
	IC	37.0 (22.4)	**91.4 (51.7)**	82.2 (66.3)	100 (29.9)	32.5 (27.4)
	ESM	77.0 (38.6)	**98.4 (30.4)**	97.5 (36.9)	100 (51.3)	32.6 (21.3)
1st- and 2nd-order Descriptor Fields	FA	**93.3 (35.9)**	76.1 (63.3)	**89.3 (69.9)**	**100 (24.4)**	39.0 (30.7)
	IC	**61.9 (23.6)**	82.7 (47.2)	**85.2 (62.3)**	**100 (22.7)**	**32.5 (24.7)**
	ESM	**87.5 (33.9)**	92.8 (42.4)	**97.8 (39.4)**	**100 (19.0)**	**33.4 (18.5)**

Table 5.1 – Experimental results. We give the percentages of correctly calibrated frames and the average number of iterations in parentheses for each descriptor, each video sequence, and each optimization method we considered. The best results for each video and each optimization methods are in bold. Our Descriptor Fields consistently outperform the other descriptors.

thresholds are not critical: when a frame is not correctly registered, the rotation and translation errors tend to be very large.

As can be seen in the table, the results with our Descriptors Fields are consistently better than the other approaches, for all the videos and the optimization methods.

In all the specular video sequences, our descriptor with first-order Gaussian derivatives outperforms all other approaches based on first-order derivatives. Using both first- and second-order derivatives can further improve performances at a higher computational cost.

Figures 5.7 and 5.8 show some images from our datasets augmented with virtual objects using the poses estimated with our first-order Descriptor Fields. The virtual objects are consistently integrated in the images, which assesses that the camera poses were correctly estimated.

An interesting question arising from the experimental results is, which alignment algorithm should be chosen. As described in Section 4.5, the theoretical advantaged of some methods are only valid when some regularity assumptions about the warps hold, but it is not the case here. Moreover, in our case the computational cost of each iteration of the Inverse Compositional algorithm is comparable to that of the other methods because inversion and composition of the warp of Figure 4.6 are computationally expensive. We observed that IC algorithm performs better in the model video sequences, where the template is made by a picture taken in a controlled environment, with good lighting conditions and no motion blur; on the other hand, FA optimization performs better on the ATLAS video sequences, where key-frames were extracted from videos and are noisy and blurred.

If it is not possible to know in advance whether the templates or the images will be more noisy, then ESM is probably the most reliable choice, even for warps that violate assumption (4.39), since it averages the image and template gradients. Otherwise, relying on the gradients of the less noisy frames (those of the template for IC or those of the incoming frames for FA) would be a suitable choice.

5.3.4 Evaluation of the Distance Function

As shown in Table 5.1, our Descriptor Fields consistently improve the performances of dense descriptors across several optimization methods. We performed a further experiment in order to verify that their efficacy does not rely on the choice of a particular distance function [3] in the optimization functions (5.1) and (5.3). For the 2 images shown in the first line of Figure 5.4, we computed the values of the objective functions:

$$F(\mathbf{p}) = \rho\Big(\mathbf{I}_{\mathbf{p}}^{d}, \mathbf{T}^{d}\Big), \qquad F(\mathbf{p};\sigma) = \rho\Big(\mathbf{I}_{\mathbf{p}}^{D_\sigma}, \mathbf{T}^{D_\sigma}\Big), \tag{5.12}$$

where ρ is a distance function and, given a set of pixel locations $\mathbf{x}$, $\mathbf{T}^d$ and $\mathbf{I}_{\mathbf{p}}^d$ are the column vectors whose components are given, respectively, by $d(T, \mathbf{x})$ and $d(I, \mathbf{W}(\mathbf{x}, \mathbf{p}))$; similarly, the components of the vectors $\mathbf{T}^{D_\sigma}$ and $\mathbf{I}_{\mathbf{p}}^{D_\sigma}$ are given, by the smoothed descriptors $D_\sigma(T, \mathbf{x})$ and

---

[3] Mathematically speaking, the term “distance function”could be applied only to SDD, which is a metric, while NCC and MI are just “distance scores”. We employ this slight abuse of terminology since it does not lead to confusion.

$D_\sigma(I, \mathbf{W}(\mathbf{x}, \mathbf{p}))$; we evaluated the following distance metrics:

- **SSD:** The Sum of Squared Differences is the distance metric employed for all the tests reported in Table 5.1; it is widely employed thanks to its simplicity and efficiency of evaluation. For 2 generic column vectors $\mathbf{L} = [L_1, \ldots, L_N]$, and $\mathbf{M} = [M_1, \ldots, M_N]$, a it is defined as:

$$\text{SSD} \;:\; \rho(\mathbf{L}, \mathbf{M}) = \Big(\mathbf{L} - \mathbf{M}\Big)^\top \Big(\mathbf{L} - \mathbf{M}\Big). \tag{5.13}$$

- **NCC:** The Normalized Cross Correlation between two column vectors $\mathbf{L},\ \mathbf{M} \in \mathbb{R}^N$ is computed as:

$$\text{NCC} \;:\; \rho(\mathbf{L}, \mathbf{M}) = \frac{(\mathbf{L} - \bar{\mathbf{L}})^\top (\mathbf{M} - \bar{\mathbf{M}})}{\sqrt{(\mathbf{L} - \bar{\mathbf{L}})^\top (\mathbf{L} - \bar{\mathbf{L}})}\sqrt{(\mathbf{M} - \bar{\mathbf{M}})^\top (\mathbf{M} - \bar{\mathbf{M}})}}, \tag{5.14}$$

  where $\bar{\mathbf{L}}$ and $\bar{\mathbf{M}}$ are column vectors whose components are all equal to, respectively, $\bar{L} = 1/N \sum_i L_i$ and $\bar{M} = 1/N \sum_i M_i$. NCC is always comprised between -1 and 1; with respect to SSD, it is invariant to affine changes in intensity values, but it is slower to compute.

- **MI**: For computing the Mutual Information score, the components of the vectors $\mathbf{L}$ and $\mathbf{M}$ are interpreted as sets of samples of two discrete random variables L and M, with probability distributions p_L and p_M . Then, their Mutual Information [126] is defined as:

$$\text{MI}(L, M) = \sum_{l,m} p_{LM}(l, m) \log \left(\frac{p_{LM}(l, m)}{p_L(l) p_M(m)} \right), \tag{5.15}$$

  where the sum is extended over all the possible values l, m taken by, respectively, L and M, and $p_{LM}(l, m) = p(L = l \cap M = m)$ is the joint probability distribution of L and M. Practically, the probability distributions are replaced by normalized frequency histograms of the values of $\mathbf{L}$ and $\mathbf{M}$: after fixing 2 sets of equally spaced *bin centers* $l_1, \ldots, l_{N_L}$ and $m_1, \ldots, m_{N_M}$, with $N_L, N_M << N$ (we employed $N_L = N_M = 9$ in all our experiments), the probability distributions are estimated as:

$$p_L(l_i) = \frac{1}{N} \sum_{k=1}^{N} \phi_{\delta l}(l_i - L_k)$$

$$p_M(m_j) = \frac{1}{N} \sum_{k=1}^{N} \phi_{\delta m}(m_j - M_k)$$

$$p_{LM}(l_i, m_j) = \frac{1}{N} \sum_{k=1}^{N} \phi_{\delta l}(l_i - L_k) \phi_{\delta m}(m_j - M_k),$$

  where ϕ_δ is the indicator function:

$$\phi_\delta(x) = \begin{cases} 1 & \text{if} \quad |x| < \delta \\ o & \text{otherwise,} \end{cases}$$

and $\delta_l = (l_2 - l_1)/2$, $\delta_m = (m_2 - m_1)/2$. When employing MI in iterative alignment, other functions ϕ may be employed, such as Gaussians or B-Splines [127, 105], that implement histogram soft assignment and make MI differentiable. MI is considered as very resilient to illumination changes and is also the choice of reference for *multi-modal* alignment; unfortunately, building the histograms is computationally very expensive, so that existing approaches for MI-based 3D tracking [106, 128] are not suited for real-time applications.

We computed the value of the objective functions of Equation (5.12) for different values of $\mathbf{p}$ around the ground-truth value, varying 2 of the translation parameters in the range $[-10\text{ cm}, 10\text{ cm}]$, for 3 different descriptors, respectively image intensity, first-order local jets and first order Descriptors Fields. For the smoothed descriptors, we employed a Gaussian filter with $\sigma = 20$ for all the metrics and all the descriptors; results are shown in Figure 5.10.

Some interesting observations arise from the graphs. First, MI appears to be much more discriminative than the other metrics when close to the global optimum, while, unsurprisingly, NCC and SSD behave in a qualitatively similar way. Unfortunately, MI is also much more expensive to compute, and it is not suitable for real-time 3D tracking. Real-time is achieved in [105] for planar targets tracking, coupled with an efficient inverse compositional optimization scheme that allows the image histograms to be evaluated only once at the beginning of the optimization, while in [128] a running time of about 4s per frame is reported for an application similar to ours (model-based 3D tracking). The effects of the Descriptor Fields are analogous for all the considered distance functions: given a distance function,

- the global optimum is much better defined for Descriptor Fields than for other descriptors;
- smoothing the descriptors does effectively enlarge the basin of attraction of the sought optimum, without degrading its original position.

5.3.5 Rotation Invariance

While extremely resilient with respect to illumination and translation changes, the Descriptor Fields are not invariant with respect to in-plane rotation changes, as opposed, for example, to the image intensity or to the magnitude of the image gradients. Nonetheless, depending on the application, several solutions can be adopted in order to enhance the descriptor invariance.

For instance, in [119] we implemented a demo for real-time tracking of planar surfaces based on dense image alignment and Descriptor Fields, as described in Section 5.4. There, a sequence of video frames $I_0,\ I_1, \dots, I_t, \dots$ are tracked against the first frame of the sequence, which is employed as template. After estimating the parameters $\mathbf{p}_t$ for frame I_t, we warp the template with $\mathbf{p}_t$, then we align the next frame I_{t+1} to the warped template T_t, estimating an incremental warp $\mathbf{p}_{t \to t+1}$ and we finally obtain the current parameters estimate through the compositional update $\mathbf{W}(\cdot, \mathbf{p}_{t+1}) = \mathbf{W}(\mathbf{W}(\cdot, \mathbf{p}_t), \mathbf{p}_{t \to t+1})$. Notice that this tracking scheme guarantees that the iterative alignment is performed between images with similar warps; at the same time, the tracking is drift-free, since inaccuracies in the estimation of $\mathbf{p}_t$ are compensated in the estimation of $\mathbf{p}_{t\rightarrow t+1}$.

In [110], we adopted a similar approach for tracking 3D deformable surfaces in the form of triangular meshes: there, in order to compare pixel descriptors in the same, unrotated coordinate system, the warp estimated for each frame is used to establish a local coordinate system for each mesh facet; then, the template partial derivatives within each projected facet are rotated according to the local coordinate system before computing the Descriptor Fields and aligning the next incoming frame.

5.4 Applications and Further Developments

Since their original description in [9], we employed Descriptor Fields for several 3D tracking applications, further validating their employability for several related purposes.

In [119], we implemented a demo for real-time 3D tracking of planar surfaces. A screenshot is shown in Figure 5.9-(a). Based on the user's keystroke, a template T is captured in a central rectangular region of the screen and tracked along the successive frames acquired by a webcam; the user can switch among several dense descriptors such as intensities, gradient magnitudes and Descriptor Fields for a quick qualitative evaluation of the performances of each descriptor for different kinds of surfaces. The code is publicly available[4].

Figure 5.9-(b) shows an application for face tracking: a rigid 3D model of a human face is deformed and superposed to the user's face on monocular images captured by a webcam in real time; the initial pose is computed by detecting facial landmarks, and refined using IC algorithm and Descriptor Fields [129].

We implemented an application to 3D tracking of non-rigid surfaces in [110]. A dense alignment pipeline is employed for tracking poorly textured deformable surfaces; a surface is modeled as a 3D triangular mesh and the mesh deformation is reconstructed based on dense iterative alignment of monocular images. For this application, the optimization is sensibly more complex than for rigid object tracking: the problem unknowns are no longer given by the 6 degrees of freedom of the 3D pose of a rigid object, as for the rigid object tracking, but by the 3D coordinates of some tens (or hundreds) of mesh vertices. Moreover, monocular 3D surface reconstruction is an under-constrained problem, since different 3D shapes may have the same reprojection on the image plane: additional constraints must be added to the minimization problem of Equation (5.1), such as isometric deformation constraints, enforcing that the surface should not stretch or shrink, and a regularization term that penalizes non-rigid deformations too fare from the template shape. In this way, the optimization is well posed and any of the methods described in Chapter 4 can be employed.

Another improvement proposed in [110] for enhancing robustness to occlusions and poorly textured regions is to compute a so-called *relevancy score* $\tilde{\omega}(\mathbf{x})$ for each pixel $\mathbf{x}$ and employ the score as a

---

[4] https://github.com/albertoCrive/homographyTrackingDemo

weight for each term in the sum of Equation (5.1):

$$\mathbf{p}_I = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \tilde{\omega}(\mathbf{x}) \Big(I(\mathbf{W}(\mathbf{x}, \mathbf{p})) - T(\mathbf{x}) \Big)^2 . \tag{5.16}$$

The relevancy score for pixel $\mathbf{x}$ is computed as:

$$\omega(\mathbf{x}) = \max_{\delta} \mathrm{NCC}\Big(Q_T(\mathbf{x}), Q_I(\mathbf{W}(\mathbf{x}, \tilde{\mathbf{p}}) + \delta) \Big), \tag{5.17}$$

where $\tilde{\mathbf{p}}$ is the best parameters estimate available, for instance the parameters estimated for the previous tracked frame in a video sequence, NCC is the normalized cross-correlation introduced in Equation (5.14), $Q_T(\cdot),\ Q_I(\cdot)$ are small patches extracted around a pixel on image T and I, and $\delta = [\delta_x, \delta_y]^T$ is a pixel offset varying in a suited range (in [110] patches of size 26×26 are extracted, and δ_x, δ_y vary over a range of $[-30, 30]$ pixels). Normalized weights $\tilde{\omega}(\mathbf{x})$ are obtained, by rescaling the scores to lie in $[0, 1]$ and clamping the values of the weights too far from the average score for limiting the influence of outliers. This approach is shown to be effective for reducing at once the influence of occluded and un-textured regions; on the other hand, it sensibly increases the computational complexity of the optimization, so that it is not suited for real-time applications.

5.5 Conclusion

In this chapter, we presented a local descriptor that makes dense alignment methods such as the ones presented in Chapter 4 much more robust to various imaging artefacts. Descriptor Fields are efficient and very simple to implement, so that it is very easy to integrate them into existing image alignment algorithms, to improve their robustness. On the other hand, while our global approach enhances robustness with respect to local image artifacts, its application may be problematic when tracking small objects undergoing extreme occlusions. The approach presented in Chapter 6 aims at resolving these issues employing a part-based approach. The two approaches are meant to be complementary, the choice of the most suited method depending on the target application.

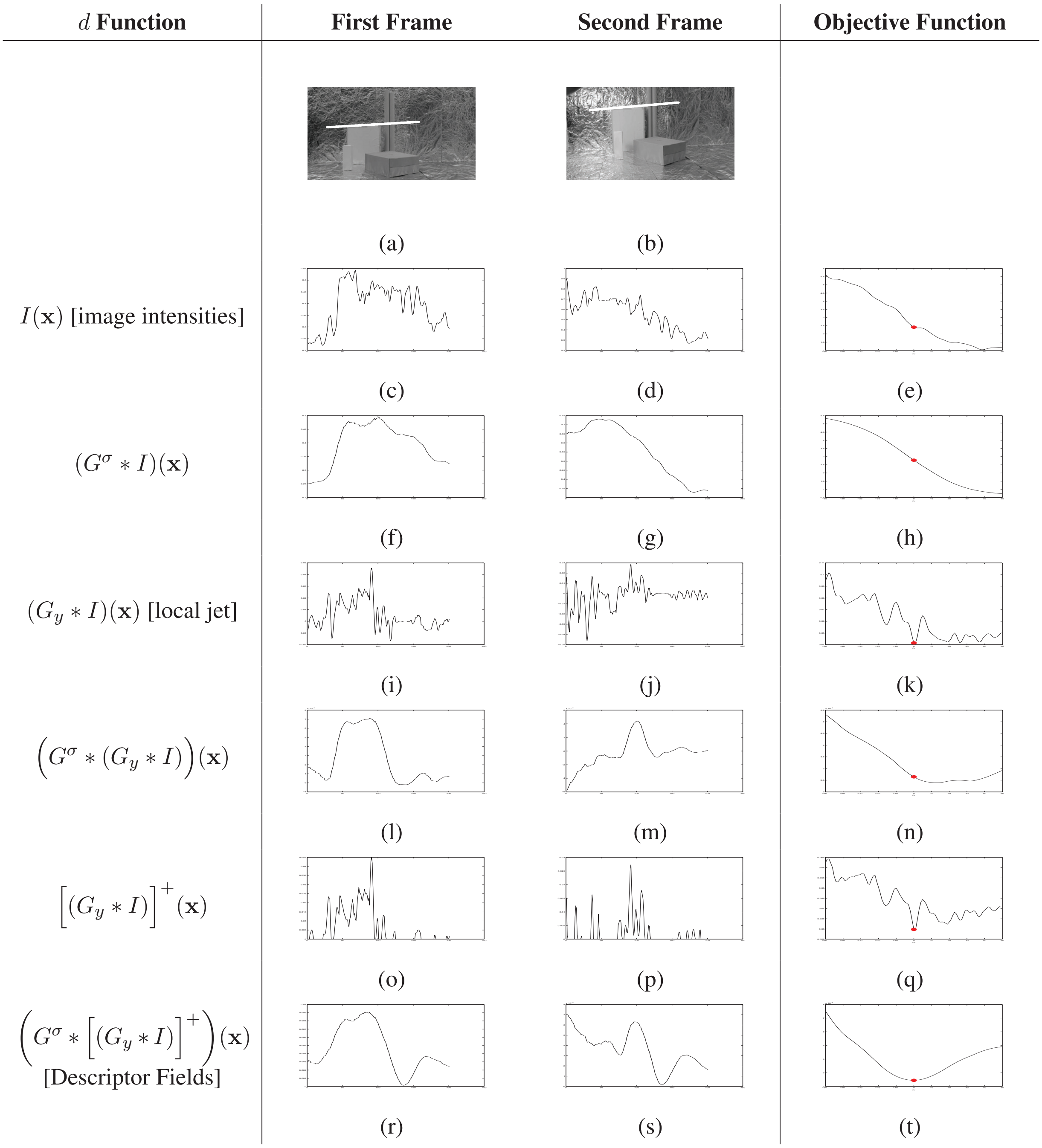
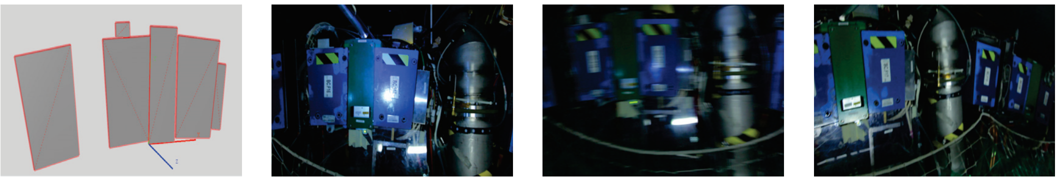
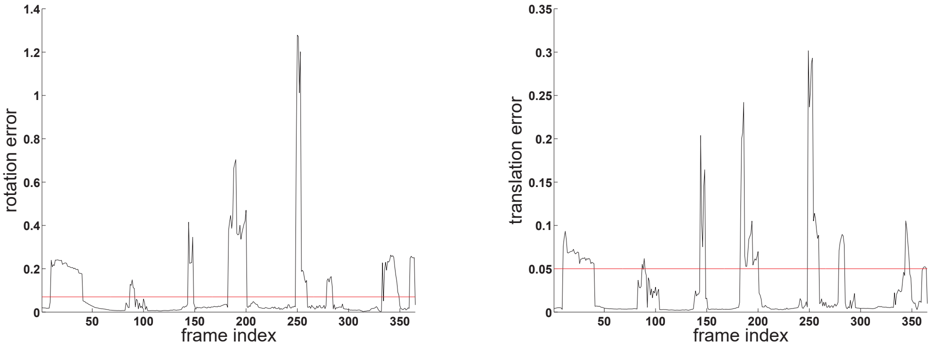


Figure 5.4 – Different d functions on a specular surface, and corresponding objective functions for a translation of a 3D scene (see Eqs. (5.1) and (5.3)), as explained in Section 5.2.2. Second and third column: value of different descriptors d sampled on 200 equi-spaced points along the reprojections of a 3D line lying on the background in the 2 images (a) and (b). Last column: objective function for these signals, with the expected location for the global minimum is marked with a red dot.



(a) (b) (c) (d)

Figure 5.5 – (a) The 3D model we use for the ATLAS dataset. (b,c,d) Some images from the second video sequence of this dataset. The images exhibit large and bright specular spots and strong motion blur.



(a) (b)

Figure 5.6 – (a) Rotation and (b) translation errors over the second video sequence of the Experimental Setup dataset, using ESM and our 1^{st}-order Descriptor Fields. The horizontal lines correspond to the thresholds used to detect incorrectly registered frames.

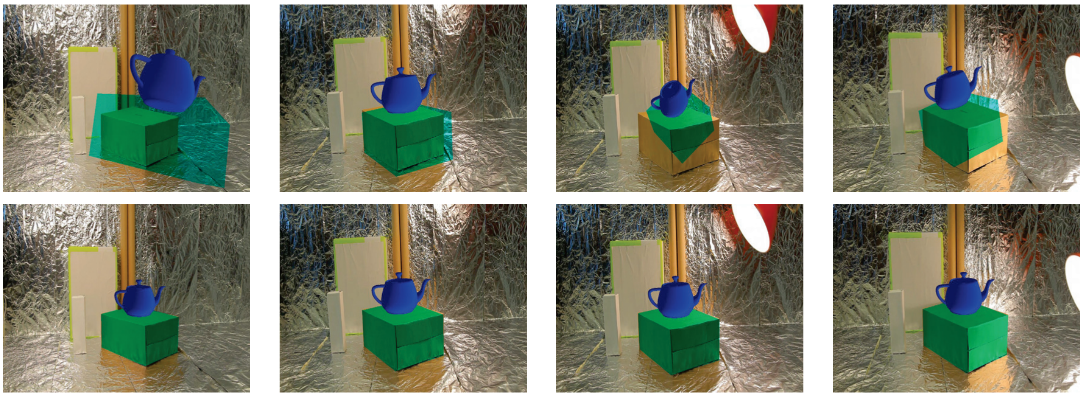


Figure 5.7 – Comparisons on our Experimental Setup dataset. First row: Using the image intensities. Second row: Using our Descriptor Fields. The scene is augmented with the obligatory teapot to visually attest the accuracy of the estimated poses. With our method, the teapot is correctly added to the images, despite the moving lamp that changes the lighting and partially occludes the scene. The full video is available on the project page at http://cvlab.epfl.ch/research.

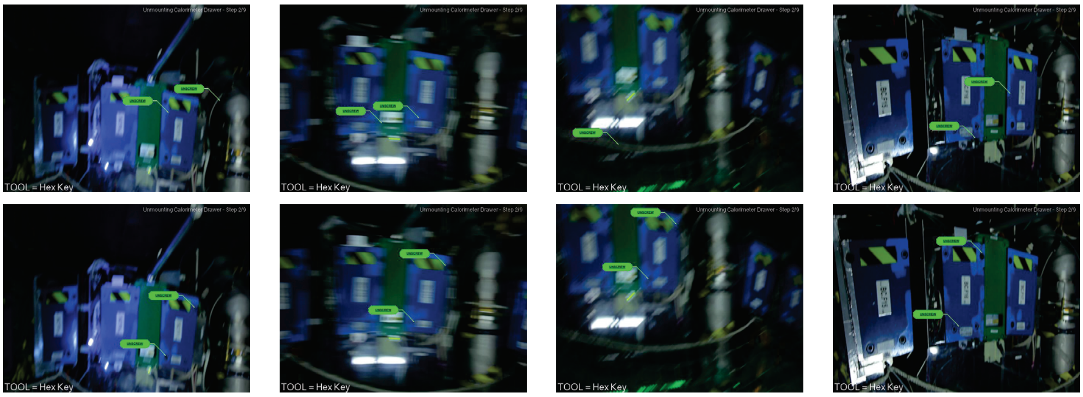
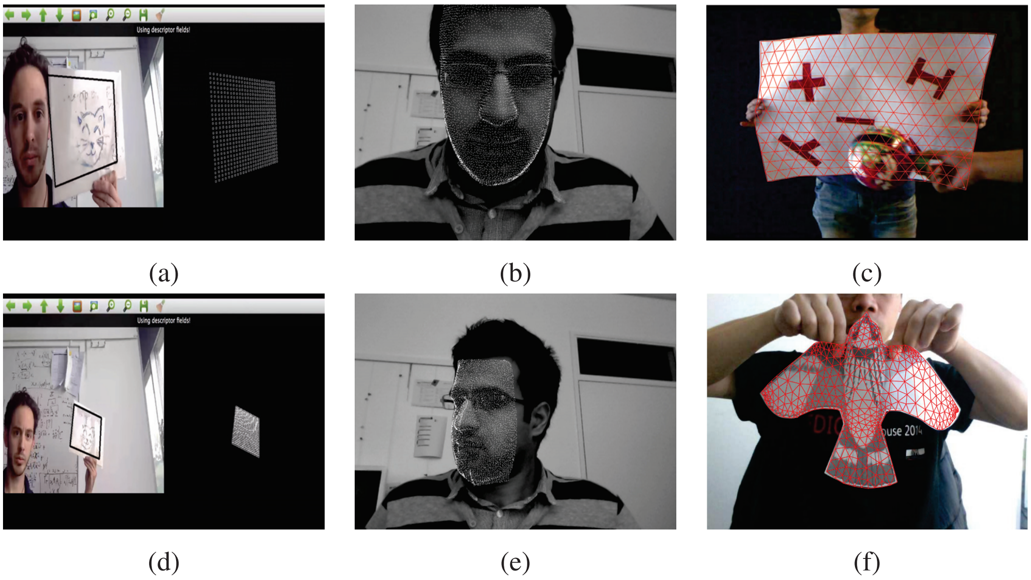


Figure 5.8 – Comparisons on our ATLAS dataset. First row: Using the image intensities. Second row: Using our Descriptor Fields. Despite the bright specularities and the motion blur, we can add virtual labels at the right place in the images with our method. The full video is available on the project page at http://cvlab.epfl.ch/research.


(a) (b) (c)

(d) (e) (f)

Figure 5.9 – Some applications of image alignment based on our Descriptor Fields, described in Section 5.4. (a), (d) Real-time 3D tracking of planar surfaces; (b), (e) Real-time 3D face tracking (images courtesy of Mahdi Rad). (c), (f) tracking of non-rigid surfaces.

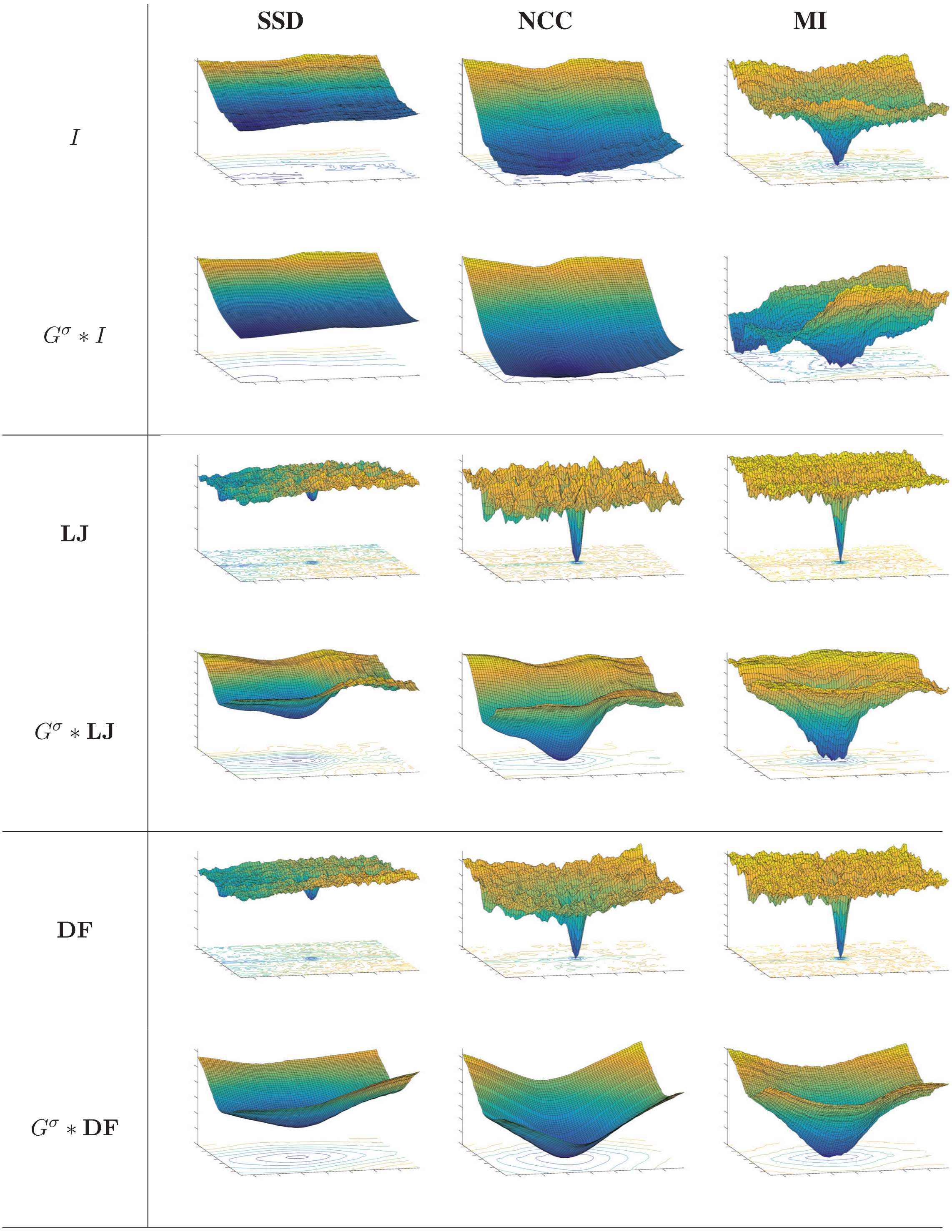


Figure 5.10 – **Comparison between different distance functions for image alignment.** Shape of the objective functions of Equation (5.12) for the 2 images shown in the first line of Figure 5.4 for different values of $\mathbf{p}$ around the ground-truth value, varying 2 of the translation parameters in the range $[-10\text{cm}, 10\text{cm}]$. The groundtruth parameters value lies in the middle of the xy plane, where the optimization should end. Each column represents a different distance function, respectively the SSD, the NCC and the MI. For easier comparisons, we flipped the z axis for NCC and MI. **LJ** are the 1^{st}-order local jet descriptor, **DF** are the first-order Descriptor Fields.

6

Robust 3D Tracking using Stable Parts

In Chapter 5 we proposed a 3D tracking approach based on a dense image alignment framework and a robust dense descriptor for enhancing robustness in presence of local artifacts issued from illumination, occlusions, or locally ambiguous patterns. While relying on a dense image alignment framework allows to effectively handle local artifacts and nuisances, it can fail when dealing with objects undergoing extreme occlusions in cluttered environments. Moreover, some information about the scene geometry must be known, such as the CAD model shown in Figure 5.1 (a). This can become problematic for several common test cases, such as the electric box depicted in Figure 6.1, tracked for an Augmented Reality application: the target object is only partially visible, surrounded by an unknown, cluttered, environment with moving distractor objects; moreover, it undergoes heavy occlusions, at such an extent that a wide part of the object itself, the inner part of the box, acts as an occlusion: the disposition of the objects contained in an electric box is generally unknown before run time; moreover, its content is likely to change at run time, for instance when the user removes and replaces some parts during a technical intervention. Even obtaining geometric information about most part of the target object would be challenging in these conditions.

We tested the dense tracking framework described in Chapter 5 in this scenario, but failures often arise, due to the extremely limited amount of image information that can be exploited for tracking. As for the other state-of-the art methods described in Section 2, each of them has its own weaknesses: Many of these approaches [42, 44, 54, 130] rely on a depth sensor, which would fail on metallic objects or outdoor scenes; methods based on feature points [29, 131] expect textured objects; those based on edges [132, 133] are sensitive to cluttered background; most of these methods are not robust to huge occlusions.

In this chapter we describe a 3D tracking approach, that undertakes the challenge of tracking higly occluded objects relying on the efficient detection of discriminative parts of the target object.

Relying on parts for 3D object detection has already been proposed in previous works [46, 72, 73, 76, 74]; the main novelty of our approach resides in a powerful representation of the pose of

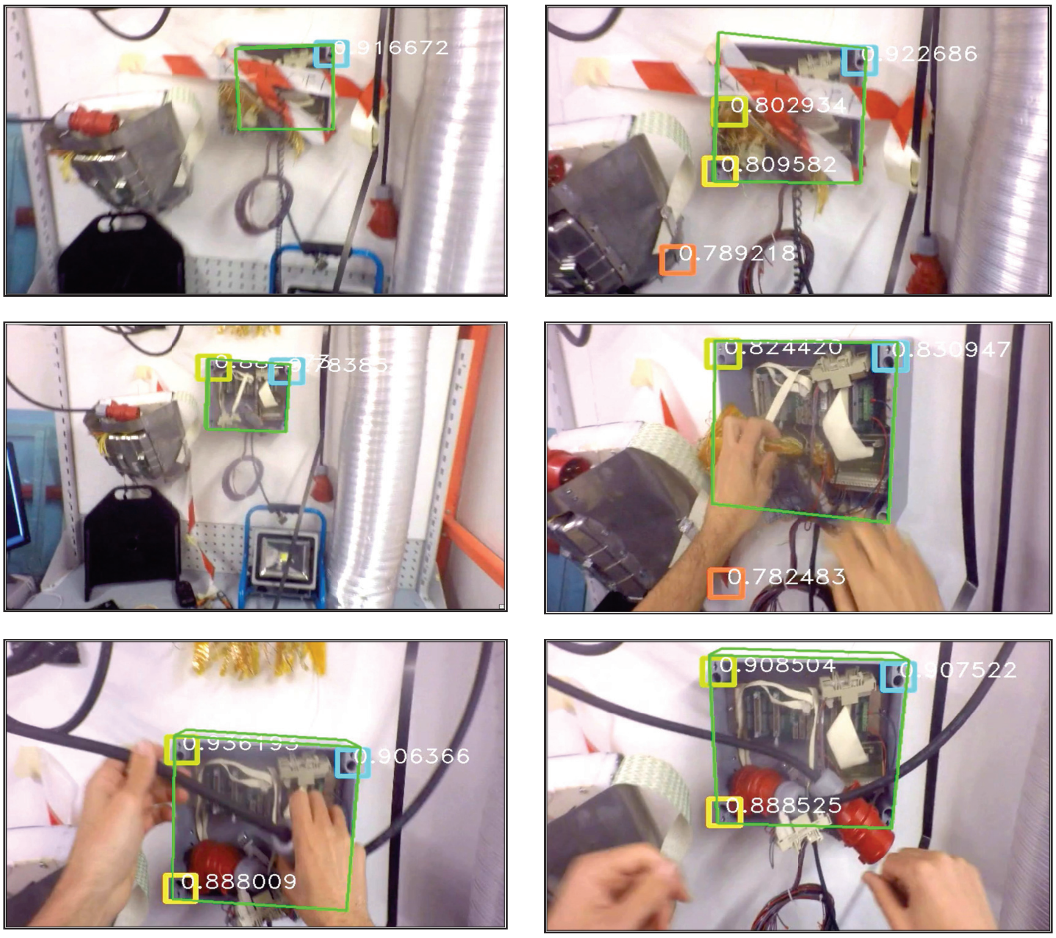
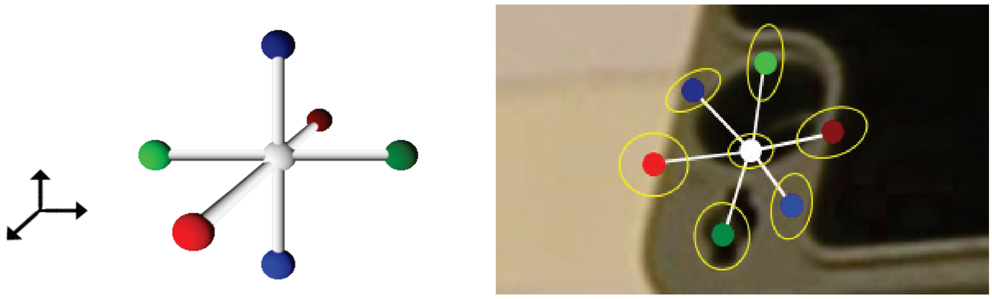


Figure 6.1 – Our part-based method in action during a demonstrative technical intervention at CERN, Geneva. Detected parts are shown as colored rectangles. The appearance of the scene constantly changes and undergoes heavy occlusions. Despite these difficulties, we accurately estimate the 3D pose of the box, even if only one part is detected or in presence of false detections caused by the cluttered environment.

each part. Some previous methods assume homographies [77, 46, 74] to represent a part pose, however this can only applied to piece-wise planar objects, and it is not easy to combine the homographies from several parts together to compute a better pose for the target object. Moreover, feature point-based methods simply use the 2D locations of the feature points, which wastes very useful information.

Our pose representation is represented by the *2D reprojections of a small set of 3D control points*, shown in Fig. 6.2. The control points are only “virtual”, in the sense they do not have to correspond to specific image features nor to 3D prominent points on the object. Among others, this representation is invariant to the image location of the part and only depends on its appearance. We employ a Convolutional Neural Network (CNN) [134] to accurately predict the locations of these reprojections, as well as the uncertainty of the location estimates.



(a) (b)

Figure 6.2 – Our representation of the 3D pose of an object part. (a) We consider seven 3D control points for each part, arranged to span 3 orthogonal directions. (b) Given an image patch of the part, we predict the 2D reprojections of these control points using a regressor, and the uncertainty of the predictions.

In Section 6.3, we analyze in detail the theoretical underpinnings of why this representation is more effective than alternative approaches; our experimental results confirm our analysis showing a substantial performance gain when employing our part representation.

Our tracking pipeline consists in three main steps: Given an input image, we first run a detector to locate each part on the image. As described in Section 6.2, we also use a CNN for this task, but another detection method could be used. Then, we predict the reprojections of the control points by applying a specific CNN to each detection hypothesis. This gives us a set of 3D-2D correspondences, from which we can finally compute the 3D pose of the target object with a simple robust algorithm.

This approach has several advantages:

- We do not need to assume the parts are planar, as was done in some previous work;
- we can predict the 3D pose of the object even when only one part is visible;
- when several parts are visible, we can combine them easily to compute a better pose of the object;
- the 3D pose we obtain is usually very accurate, even when only few parts (possibly a single one) are visible.

The algorithm described in this chapter has been implemented in the EDUSAFE Augmented Reality Prototype described in Section 1.2.1, providing reliable rigid pose estimation for Augmented Reality-based assistance to maintenance interventions at CERN.

The content of this chapter was previously published in [10] and in [135], and demonstrated at CVPR 2016.

6.1 Overview of the Method

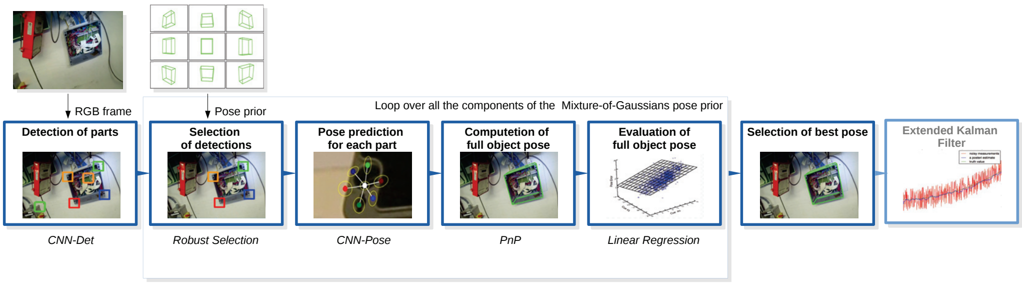

RGB frame
Pose prior
Loop over all the components of the Mixture-of-Gaussians pose prior
Detection of parts
Selection of detections
Pose prediction for each part
Computetion of full object pose
Evaluation of full object pose
Selection of best pose
Extended Kalman Filter
CNN-Det
Robust Selection
CNN-Pose
PnP
Linear Regression

Figure 6.3 – Main steps of our part-based algorithm. Given an input image, we detect reliable parts of the target object, as explained in Section 6.2. After pruning the set of detection candidates for removing erroneous detections, as described in Section 6.4.2, we predict a pose for each part (Section 6.3); finally, we combine the estimations of all the detected parts for obtaining an accurate pose of the target object (Section 6.4). Since more poses are computed starting from multiple prior pose hypotheses, we select our final pose as explained in Sections 6.4.3 and 6.4.4. Each frame can be tracked independently; the Extended Kalman filter described in Section 6.5 is employed when tracking frames along a video sequence.

Our goal is to estimate the 3D pose of a known rigid object with respect to a projective camera given an input grayscale image of the object. We assume the internal parameters of the camera are known. Additionally, we assume that we are given some geometric information about the object, such as a non-textured 3D model, and a set of manually labeled parts on it. A *part* is simply defined as a discriminative region of the object, which can easily be detected on an input image. The object model is only used for annotating the 3D location of the parts on the object and for computing the silhouette of the object under different views, as described in Section 6.4.4. This allows us to use extremely simple models, for example a parallelepiped for an electric box, or a cylinder for a food can. We can thus neglect details that would be difficult or impossible to reconstruct, such as the interior of the electric box depicted in Figure 6.1.

Ideally, the parts should be spread over the object. No assumption is made about their size: usually, bigger parts are more discriminative, but smaller parts are less likely to be partially occluded. The 3D pose of the object is retrieved exclusively from its parts, while the appearance of the rest of the object can freely vary with occlusions, clutter, etc., without affecting the final result. A very small number of parts is required by our framework—in all our tests we employed at most 4 parts for an object, and, in general, our objects of interest have very few discriminative regions, so we select the parts by hand. For training our algorithm, we make use of a set of registered training images, showing the parts under different poses and lighting conditions.

The main steps of the algorithm described in this chapter are illustrated in Figure 6.3. After detecting several candidates for each of the parts of the target object, as described in Section 6.2, we select the most likely candidates given a prior on the pose with the procedure explained in Section 6.4.2. For each selected candidate, we estimate the 3D pose of the target part (Section 6.3) and, if more than one part are visible, we combine the 3D poses computed for each part for

symbol	meaning
i	index of a training image
p	index of a part
k	index of a control point or its projection
l	index of a detection candidate on a testing image
$\mathbf{C}_p$	3D center of the p-th part
I_i	$i-$th training image
$\mathbf{c}_{ip}$	projection of $\mathbf{C}_p$ in the i-th training image
$\mathbf{V}_{pk}$	k-th 3D control point of the p-th part
$\mathbf{v}_{ipk}$	projection of $\mathbf{V}_{pk}$ in the i-th training image
$\hat{\mathbf{c}}_{pl}$	l-th detection candidate for the projection of $\mathbf{C}_p$ in an input image
s_{pl}	score for this detection
$\hat{\mathbf{v}}_{pk}$	prediction for the projection of $\mathbf{V}_{pk}$ (no outliers)
$\mathbf{S}_{pk}$	covariance for prediction for the projection of $\mathbf{V}_{pk}$ (no outliers)
$\hat{\mathbf{v}}_{pkl}$	l-th prediction for the projection of $\mathbf{V}_{pk}$ in an input image
Q	an image patch
S_q	Size of image patch Q
I	incoming image at test time
M	number of components of the Mixture-of-Gaussians pose prior
$(\overline{\mathbf{p}}_m, \mathbf{S}_m)$	average and covariance of the m-th component of the Mixture-of-Gaussians pose prior
$\hat{\mathbf{p}}^{(m)}$	pose estimated starting from the m-th component of the prior
$\hat{\mathbf{p}}$	final estimation of the pose

Table 6.1 – Main notations employed in Chapter 6.

estimating the pose of the target object (Section 6.4). Since several priors can be used at the same time, we assign a score to each of the computed poses. This score depends on several cues, and is also learned using linear regression (Sections 6.4.3 and 6.4.4). Finally, we select the pose with the best score as our final estimation. When tracking frames across a video sequence, we employ the Extended Kalman filter described in Section 6.5 in order to reduce the jitter and provide smoother trajectories.

For sake of clarity, the main notations employed in this chapter are summarized in Table 6.1.

6.2 Part Detection

The first step of our pipeline is the detection of the visible object parts on the image. Different methods could be employed for this step. Motivated by the recent success of the Convolutional Neural Networks for object detection [136, 137, 138, 139], we use a CNN for predicting the parts locations on the image, which appears to work also well for this task.

In order to learn to detect the parts, we exploit a set of registered training images as the one shown in Figure 6.4(a). We denote our training data as:

$$\mathcal{T} = \left\{ \left(I_i, \left\{ \mathbf{c}_{ip} \right\}_p, \left\{ \mathbf{v}_{ipk} \right\}_{pk} \right) \right\}_i \quad , \tag{6.1}$$

where I_i is the i-th training image, $\mathbf{c}_{ip}$ the projection of the center $\mathbf{C}_p$ of the p-th part on I_i, and $\mathbf{v}_{ipk}$ the projection of the k-th control point of the p-th part on the image.

During an offline stage, we train a CNN with a standard multi-class architecture shown in Figure 6.5 to detect the parts. The input to this CNN is a 32×32 image patch Q, its output consists of the likelihoods of the patch to correspond to one of the $N_P + 1$ parts. We train the CNN with patches randomly extracted around the centers $\mathbf{c}_{ip}$ of the parts in the training images I_i and patches extracted from the background, and by optimizing the negative log-likelihood over the parameters w of the CNN:

$$\widehat{w} = \arg\min \sum_{p=0}^{N_P} \sum_{Q \in \mathcal{T}_p} - \log \text{softmax}(\text{CNN}_w^{\text{part-det}}(Q))[p] \quad , \tag{6.2}$$

where $\mathcal{T}_p$ is a training set made of image patches centered on part p and $\mathcal{T}_0$ is a training set made of image patches from the background, $\text{CNN}_w^{\text{part-det}}(Q)$ is the $N_P + 1$-vector output by the CNN when applied to patch Q, and $\text{softmax}(\text{CNN}_w^{\text{part-det}}(Q))[p]$ is the p-th coordinate of vector $\text{softmax}(\text{CNN}_w^{\text{part-det}}(Q))$.

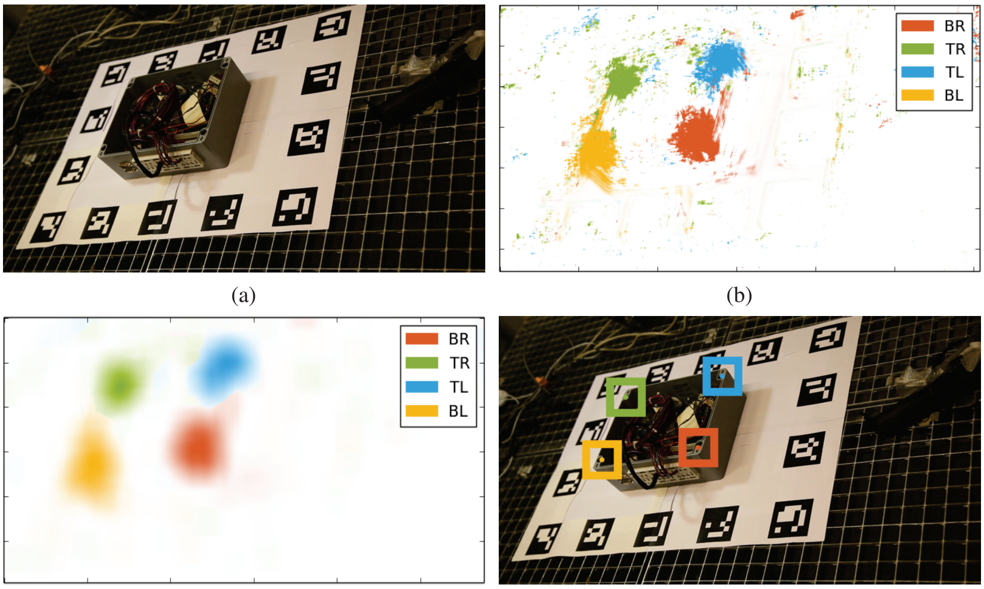
At run time, we apply this CNN to each 32×32 patch in the input images captured by the camera. This can be done very efficiently as the convolutions performed by the CNN can be shared between the patches [140]. As shown in Figure 6.4, we typically obtain clusters of large values for the likelihood of each part around the centers of the parts. We therefore apply a smoothing Gaussian filter on the output of the CNN, and retain only the local maxima of these values as candidates for the locations of the parts.

The result of this step is, for each part p, a set $\mathcal{S}_p = \{(\hat{\mathbf{c}}_{pl}, s_{pl})\}_l$ of 2D location candidates $\hat{\mathbf{c}}_{pl}$ for the part along with a score s_{pl} given by the value of the local maxima returned by the CNN. We will exploit this score in our pose estimation algorithm described in Section 6.4. We typically get up to 4 detections for each part on an input image.

For better robustness to illumination changes, we normalize the patches with a Difference-of-Gaussians:

$$Q = (G^{\sigma_2} - G^{\sigma_1}) * Q' \tag{6.3}$$

where Q' is the original grayscale input patch before normalization, G^{σ_1} and G^{σ_2} are 2D Gaussian kernels of manually selected standard deviations σ_1 and σ_2 respectively, and $*$ the symbol for the product of convolution. We experimentally found this method to perform better than Local Contrast Normalization, which is often the normalization method used with Convolutional Neural Networks.



(a) (b)

(c) (d)

Figure 6.4 – Detecting the parts. (a) An input image of the box. (b) The output of the CNN$^{\text{part-det}}$ for each image location. Each color corresponds to a different part. (c) The output after Gaussian smoothing. (d) The detected parts, corresponding to the local maxima in (c).

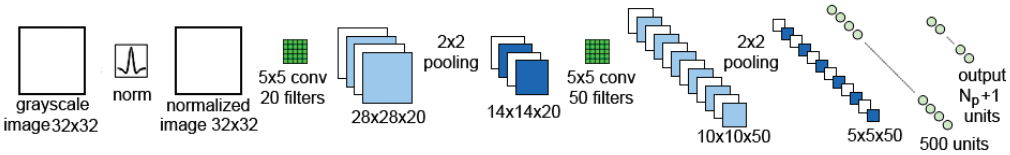


Figure 6.5 – Architecture of CNN$^{\text{part-det}}$ for part detection. The last layer outputs the likelihoods of the patch to correspond to each part or to the background.

6.3 Part Pose Estimation

The second step of our pipeline consists in predicting the pose of each part, starting from information about its local appearance, i.e. an image patch Q extracted on an image I around the projection of the part center $\mathbf{c}$.

6.3.1 Representation of the Part Pose

We now detail how we practically represent the pose of each part, for which different parametrizations are possible. Selecting the part pose representation consists in seeking a function:

$$\mathcal{F} : \ \mathbb{Q} \times \mathbb{R}^2 \ \longrightarrow \ SE(3) \tag{6.4}$$

that, given an image patch $Q \in \mathbb{Q}$ and the part center $\mathbf{c} \in \mathbb{R}^2$, computes the pose of the part $\mathbf{p} = \mathcal{F}(Q, \mathbf{c}) \in SE(3)$ on image I; $\mathbb{Q}$ and $SE(3)$ are, respectively, the space of the image patches of size $S_q \times S_q$, and the space of the 3D rigid transforms.

For a given $\mathbf{c}$, $\mathcal{F}(\cdot, \mathbf{c})$ should be insensitive to imaging changes due to noise, light conditions, etc., and it has no clear analytical form. A judicious choice is to train a non-linear regressor for robustly approximating $\mathcal{F}(\cdot, \mathbf{c})$, learning which appearance changes are induced by viewpoint changes.

Is it possible to train a regressor exclusively based on patches extracted on training images, without any information about the position where each training patch was extracted on the image? In other words, we seek for a pose representation $\mathcal{F}$ that can be decomposed as:

$$\mathcal{F}(Q, \mathbf{c}) = \mathcal{R}(\mathcal{Q}(Q), \mathbf{c}), \tag{6.5}$$

where $\mathcal{Q}(Q)$ is some representation of the pose of the patch *that does not depend on the position of the patch on the image*, and $\mathcal{R}$ is a function that does not depend on the patch appearance, but on the pose representation computed by $\mathcal{Q}$ and possibly from the position of the patch on the image $\mathbf{c}$.

Under a full perspective model, the projection of a non-planar object part already encloses all the information necessary for retrieving the full pose, so $\mathcal{F}(\mathcal{Q}(Q), \mathbf{c}) = \mathcal{F}(\mathcal{Q}(Q))$. This is no longer true when the dimension of the part along the axis perpendicular to the image plane is much smaller than the *depth*, that is, the component of the distance between the camera and the object part along the optical axis: in this case, the projection degenerates to an affine model such as those described in Section 3.3, and it becomes impossible to retrieve the pose of the part represented on a patch without knowing the position of the patch on the image. On the other hand, in this case, for suited representations $\mathcal{Q}(Q)$ the dependence of the pose of a patch Q on its position on the image $\mathcal{F}(\mathcal{Q}(Q), \cdot)$ is a deterministic function.

A crucial point to address is how to define $\mathcal{Q}(\cdot)$, that is, how to choose the most suitable representation for the pose of each part. $\mathcal{Q}$ should satisfy the following constraints:

- Combining the poses of an arbitrary number of parts must be easy and efficient;

- the pose representation should be equally valid under full perspective and under affine projection assumptions;
- since we approximate $\mathcal{Q}$ with a regressor, the pose representation should be tied-in with the regressor's capabilities. For example, as our experimental results show, it is very hard for a regressor to accurately estimate the scale or the depth of a part from a patch.

A priori, we can imagine several ways to represent the 3D poses of the parts:

- **Homography**: it is possible to use homographies for representing the pose of each part [77, 46, 74]. However, this assumes that the part surface is planar, and makes it difficult to merge the individual pose estimations from the different parts.
- **3D Pose**: Another possibility is taking the output of $\mathcal{Q}(Q)$ to consist in a 3D rotation and a depth value for the patch center. It is then possible to retrieve the 3D translation as well, from the location of the patch on the image and the predicted depth. However, it is not easy to merge rotations for estimating the pose of the whole target object. Moreover, this choice requires to predict the depth accurately from a single image patch, which appears to be very difficult to do accurately in our experiments: among others, this task can become ill-conditioned under affine projection assumptions.
- **3D Control points**: Since our final solution is based on 3D control points, as already mentioned, we could set the output of $\mathcal{Q}(Q)$ to be the 3D locations of the control points in the camera reference system. In this case, estimating the pose becomes simple, since it only involves computing the rigid motion between two sets of 3D points [141]. Moreover, also combining the poses of the parts becomes a trivial task, as it boils down to computing the rigid motion between multiple sets of 3D points. However, as it will be shown in Section 6.6.4, accurately predicting the 3D points is difficult.
- **2D reprojections of 3D control points**: This is the representation we proposed in [10]. The part poses are represented as the 2D reprojections of a set of 3D control points. With this representation, it is straightforward to combine the poses of an arbitrary number of parts, by grouping all the 2D reprojections together and solving a PnP problem. Moreover, we do not have to predict the depths or the 3D locations of the control points, which, as noted above, is very difficult to do accurately. Finally, we notice that the representation is adapted to both fully perspective and affine projection models. These advantages entail a significant accuracy gain, as shown by our results in Section 6.6.4. The control points are purely virtual and do not correspond to any physical feature of the parts, therefore we can freely set their configuration. We evaluate different configurations in Section 6.6.5.

6.3.2 Prediction of the Reprojections of the Control Points

Once the parts are detected, we apply a regressor to the patches centered on the candidates $\hat{\mathbf{c}}_{pl}$ to predict the projections of the control points for these candidates. We also implemented this regressor as a CNN, and each part has its specific CNN. As shown in Figure 6.6, these networks

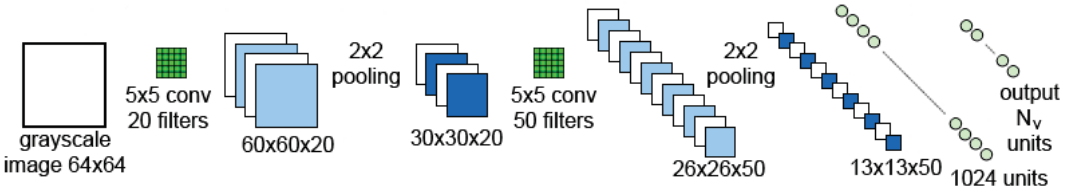


Figure 6.6 – Architecture of the CNN $\text{CNN}^{\text{cp-pred-}p}$ predicting the projections of the control points.

take as input a patch of size of 64×64. The output layer is made of $2N_V$ neurons, with N_V the number of control points of the part, which predicts the 2D locations of the control points. We train each of these CNNs during an offline stage by simply minimizing over the parameters w of the CNN the squared loss of the predictions:

$$\widehat{w} = \arg\min \sum_{(Q,\mathbf{w})\in\mathcal{V}_p} ||\mathbf{w} - \text{CNN}_w^{\text{cp-pred-}p}(Q)||^2 \ , \tag{6.6}$$

where $\mathcal{V}_p$ is a training set of image patches Q centered on part p and the corresponding 2D locations of the control points concatenated in a $(2N_V)$-vector $\mathbf{w}$, and $\text{CNN}_w^{\text{cp-pred-}p}(Q)$ is the prediction for these locations made by the CNN specific for part p, given patch Q as input.

At run-time, we obtain for each $\hat{\mathbf{c}}_{pl}$ candidate, several predictions $\{\hat{\mathbf{v}}_{pkl}\}$ for the control points projections. In addition, we can estimate the 2D uncertainty for the predictions, by propagating the image noise through the CNN that predicts the control point projections [142]. Let us consider the matrix:

$$\mathbf{S}_V = \mathbf{J}_{\hat{\mathbf{c}}}(\sigma\mathbf{I})\mathbf{J}_{\hat{\mathbf{c}}}^\top = \sigma\mathbf{J}_{\hat{\mathbf{c}}}\mathbf{J}_{\hat{\mathbf{c}}}^\top \ , \tag{6.7}$$

where σ is the standard deviation of the image noise assumed to be Gaussian and affect each image pixel independently, $\mathbf{I}$ the $64^2 \times 64^2$ Identity matrix, and $\mathbf{J}_{\hat{\mathbf{c}}}$ the Jacobian of the function computed by the CNN, evaluated at the patch centered on the candidate $\hat{\mathbf{c}}$. Such a Jacobian matrix can be computed easily with a Deep Learning framework such as Theano [143] thanks to the Chain Rule, by multiplying the Jacobians of the successive layers of the network together. By neglecting the correlation between the different control points, we can compute the 2×2 uncertainty matrix $\mathbf{S}_{pk}$ for each control point k efficiently of part p, without having to compute the entire, and very large, product in Equation (6.7):

$$\mathbf{S}_{pk} = \sigma\mathbf{J}_{\hat{\mathbf{c}}}^{pk}\mathbf{J}_{\hat{\mathbf{c}}}^{pk\top} \ , \tag{6.8}$$

where $\mathbf{J}_{\hat{\mathbf{c}}}^{pk}$ is made of the two columns of $\mathbf{J}_{\hat{\mathbf{c}}}$ that correspond to the reprojection of the control point k. An example of predicted control points is shown in Figure 6.2(b).

6.4 Object Pose Estimation

In this section, we detail how we use the predicted reprojections to robustly estimate the object pose.

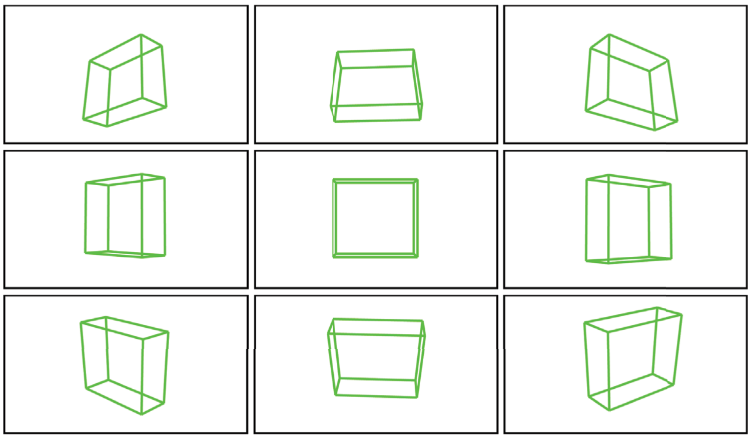


Figure 6.7 – Visualisation of the pose prior for an electric box: Projections of the box by each of the 9 Gaussians centers $\overline{\mathbf{p}}_m$.

As in previous work [144], we assume that we are given a prior on the pose $\mathbf{p}$, in the form of a Mixture-of-Gaussians $\{(\overline{\mathbf{p}}_m, \mathbf{S}_m)\}$. This prior is very general, and allows us to define the normal action range of the camera. For example, the camera is unlikely to be a few centimetres from the object, or more than tens of meters away, or facing the object upside-down. Moreover, the pose computed for the previous frames can be easily incorporated within this framework to exploit temporal consistency.

In the following, we will first assume that this prior is defined as a single Gaussian distribution of mean and covariance $(\overline{\mathbf{p}}_0, \mathbf{S}_0)$. We will extend our approach to the Mixture-of-Gaussians in Section 6.4.3.

6.4.1 Using a Single Gaussian Pose Prior

Let us first assume there is no outlier returned by the part detection process or by the control point prediction, and that all the parts are visible. Then, the object pose $\hat{\mathbf{p}}$ can be estimated as the minimizer of:

$$F(\mathbf{p}) = \frac{1}{N_V N_P} \sum_{p,k} \mathrm{dist}^2(\mathbf{S}_{pk}, \mathcal{P}_{\mathbf{p}}(\mathbf{V}_{pk}), \hat{\mathbf{v}}_{pk}) \; + (\mathbf{p} - \overline{\mathbf{p}}_0)^\top \mathbf{S}_0^{-1} (\mathbf{p} - \overline{\mathbf{p}}_0) \quad , \tag{6.9}$$

where the sum is extended over all the control points of all the parts, and $\mathcal{P}_{\mathbf{p}}(\mathbf{V})$ is the perspective projection described in Section 3.2 of the 3D point $\mathbf{V}$ under pose $\mathbf{p}$. $\hat{\mathbf{v}}_{pk}$ is the projection of control point $\mathbf{V}_{pk}$ and $\mathbf{S}_{pk}$ its uncertainty estimated as in Equation (6.8). Since we assume here that there is no outlier, we dropped here the l index corresponding to the multiple detections that may occur for a single part. $\mathrm{dist}(.)$ is the Mahalanobis distance:

$$\mathrm{dist}^2(\mathbf{S}, \mathbf{v}_1, \mathbf{v}_2) = (\mathbf{v}_1 - \mathbf{v}_2)^\top \mathbf{S}^{-1} (\mathbf{v}_1 - \mathbf{v}_2) \quad . \tag{6.10}$$

$F(\mathbf{p})$ is minimized using the Gauss-Newton algorithm initialized with $\overline{\mathbf{p}}_0$. At each iteration, we update the estimated covariance of the computed pose using the Extended Kalman Filter update formula [142] when optimizing Equation (6.9).

6.4.2 Outlier Rejection for the Detected Parts

In practice, for the location of the p-th part, the detection procedure described in Section 6.2 returns a set of hypotheses $\mathcal{S}_p = \{\hat{\mathbf{c}}_{pl}\}_l$, among which at most one is correct. To reject outliers in detection, we exploit the fact that for each part there is at most one true positive detection, and, similarly to [144], we exploit the pose prior to select the most likely set of detections: For each part, after ranking the candidates according to their score s_{pl}, we consider only the best three candidates; after that, we form all the possible sets $\mathcal{C} = \{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_p, \ldots\}$ of detections containing at most one candidate for each part. Given the pose prior $\overline{\mathbf{p}}_0$, we evaluate all the sets of candidates $\mathcal{C}$ with the following steps:

1. Select two random candidates $\hat{\mathbf{c}}_{p_1},\ \hat{\mathbf{c}}_{p_2} \in \mathcal{C}$, and translate the pose prior $\overline{\mathbf{p}}_0$ to obtain a new prior $\overline{\mathbf{p}}_0^{TS}$ that best fits $\hat{\mathbf{c}}_{p_1},\ \hat{\mathbf{c}}_{p_2}$. More exactly, we adjust the in-plane translation such that:

$$\mathcal{P}_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_{p_1}) + \mathcal{P}_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_{p_2}) = \hat{\mathbf{c}}_{p_1} + \hat{\mathbf{c}}_{p_2} \tag{6.11}$$

   and the off-plane translation component such that:

$$||\mathcal{P}_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_{p_1}) - \mathcal{P}_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_{p_2})|| = ||\hat{\mathbf{c}}_{p_1} - \hat{\mathbf{c}}_{p_2}||. \tag{6.12}$$

2. We keep considering $\mathcal{C}$ only if all the detections it contains are consistent with the new prior. This test can be formalized as:

$$\begin{array}{ll} \forall \hat{\mathbf{c}}_p \in \mathcal{C}: & \rho_p < T^2 \\ \text{with} & \rho_p = \mathrm{dist}^2(\hat{\mathbf{S}}_0(\mathbf{C}_p), \mathcal{P}_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_p), \hat{\mathbf{c}}_p) \end{array} \tag{6.13}$$

   where $\hat{\mathbf{S}}_0(\mathbf{C}_p) = \mathbf{J}\ \mathbf{S}_0\mathbf{J}^\top$, with $\mathbf{J}$ the jacobian of $\mathcal{P}_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_p)$, is the covariance of the projected control point $\mathcal{P}_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_p)$; we set the threshold $T = 40$ pixels in all our experiments.

3. If several sets $\mathcal{C}$ pass this test, we retain the one with the largest number of detected parts. If several retained sets have the same number of points, we keep the one with the smallest average error $\overline{\rho} = \frac{1}{|\mathcal{C}|}\sum_p \rho_p$ of its points.

4. Finally, we run the Gauss-Newton optimization of Equation (6.9) using the detections in the retained set to obtain a pose estimate.

If the object of interest has a single part, we simply select the detection candidate with the highest score.

6.4.3 Using a Mixture-of-Gaussians for the Pose Prior

In practice, the prior for the pose is in the form of a Mixture-of-Gaussians $\{(\overline{\mathbf{p}}_m, \mathbf{S}_m)\}_m$ with $M = 9$ components. The prior we use for the BOX dataset is shown in Figure 6.7. We apply the method described in Sections 6.4, 6.4.2 to each component, and obtain M possible pose estimates: $\hat{\mathbf{p}}^{(1)}, \ldots, \hat{\mathbf{p}}^{(M)}$.

6.4.4 Identifying the Best Pose Estimate

To finally identify the best pose estimate $\hat{\mathbf{p}}$ among the different estimates obtained with the Mixture-of-Gaussians prior, we evaluate each $\hat{\mathbf{p}}^{(m)}$ using several cues. As it is difficult to combine cues of different natures, our key idea here is to train a linear regressor to weight the contributions of the different cues and predict a penalty.

More exactly, we use the scale difference δ_{scale} and the angle α between the quaternions for $\hat{\mathbf{p}}^{(m)}$ and the corresponding component of the prior, the final value of the objective function $F(\hat{\mathbf{p}}^{(m)})$ defined in Equation (6.9), and a score $\xi(\hat{\mathbf{p}}^{(m)})$ measuring the correlation between the edges in the image and the object contours after projection by $\hat{\mathbf{p}}^{(m)}$. $\xi(\hat{\mathbf{p}}^{(m)})$ is computed as:

$$\xi(\hat{\mathbf{p}}^{(m)}) = \sum_{\mathbf{x}} \left(\mathbf{n}(\mathbf{x}) \cdot [I_u(\mathbf{x}), I_v(\mathbf{x})]^\top \right) \ , \tag{6.14}$$

where $\mathbf{n}(\mathbf{x})$ is the unit normal of the projected object contour at pixel $\mathbf{x}$, $I_u(\mathbf{x})$ and $I_v(\mathbf{x})$ are the partial derivatives of the incoming image I at pixel $\mathbf{x}$, and the sum is over the pixels $\mathbf{x}$ lying on the re-projected contours of the object.

Offline, we create a training set generated from the training sequence by adding noise to the ground truth poses, and computing the values of our different cues. For each sample, we compute a penalty that is the sum of the euclidean norms of the rotation and translation components of the absolute pose error [145] introduced by the noise. We can then train a linear regressor to predict this penalty given our different cues. At run-time, we simply have to use the linear regressor to predict the penalties of the pose estimates, and keep the one with the smallest penalty.

6.5 Tracking Frames across a Video Sequence and Pose Filtering

When tracking an object across a video sequence, if a pose is estimated for a given frame, we add it as a new component of the Mixture-of-Gaussians pose prior for the next frame. This allows us to easily take advantage of temporal constraints within our framework. Moreover, we use a Kalman Filter for reducing jitter and provide smoother trajectories.

6.5.1 Extended Kalman Filter for 3D Tracking

In visual tracking, Kalman Filters typically treat images as observations. However, this requires the linearisation of the imaging process with respect to the 3D pose, which can result in a poor approximation. Therefore, we chose to consider our pose estimation method described in Sections 6.1- 6.4 as a "black box", and we treat the poses it predicts as *observations* for the filter, alleviating the need for linearisation.

State Vector

We model the camera motion as a first order, discrete-time dynamic system, and the state vector at time t is provided by the column vector of size 12:

$$\mathbf{s}_t = [\mathbf{t}_t^\top, \mathbf{r}_t^\top, \mathbf{v}_t^\top, \omega_t^\top]^\top \quad , \tag{6.15}$$

where $\mathbf{t}_t$ is the translation component of the camera pose, $\mathbf{r}_t$ is the exponential map representation of the rotation component of the pose, $\mathbf{v}_t$ is the linear velocity and ω_t the angular velocity.

At each time step, our estimation of the system state is updated according to the available observations with the *predictor-corrector* scheme of Kalman Filters. First, the state estimate $\tilde{\mathbf{s}}_{t-1}$ and its covariance $\tilde{\mathbf{S}}_{t-1}$ are updated with a motion model for predicting the state at current time $\tilde{\mathbf{s}}_{t-1}^t$ and the covariance $\tilde{\mathbf{S}}_{t-1}^t$. Then, the observation of the current state is employed for correcting the initial prediction and obtain the final state estimation $\tilde{\mathbf{s}}_t$.

Notations

For sake of clarity, we summarize here the notation convention employed for the Kalman filter described in this section. For a given quantity $\mathbf{x}$, then:

- $\tilde{\mathbf{x}}_{t-1}$ is the estimate of $\mathbf{x}$ at the end of step $t-1$;
- $\tilde{\mathbf{x}}_{t-1}^t$ is the estimate of $\mathbf{x}$ at time t predicted by updating $\tilde{\mathbf{x}}_{t-1}$ according to some dynamic model;
- $\hat{\mathbf{x}}_t$ is the observed value of $\mathbf{x}$ at step t, typically the camera pose predicted by the method described above.
- $\tilde{\mathbf{x}}_t$ is the final estimate of $\mathbf{x}$ at time t, obtained correcting $\tilde{\mathbf{x}}_{t-1}^t$ according to the observation $\hat{\mathbf{x}}_t$.

Predictor: State Update

The state at each time step is predicted from the estimate of the state at the previous time step using the following motion model:

$$\begin{aligned}
\tilde{\mathbf{t}}^t_{t-1} &= \tilde{\mathbf{t}}_{t-1} + \delta t \; \tilde{\mathbf{v}}_{t-1} \\
\tilde{\mathbf{r}}^t_{t-1} &= \tilde{\omega}_{t-1} \; \circ \; \tilde{\mathbf{r}}_{t-1} \\
\tilde{\mathbf{v}}^t_{t-1} &= \tilde{\mathbf{v}}_{t-1} \\
\tilde{\omega}^t_{t-1} &= \tilde{\omega}_{t-1} \;\; ,
\end{aligned} \tag{6.16}$$

where δt is the time difference between 2 subsequent time steps, $\circ$ denotes the composition of rotations. Without loss of generality, we will take $\delta t = 1$.

The covariance of the state is updated using:

$$\tilde{\mathbf{S}}^t_{t-1} = \mathbf{J}_{update} \tilde{\mathbf{S}}_{t-1} \mathbf{J}^\top_{update} + \mathbf{A} \;\; , \tag{6.17}$$

where $\mathbf{J}_{update}$ is the $(12 \times 12)-$jacobian matrix of the update (6.16), and $\mathbf{A}$ is given by :

$$\mathbf{A} = \begin{bmatrix} \frac{1}{3}a\mathbf{I} & 0 & \frac{1}{2}a\mathbf{I} & 0 \\ 0 & \frac{1}{3}b\mathbf{I} & 0 & \frac{1}{2}b\mathbf{I} \\ \frac{1}{2}a\mathbf{I} & 0 & a\mathbf{I} & 0 \\ 0 & \frac{1}{2}b\mathbf{I} & 0 & b\mathbf{I} \end{bmatrix} , \tag{6.18}$$

where $\mathbf{I}$ is the $(3 \times 3)-$identity matrix, and a and b are 2 parameters corresponding to the incertitude about the temporal derivatives of the velocities. We empirically set $a = b = 100$ in all our experiments. Interested readers can refer to [146] and its references for further details about the derivation of matrix $\mathbf{A}$.

Corrector: Taking into Account Observations

After computing a prediction of the current state and its covariance, we correct it taking into account our observation, the pose $\hat{\mathbf{p}}_t$. Since we cannot observe the velocities directly, their estimations would stay indefinitely stuck in the initial state if we only use the motion model of Equation (6.16). To avoid this problem, we compute the "observed" velocities as:

$$\hat{\mathbf{v}}_t = (\hat{\mathbf{t}}_t - \tilde{\mathbf{t}}_{t-1})/\delta t \quad \text{and} \quad \hat{\omega}_t = \omega(\tilde{\mathbf{r}}_{t-1}, \hat{\mathbf{r}}_t) \;\; ; \tag{6.19}$$

the angular velocity $\omega(\mathbf{r}_1, \mathbf{r}_2)$ between 2 consecutive rotations $\mathbf{r}_1,\ \mathbf{r}_2$ is estimated with the log mapping of Equation (3.18):

$$\begin{aligned}
&\mathbf{R}_1 = \mathbf{R}(\mathbf{r}_1)\ , \qquad \mathbf{R}_2 = \mathbf{R}(\mathbf{r}_2)\ , \qquad \delta\mathbf{R} = \mathbf{R}_2\mathbf{R}_1^\top\ , \\
&\theta = \mathrm{acos}\left(\frac{\mathrm{trace}(\delta\mathbf{R}) - 1}{2}\right)\ , \qquad \Omega = \frac{\theta}{2\delta t \sin(\theta)}(\delta\mathbf{R} - \delta\mathbf{R}^\top)\ , \\
&\omega(\mathbf{r}_1, \mathbf{r}_2) = [\Omega_{32}, \Omega_{13}, \Omega_{21}]^\top,
\end{aligned}$$

where $\mathbf{R}(\mathbf{r})$ is the $(3 \times 3)-$rotation matrix corresponding to the rotation vector $\mathbf{r}$, computed with Equation (3.17) and Ω_{ij} denotes the element at the i-th row and j-th column of the 3×3 matrix Ω. We set $\omega\ =\ [0,\ 0,\ 0]^\top$ if $||\theta||$ is smaller than a threshold for preventing division by 0.

At first sight, Equation (6.19) may seem arbitrary: the observed velocities could be estimated from the observed poses for 2 subsequent frames, that is, employing $\hat{\mathbf{t}}_{t-1}^\top$ and $\hat{\mathbf{r}}_{t-1}^\top$, instead of, respectively, $\tilde{\mathbf{t}}_{t-1}$ and $\tilde{\mathbf{r}}_{t-1}$ in Equation (6.19). According to our experiments, this is not a good choice, since pose observations for successive frames may be affected by jitter, so that velocities estimated exclusively based on observed poses may lead to completely inconsistent results; computing observed velocities with Equation (6.19) sensibly lowers the jitter nuisance.

As covariance of the observed state $\hat{\mathbf{S}}_t$, we employ a constant, diagonal covariance matrix:

$$\hat{\mathbf{S}}_t = \alpha \begin{bmatrix} \mathbf{I} & 0 & 0 & 0 \\ 0 & \mathbf{I} & 0 & 0 \\ 0 & 0 & \beta\mathbf{I} & 0 \\ 0 & 0 & 0 & \beta\mathbf{I} \end{bmatrix}, \tag{6.20}$$

where we empirically set $\alpha = 10^{-3}$ and $\beta = 10$.

Finally, we simply apply standard Kalman update equations for correcting the state estimate:

$$\begin{aligned}
\mathbf{y} &= \hat{\mathbf{s}}_t - \tilde{\mathbf{s}}_{t-1}^t \\
\mathbf{K} &= \tilde{\mathbf{S}}_{t-1}^t\left(\tilde{\mathbf{S}}_{t-1}^t + \hat{\mathbf{S}}_t\right)^{-1} \\
\tilde{\mathbf{s}}_t &= \tilde{\mathbf{s}}_{t-1}^t + \mathbf{K}\mathbf{y} \\
\tilde{\mathbf{S}}_t &= \left(\mathbf{I}^{12} - \mathbf{K}\right)\tilde{\mathbf{S}}_{t-1}^t,
\end{aligned} \tag{6.21}$$

where matrix $\mathbf{K}$ is called the Kalman gain and $\mathbf{I}^{12}$ is the 12×12 identity matrix.

Initialization - Outlier Rejection

For the first frame of the video sequence, we initialize the state vector estimate with $\hat{\mathbf{p}}_t$ and null velocities. Special care must be taken in order to detect and reject outliers in the observed poses. In practice, we use the following tests:

- if an observed pose $\hat{\mathbf{p}}_t$ is not close to the last estimation $\tilde{\mathbf{p}}_{t-1}$, then it is probably an outlier and should not be taken into account;
- if 2 consecutive observed poses $\hat{\mathbf{p}}_{t-1}$ and $\hat{\mathbf{p}}_t$ are close to each other, then they are probably not outliers, even if they are far from the last pose estimate.

If the observed pose $\hat{\mathbf{p}}_t$ is detected as an outlier according to these tests, we then set $\tilde{\mathbf{s}}_t := \tilde{\mathbf{s}}^t_{t-1}$. If outlier poses are observed for more than 3 frames in a row, we assume that tracking is lost. Tracking is then automatically re-initialized with the observed pose as soon as 2 consecutive poses are observed, sufficiently close to each other.

6.6 Experimental Results

In this section, after describing the datasets we use for evaluating our part-based method in Section 6.6.2, we present and discuss the results of our evaluation. In Section 6.6.3 we assess the effectiveness of our detector method, as well as that of the Diffference-of-Gaussians (DoG) Normalization introduced in Section 6.2. In Section 6.6.4 we evaluate different pose representations introduced in Section 6.3.1, showing that our representation based on the reprojections of control points entails substantial performance gain. Then, in Sections 6.6.6 - 6.6.8 we present the results of an extensive comparison with other methods, showing that our approach achieves state-of-the-art performances on our challenging sequences.

6.6.1 Evaluation Protocol

In order to quantitatively evaluate the performances of a method on a video sequence, we compute the rotation and translation components of the absolute pose error [145] for each frame, and then trace their Cumulative Distribution Functions (CDF), as shown for example in Figures 6.14, 6.15, 6.16. The normalized Area Under Curve (AUC) score, defined as the integral of the CDF curve divided by the maximum error (0.5 in all our graphs), is reported for facilitating comparisons between methods, for example in Table 6.3. The translation error is in meters, while the rotation error is a pure number. We employed CDF curves and AUC scores also for evaluating the performances of different detectors in Section 6.6.3. In this case, the detection error is expressed in pixels.

6.6.2 Datasets

At the best of our knowledge, no state-of-the-art method has been tested on objects undergoing heavy occlusions and clutter as shown in Figure 6.1. For this reason, we run our extensive evaluations on the datasets originally introduced in [10], consisting of both learning data and testing video sequences representing several non-textured, highly occluded objects. The dataset for each object includes a non-textured CAD model and the groundtruth pose. All the images are in the VGA resolution (640×480). For each dataset, we randomly select 3000 frames from the training images as training set. We test our approach on the following datasets:

- **BOX Dataset:** The target object for this dataset is an electric box. In the test videos, it is manipulated by a user, filled and emptied with objects, simulating, for example, a maintenance intervention by a technician. The training images show the box on a uniform background, with different objects inside and outside it. A CAD model is made by a simple parallelepiped. We use 4 corners of the box as parts, as shown in Figure 6.9(a).
- **CAN Dataset:** The target object of this dataset is a food can. The label is completely blank, and the top of the can is specular. Distractor objects are present in the scene and large occlusions occur. Only the can lid breaks the the cylindrical symmetry of the object, making the pose estimation almost ambiguous. We use the top of the can as a single part, Figure 6.9(b). A CAD model of the can is provided.
- **DOOR Dataset:** This datasets consists of one video showing a daily set-up where a non-textured door is opened and closed by a user. Despite the apparent triviality of the sequence, our tests show that it is very challenging to track the pose of the door along the full video, when it moves on a cluttered background. For this dataset, we track the 3 parts shown in Figure 6.9(c), the knob, the keyhole and the lock of the door. A simple CAD model of the door is available as well.

The images of the training and testing videos of the datasets were registered using the ARUCO marker tracking tool [147]. The markers on the test sequences were cropped or masked, so that they could not influence detection and tracking performance when testing the methods.

We also manually labelled the ground-truth locations of the detected parts for all the test video sequences of the original dataset presented in [10], so that more accurate experiments for evaluating the detector can be performed, such as those presented in Section 6.6.3. The manually labelled parts have also been employed for refining the ground-truth poses. Because of this, some of the experimental results presented in this work may be numerically slightly different from the ones reported in [10], although no substantial difference in the results has been detected. All the refined datasets are publicly available at `http://cvlab.epfl.ch/data/3d_object_tracking`.

6.6.3 Part Detection

Our pipeline does not depend on a particular choice of a detector for localizing the object parts on the image. Nonetheless, the detector described in Section 6.2 provides an excellent trade-off between speed and accuracy: We assess here our choice by comparing it with a state-of-the-art detector, LINE-2D [44].[1] In this case, we trained an instance of LINE-2D for each part, starting from 32 × 32 RGB patches surrounding the part of interest. The amount of learning data was the same as for our CNN-based detector.

---

[1] For all the tests presented in this chapter, we employed the LINE-2D implementation provided by OpenCV-2.4.12. Implementation of the authors was used for LSD-SLAM and PWP3D.

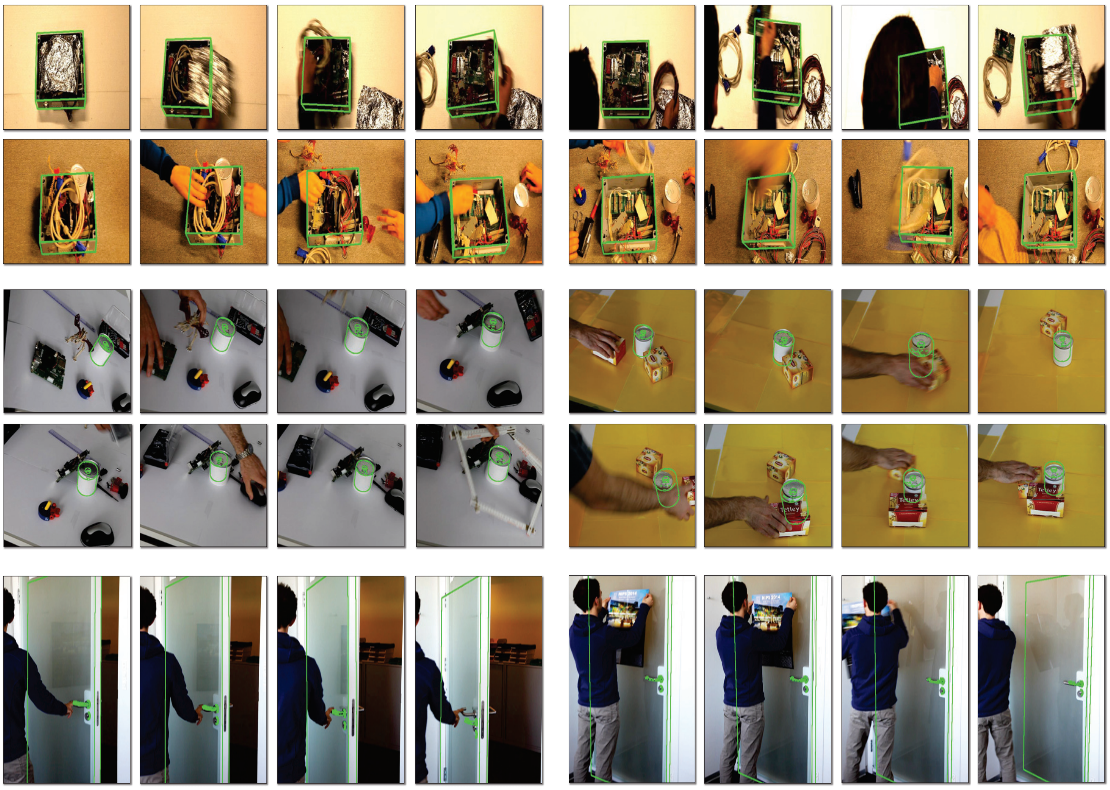
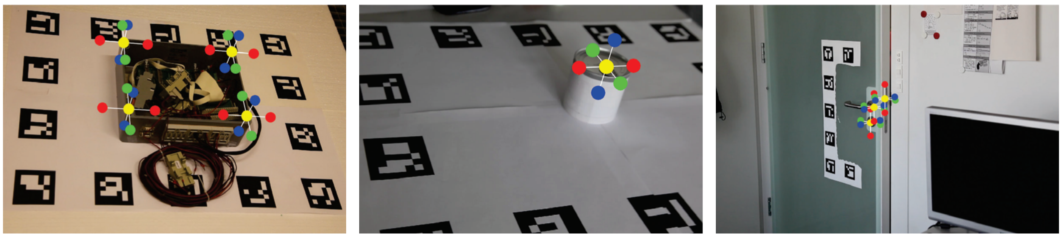


Figure 6.8 – Qualitative results for our challenging datasets. **Top:** We track the box despite large changes in the background and in the lighting conditions on both sequences of the BOX dataset. **Middle:** Our method correctly estimates the 3D pose of the can using the can tab only. **Bottom:** The pose of the door is retrieved starting from the door knob, the keyhole and the lock.



(a) (b) (c)

Figure 6.9 – Training images and control points we used for the BOX, the CAN and the DOOR datasets. The center of each part is shown in yellow. Control points are zoomed for better visualization.

Experiment	BOX dataset Video #1				BOX dataset Video #2			
	Part #1	Part #2	Part #3	Part #4	Part #1	Part #2	Part #3	Part #4
$\text{CNN}^{\text{part-det}}$	0.82	0.75	0.89	0.94	0.45	0.80	0.43	0.82
$\text{CNN}^{\text{part-det}}$ +DoG	0.87	0.88	0.90	0.95	0.44	0.90	0.74	0.92
LINE-2D	0.30	0.27	0.63	0.60	0.29	0.10	0.55	0.59

Table 6.2 – Detection error results for the BOX Dataset. We report the AUC scores for the detection error relative to each part, as described in Section 6.6.3.

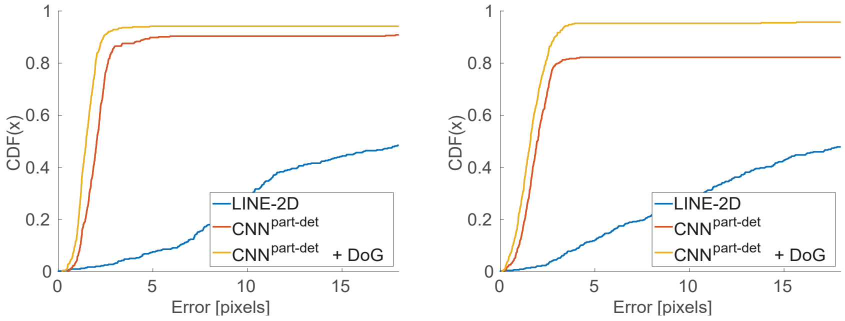
At test time, we kept the best 4 candidates in each image for each detector and computed the detection error as the euclidean norm between the ground-truth position of the part on the image and the closest detection candidate. The CDF curves for the BOX dataset are shown in Figure 6.10, while AUC scores are reported in Table 6.2. We also assessed the importance of the DoG normalization introduced in Section 6.2. For all parts, our detector consistently outperforms LINE-2D, and the DoG normalization further increases performances in most of the cases.

In both videos, LINE-2D performs reasonably well on the upper corners of the box -parts #3 and #4- while the accuracy for the two other corners is much lower. This is probably due to the fact that in our test dataset, the edges of the upper corners are often visible against a bright background and their shapes are easily recognizable. We also observed that DoG normalization is particularly effective for the Video #2, where the lighting conditions and the background are completely different from the training videos, as opposed to the Video #1. Finally, we noticed that the scores of all detectors for the Video #2, for the bottom-left corner (Part #1) is significantly lower. This is probably due to the fact that at about half of the sequence a distractor object is very close to the part, altering its appearance, and the shadow patterns change frequently around this part. Still, we can accurately predict the pose of the target object pose because the other parts are reliably detected.

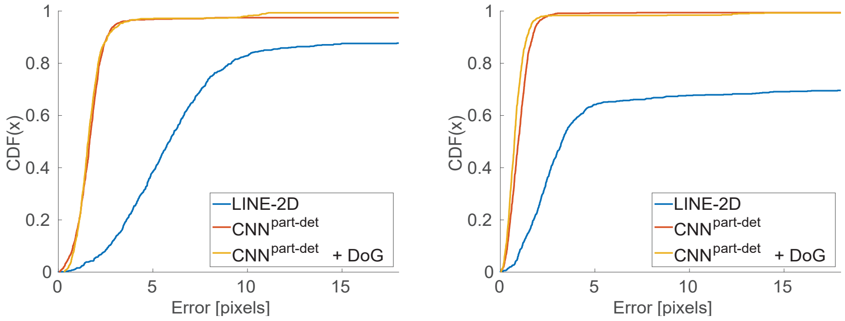
6.6.4 Validation of the Part Pose Representation

To validate our part pose representation based on the 2D reprojections of 3D control points introduced in Section 6.3, we trained several regressor CNNs for predicting the object pose of all the frames of the first video of the BOX Dataset. Each CNN was trained to predict a different part pose representation, which yields to different strategies to combine the contributions of the different parts:

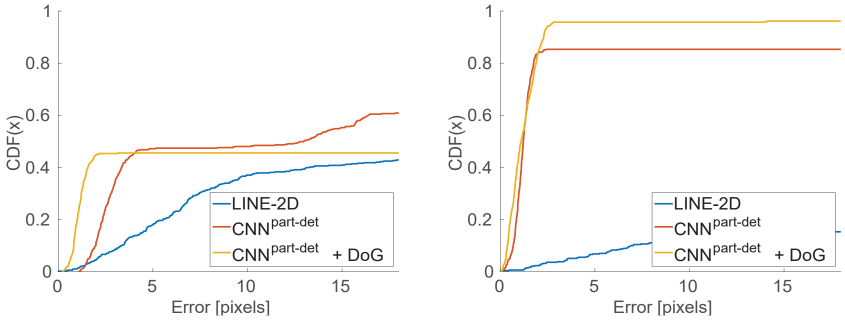
- **Averaging Poses:** The output of the CNN is a 3D rotation and a depth for each part. The in-plane components of the translation are retrieved from the position of the patch on the image. The full object pose is then obtained by averaging the parts poses. Rotations were averaged as proposed in [148].
- **3D Control Points:** The predicted representation is made by the coordinates of the 3D



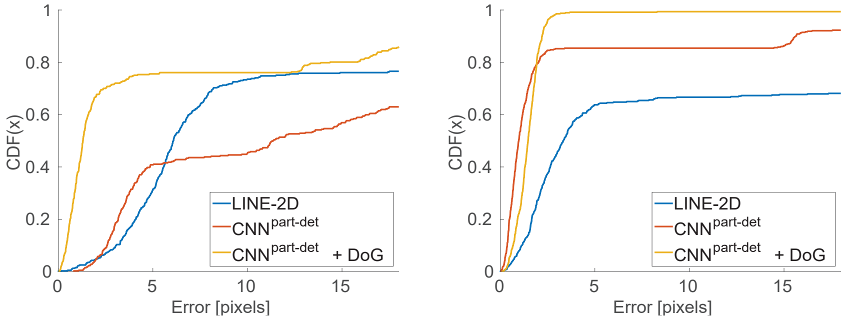
(a) BOX - Video #1 - Part 1 (b) BOX - Video #1 - Part 2



(c) BOX - Video #1 - Part 3 (d) BOX - Video #1 - Part 4



(e) BOX - Video #2 - Part 1 (f) BOX - Video #2 - Part 2



(g) BOX - Video #2 - Part 3 (h) BOX - Video #2 - Part 4

Figure 6.10 – Results of the experiment described in Section 6.6.3: detection error Cumulative Distribution Functions (CDF) for the BOX dataset for different detectors. Top row: Video #1. Bottom row: Video #2.

control points shown in Figure 6.2 in the camera reference system. Since the 3D coordinates of the control points in the camera system depend on the position of the patch on the image, we employ the following indirect estimation: The output of the CNN consists in a depth value for the center of the patch, and a set of offsets for all the other control points $\{(\delta x/\delta z, \delta y/\delta z, \delta z)\}_k$. The 3D locations of all the control points can be straightforwardly retrieved from the predicted values. The poses of the parts are then estimated and combined by computing the 3D rigid transform aligning the points in the camera and in the world reference system in a least-square sense [149].

- **2D Reprojections of 3D Control Points:** The output of the CNN is given by the coordinates of the reprojections of the control points on the image patch, as described in Section 6.3. The pose is computed by solving the PnP problem after gathering all the 3D-2D correspondences given by all the parts.

The results are shown in Figure 6.11. The last choice entails a significant accuracy gain over the previous ones.

The performance gap between the **3D Control Points** and the **2D Reprojections of 3D Control Points** representations may seem somehow surprising, since 2 of the 3 predicted coordinates for the **3D Control Points** representation basically coincide with the ones of the **2D Reprojections of 3D Control Points**. This suggests that the regressor may not predict all the degrees of freedom with the same accuracy.

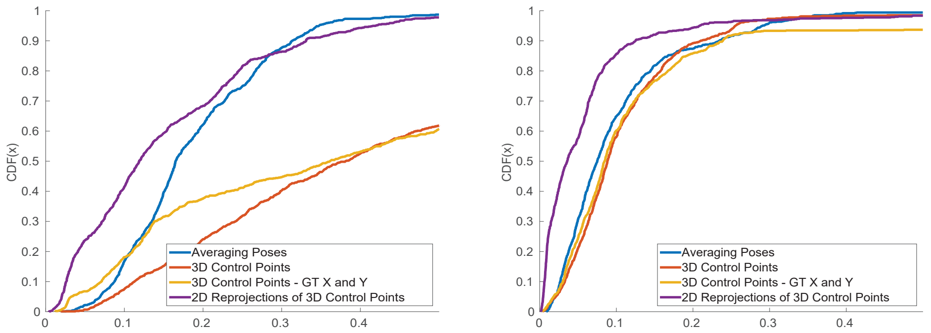
In order to further investigate this aspect, we performed two other experiments:

- we evaluated the errors of the poses obtained replacing the predicted 2D reprojections of the **3D Control Points** experiment by their ground truth (**3D Control Points - GT X and Y**) values;
- instead of replacing the 2D reprojections by the ground truth, we replaced the depths by their ground truth (**3D Control Points - GT Depth**).

In the first case, the results did not improve much over the **3D Control Points** baseline. In the second case, the results are equivalent to the ones of **2D Reprojections of 3D Control Points** (for sake of clarity, the **3D Control Points - GT Depth** curve is not shown in Figure 6.11). This shows that predicting the depths is a much more difficult task than predicting the 2D locations.

6.6.5 Virtual Points Configuration

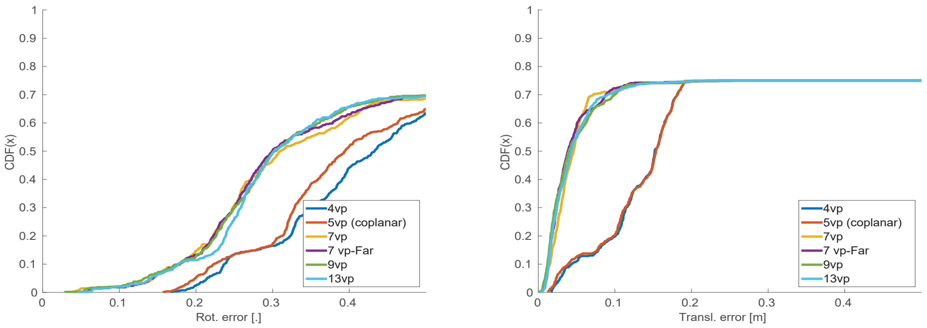
In order to assess the influence of the number and configurations of control points on the accuracy of our method, we tested the configurations shown in Figure 6.13 on the CAN dataset. We created different configurations with an increasing number of virtual points, and disposed them regularly around the part center.



(a) Rotation error CDF: BOX - Video #1

(b) Translation error CDF: BOX - Video #1

Figure 6.11 – The rotation and translation error Cumulative Distribution Functions (CDF) on the BOX dataset, Video #1 for the parametrizations of the part poses presented in Section 6.6.4. Our pose representation entails a substantial performance gain.



(a) Rotation error CDF: CAN - Video #1

(b) Translation error CDF: CAN - Video #1

Figure 6.12 – Rotation and translation error Cumulative Distribution Functions (CDF) for the configurations of virtual points shown in Figure 6.13 for the CAN dataset-Video #1. Best results are obtained by the configurations spanning the 3 orthogonal axes. Increasing the number of virtual points does not improve results above 7 virtual points.

The comparison is performed on the CAN dataset, probably the most challenging one, because the object of interest is tracked using a single part, so we expect the pose estimation results to be particularly sensitive to the disposition and number of control points. We trained one regressor for each of the configurations shown in Figure 6.13 from the same learning data, and run the pose estimation for each configuration, starting from the same detection candidates for the can lid. Results are shown in Figure 6.12. In general, we observed that:

- configurations spanning the 3 orthogonal directions perform better than planar configurations;

- increasing the number of control points improves results up to 7 points, while no noticeable improvement is obtained by using configurations with more points.

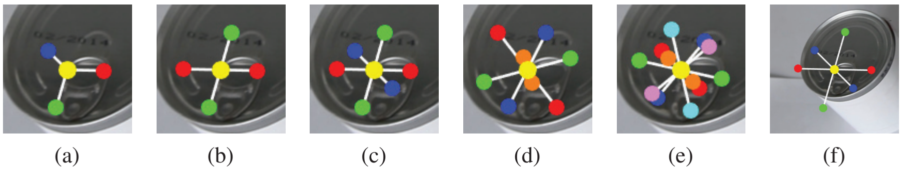


Figure 6.13 – Different configurations of control points tested on the CAN dataset, with (a) 4 control points spanning the 3 axes; (b) 5 co-planar control points; (c) 7 control points spanning the 3D axes; (d) 9 control points disposed in the center and on the corners of a cube; (e) 13 control points disposed in the center and on the corners of an icosahedron; (f) 7 control points spanning the 3 axes, with a larger spacing.

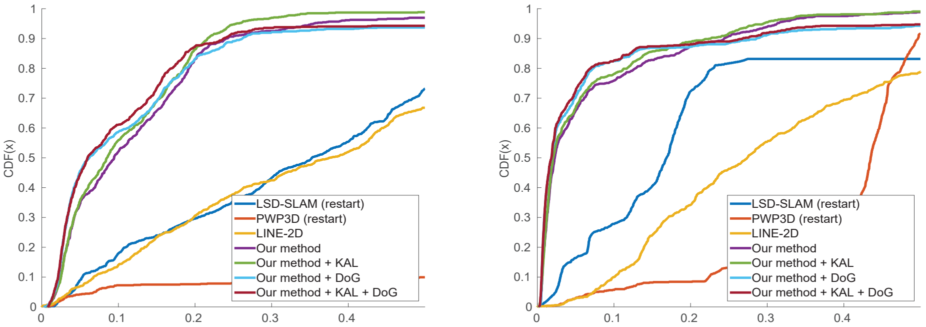
Experiment	BOX dataset		CAN dataset		DOOR dataset
	Video #1	Video #2	Video #1	Video #2	Video #1
nb. of frames	892	500	450	314	564
LSD-SLAM	0.37 - 0.61*	0.48- 0.63	0.17 - 0.29	0.38 - 0.48	0.50 - 0.38
PWP3D	0.10 - 0.20*	0.16 - 0.52	0.13 - 0.64	0.13 - 0.51	0 - 0
LINE-2D	0.34 - 0.41	0.34 - 0.44	0.20 - 0.62	0.29 - 0.65	0.13 - 0.14
Our method [10]	0.75 - 0.85	0.57 - 0.85	0.35 - 0.85	0.51 - 0.70	0.72 - 0.61
Our method - KAL	0.78 - 0.86	0.65 - 0.88	0.36 - 0.86	0.51 - 0.70	**0.79 - 0.66**
Our method - DoG	0.76 - 0.85	0.80 - 0.88	0.42 - 0.92	0.52 - 0.74	0.76 - 0.69
Our method - KAL+DoG	**0.78 - 0.86**	**0.82 - 0.90**	**0.42 - 0.93**	**0.55 - 0.75**	**0.76 - 0.70**

Table 6.3 – Experimental results for our part-based framework. We report the AUC scores for the rotation and the translation errors for the five video sequences of our datasets. A star (*) after the scores indicates that the method was re-initialized with the groundtruth for frame 500. We report results of our method as originally implemented in [10], as well as with the contributions of the Kalman filter (KAL) and the patch normalization (DoG). Both improvements sensibly enhance performances on all datasets.

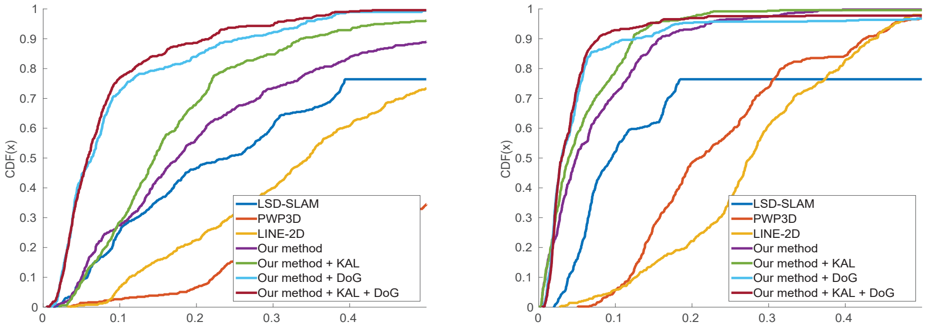
6.6.6 Comparison against the State-of-the-Art

We compared our approach with three state-of-the-art methods, LINE-2D [44], PWP3D [32] and LSD-SLAM [86]. LINE-2D proceeds using very fast template matching. PWP3D is an accurate and robust model-based 3D tracking method based on segmentation. LSD-SLAM is a recent, powerful and reliable SLAM system: amongst other things, it does not require prior 3D knowledge, while we know the 3D locations of the control points and their appearances. The comparison should therefore be taken with caution, as this method does not aim to achieve exactly the same task as us. Nevertheless, we believe the comparison highlights the strengths and weaknesses of the compared methods.

For every test video, we compare the poses computed by each method for all frames. Following the evaluation framework in [145], we align each of the trajectories with respect to the same reference system. In each test, the templates for LINE-2D were extracted by the same images we employed for training our method. PWP3D was manually initialized using the ground-truth pose data, while



(a) Rotation error CDF: BOX - Video #1 (b) Translation error CDF: BOX - Video #1



(c) Rotation error CDF: BOX - Video #2 (d) Translation error CDF: BOX - Video #2

Figure 6.14 – The rotation and translation error Cumulative Distribution Functions (CDF) on the BOX dataset. (a),(b): comparative results for Video #1: LSD-SLAM and PWP3D were both re-initialized with the groundtruth at frame 500. (c),(d): comparative results for Video #2. Here the DoG normalization is particularly effective in compensating light changes and entails a significant performance gain.

LINE-2D, LSD-SLAM and our method do not require any initial pose.

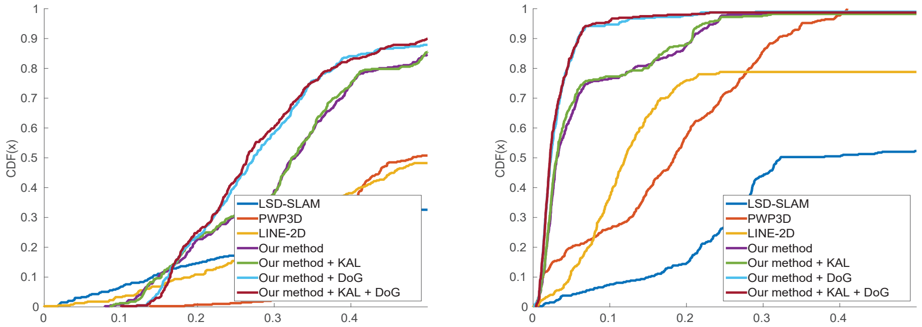
6.6.7 Training Details

Our CNNs were trained employing Stochastic Gradient Deschent with a batch size of 128 samples and 60 epochs; learning rate was set to 0.01 for all the models. Each original learning set was augmented with synthetic examples obtained randomly translating, scaling and rotating patches extracted from real images; the final size of each learning set was 500 000 patches for detection, and 300 000 for estimation of the part poses.

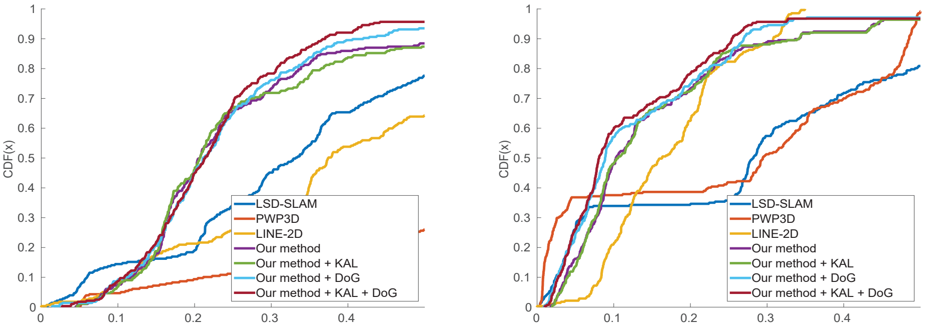
6.6.8 Results

Quantitative results of our tests are shown in Table 6.3. LINE-2D, LSD-SLAM, and PWP3D actually fail very frequently on our sequences, drifting or loosing track.

In the BOX dataset, on the longest of our video sequences, we also re-initialized LSD-SLAM and

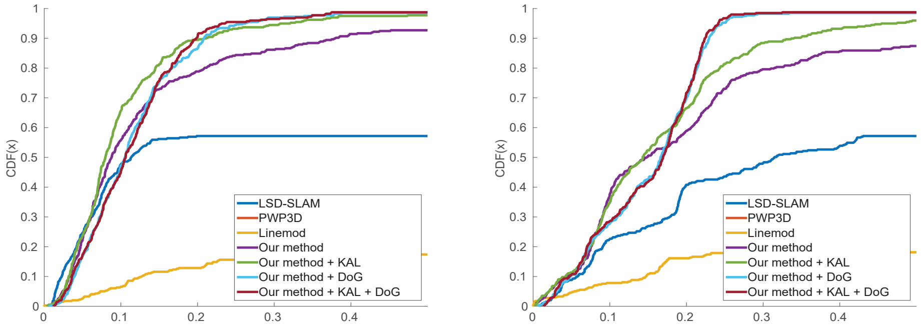


(a) Rotation error CDF: CAN - Video #1 (b) Translation error CDF: CAN - Video #1



(c) Rotation error CDF: CAN - Video #2 (d) Translation error CDF: CAN - Video #2

Figure 6.15 – The rotation and translation error Cumulative Distribution Functions (CDF) on the CAN dataset - Video #1 (a), (b), and Video #2 (c), (d). Notice the poor scores of all methods for the rotation estimation on this dataset, due to the symmetrical appearance of the food can.



(a) Rotation error CDF: DOOR - Video #1 (b) Translation error CDF: DOOR - Video #1

Figure 6.16 – The rotation and translation error Cumulative Distribution Functions(CDF) on the DOOR dataset - Video #1.

PWP3D using the ground-truth pose at roughly half of the video, but their accuracy over the whole sequence remains outperformed by our method. LINE-2D, on the other hand, often fails matching the templates not only when the contours of the box are occluded, but also because its appearance is constantly changed by objects put inside and outside it. CDF curves for the rotation and the translation errors for all the methods are shown in Figure 6.14.

For the CAN dataset, we use a single part to track the full object. In the first video the silhouette of the can is seldom occluded: LINE-2D and PWP3D achieve similar performances, while the lack of texture and the distractor objects make LSD diverge. In the second video, where occlusions occur more often but the background color is different from the one of the can, LSD-SLAM performs better. On both videos, our method consistently outperforms all other methods. Notice that all methods have a quite bad score in retrieving the rotation on this dataset, probably because of the symmetric shape of the object. Error CDF curves are shown in Figure 6.15.

In the DOOR dataset test, LSD-SLAM fails as soon as the door starts to move. LINE-2D fails very often because of the ambiguous contours present in the scene. Finally, PWP3D immediately looses tracking, while our method manages to track frames across the whole video. This result is somehow surprising, since PWP-3D exploits the appearance of the whole door, while our method just exploits a minimal part of its structure. We only use the CAD model for predicting contours and evaluating the computed poses, as explained in Section 6.4.4. Error CDF curves for this dataset are shown in Figure 6.15. On all datasets, the Kalman Filtering described in Section 6.5 and the DoG normalization described in Section 6.2 entail a significant improvement of the performances.

6.6.9 Runtimes

Our current implementation on an Intel Core i7-4820K desktop with GeForce GTX 780 Ti takes 22 ms for the part detection, plus 30 ms to predict the control points for each detected part. The pose estimation takes about 150 ms. Many optimizations are possible. For example, the control point predictions for each part could be run in parallel.

6.7 Applications and Further Developments

Our part-based method was implemented within the final demonstrator of the EDUSAFE European project described in Section 1.2.1. The implementation of the pipeline described above, operating in the environment shown in Figure 6.1 runs at 5Hz on a laptop equipped Intel i7-4720HQ CPU, Nvidia 980M GPU and 32GB RAM.

We considered different improvements for enhancing the accuracy of the method with respect to the original implementation. Among them:

- We introduced a hard negative mining scheme for the detection network, iteratively running the training, testing the networks over a cross-validation set and replacing 30% of the negative training samples with the tested negatives samples which received the highest score for any of the parts. This procedure marginally improved the resilience of the detector in presence of heavily cluttered scenes as the ones showed in Figure 6.1, without affecting the testing time.

- We investigated the possibility of a multi-scale detection scheme by running the detector over the same image cropped and resized at different scales; finally, this procedure was not kept in the final implementation because of running time constraints, and the detections shown in Figure 6.1 are obtained with a single-scale detector. Of course, a multi-scale detection scheme can be employed without re-training when timing constraints allow it.
- We also experimentally verified that employing higher resolution images enhances the accuracy, especially at the detection stage. Nonetheless, this was un-feasible due to computational constraints.

6.7.1 Articulated Objects

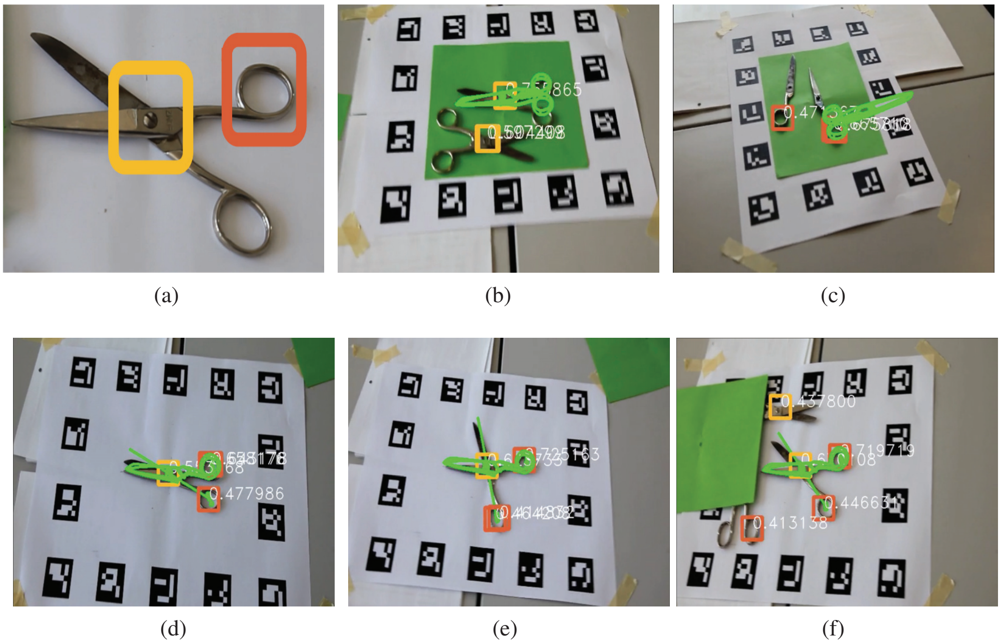
One of the advantages of our part-based framework is given by its flexibility, that makes it suited to track, for example, articulated objects [150] and objects with symmetrical parts.

We tested our pipeline for tracking a pair of scissors based on the 2 parts shown in Figure 6.17-(a), namely the scissors screw and one of the eye rings. This application offers several challenges: besides the fact that the object is highly specular, one of the selected parts, the eye ring, is symmetrical and is very similar to the other eye ring; nonetheless, the 2 eye-rings do not look exactly the same (they are chiral), and distinguishing one from the other is essential for a correct pose estimation.

This test cast new light on the limits and the potentiality of our method. In Figure 6.17-(b), (c) we show the results of the pose estimation employing only a single part. As shown in Figure 6.17-(b), the scissors screw can be reliably detected and the pose estimation is fair. As for the other object part, as shown in Figure 6.17-(c), the detector selects the good eye ring, assigning to the other a positive, but lower score; on the other hand, the pose estimation retrieved for this part is corrupted because of the spherical symmetry of the ring.

Tracking results employing the 2 parts is shown in Figure 6.17-(d), (e), (f). Thanks to the flexibility of the pose representation introduced in Section 6.3, we can still exploit some information about the pose of a symmetric part: more in particular, when tracking based on 2 parts, we keep the predictions for all the control points of the screw and only the 3 control points along the axis of symmetry of the eye ring, that are reliably predicted, while we discard the others. In fact, the projections of the control points lying out of the plane of symmetry are accurately predicted, without the need of re-training the regressor.

Moreover, we can successfully track the articulated blade of the scissors adding a simple 1D research on top of our method: first, we estimate the 3D pose of the first blade with the pipeline described above, discarding a part of the control points for the symmetric eye ring; then, we perform a simple exhaustive search over 25 equi-spaced angles $\psi \in [0^\circ,\ 120^\circ]$, projecting the 3D model of the second blade with an opening of ψ and we check if an eye-ring has been detected



(a) (b) (c)

(d) (e) (f)

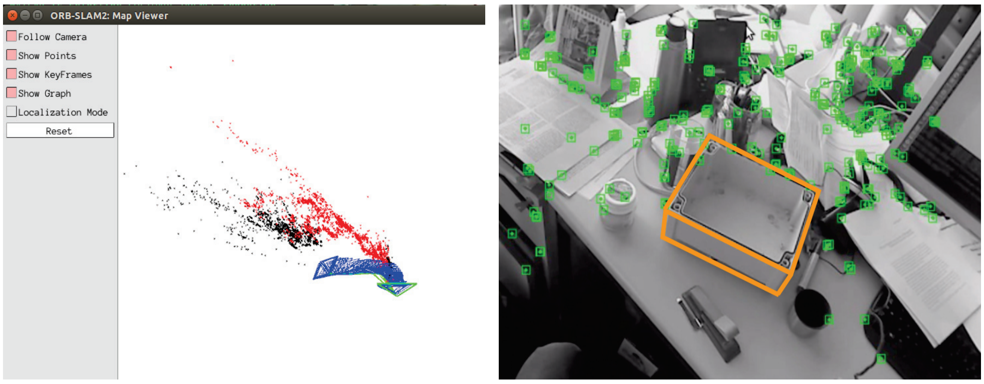
Figure 6.17 – Tracking of articulated objects with our part-based framework. (a) 2 parts are employed for tracking a pair of scissors. (b) Tracking results employing only the scissors screw. Notice how the current implementation of our algorithm returns at most one instance of one object; when only one part is employed, the detection with the highest score is selected. (c) Only the eye ring is employed for the tracking: the pose estimation fails becase of the eye ring symmetry. (d), (e), (f) rigid tracking employing 2 parts and estimation of the configuration of the articulated object, as explained in Section 6.7.1.

close to the predicted location. If a detection was found, the value of ψ is retained. The estimated position of the second blade is drawn as a straight line in Figure 6.17-(d), (e), (f).

6.7.2 Object Tracking and SLAM

As anticipated in Section 1.2, a prominent application of 3D object tracking that we propose is its use in conjunction with localization systems, such as those provided by the most recent Augmented Reality devices like the Microsoft Hololens [6]. In Figure 6.18 we show an example of our system integrated with a state-of-the-art SLAM system, ORB-SLAM [87]. The SLAM system reconstructs the sparse 3D map shown in Figure 6.18-(a) and tracks the camera motion in real time; nonetheless, it does not perform any object recognition.

Our tracking pipeline, running in parallel, computes the object 3D pose in the camera reference system, which is then employed for computing the Euclidean motion between the SLAM reference system and the object reference system. Consecutive estimations of the Euclidean motion can be averaged, or updated in a filtering framework such as the Kalman Filter described in Section 6.5.1. Notice that, once the pose of the object is estimated correctly and if the object is not moving, we no longer need to explicitly detect the object, instead, we can rely on the pose computed by the



(a) (b)

Figure 6.18 – Integration of our object pipeline and a ORB-SLAM, as described in Section 6.7.2. (a) The SLAM reconstructs a sparse 3D map of the environment and tracks the camera in real time. (b) We compute the pose of a pre-defined target object in the SLAM map with our part-based pipeline.

SLAM system, which makes the application extremely robust.

6.8 Conclusion

In this chapter, we introduced a 3D pose estimation pipeline, complementary to the one described in Chapter 5: more in particular, instead of relying on the global appearance of the image for improving resilience to local artifacts, it tracks small, stable parts of the target object leaving the rest of the scene free to vary. The resulting method is thus extremely robust to occlusions and clutter. Poses of each part are accurately estimated by a non-linear regressor and combined thanks to the pose representation introduced in Section 6.3.1.

Our method has been demonstrated with both quantitative tests and a real-life application, the EDUSAFE Augmented Reality Demonstrator at CERN, described in Section 1.2.1.

Although this framework can be considered a model-based method, the amount of object data required is extremely reduced: besides training images showing the parts under different poses and illumination conditions, all we need is to know the 3D position of the parts in the same reference system, referred as the "object" reference system. If a full 3D model of the object is not available, a simple approximation can be employed, such as a parallelepiped as a model of an electric box, for rendering the object silhouette or a part of it under different poses. This makes the extension of our tracking method to categories of objects rather than single instances a particularly appealing and natural research direction, especially for those categories that are defined by specific sets of parts, such as wheelsfor cars, handles for doors, windows for buildings, and so on.

In chapter 7, we further discuss about the main directions of our future research and some possibilities for further improvements.

7

Conclusion

In this thesis we introduced two methods tackling the challenge of reliable 3D rigid object tracking in industrial environments, based on complementary paradigms.

The first method exploits an iterative image alignment framework and a robust dense descriptor for enhancing resilience to locally ambiguous patterns, occlusions, poorly textured objects and local illumination artifacts, that we refer to as "Descriptor Fields". Our Descriptor Fields combine the discriminative power of dense gradient-based convolutional features and a sparsifying non-linear mapping for increasing the basin of attraction of classical iterative alignment methods. Our experimental results show a substantial gain in terms of robustness and convergence rate, for different optimization schemes and metrics. Their reduced computational cost makes them suited for replacing luminous intensity in numerous tracking applications.

More generally, our results show how the image intensities, the *de facto* standard dense descriptor employed in a wide number of state-of-the-art tracking methods, is far from being an optimal choice, and wide performance gains can be obtained employing other descriptors, with little implementation effort and computational overhead.

Although our dense descriptor sensibly enhances the robustness of dense image alignment frameworks to local artifacts, their effectiveness is still limited in presence of highly occluded objects. The second method we introduce bridges this gap, allowing to track objects undergoing extreme occlusions in cluttered environments. It relies on the detection and pose estimation for stable parts of the target object, leaving the rest of the scene and the object to freely vary without affecting the quality of tracking. The pose of each part of the object is reliably predicted by a non-linear regressor, also in presence of challenging illumination conditions and other local artifacts. A novel representation for the pose of the parts, based on the 2D reprojections of a small number of control points, allows to accurately predict and combine the poses of single parts for estimating the pose of the full target object.

All the proposed methods are suited for real-time applications and time-critical tasks, requiring only moderate computational resources. They have been designed for use in real-life scenarios, with extreme attention to practical aspects such as limiting the number and the influence of parameters, or avoiding cumbersome learning steps requiring manual image labelling.

7.1 Future Work

Despite achieving state-of-the-art performance in real-life scenarios, our proposed techniques still present several aspects in which they could be further improved.

Enhanced Descriptor Fields-based tracking: As for our approach based on dense image alignment, we can identify 2 main research directions: the first consists in investigating the design of more efficient, more robust dense descriptors: in principle, the sparsifying operation of (5.9) could be applied to any densely sampled real descriptor, boosting the performances of alignment. The main challenge here, lies in finding the appropriate balance between a reasonable computational burden and improved robustness. Another promising research direction lies in reducing the amount of offline computed data, extending our approach for online retrieval of geometric data and key-frames in a way similar to SLAM systems.

Optimized architectures for object detection and pose prediction: As for the part-based tracking framework, our implementation is currently based on 2 separate CNNs for the detection of the parts and for the regression of their pose. While this has multiple advantages, such as keeping the learning steps simpler and allowing for seamless substitution of the detection technique when it is needed, employing a single CNN for predicting at once the likelihood of the presence of a part on an image patch and its pose would be beneficial for reducing the computational cost of the algorithm.

Category-level pose estimation: Another promising research direction lies in pushing the limit of the variability allowed for the appearance of the tracked parts, for generalizing our part-based tracker from object instances to object categories. Although this seems feasible, it has never been directly experimented.

A unified tracking framework: More in general, our research confirmed that different tracking paradigms are more or less adapted in different situations. A very interesting and natural research direction, then, is how to combine different paradigms. Our early attempts to refine the pose issued by the part-based approach with a dense image alignment method did not success, due to the extreme level of occlusions and clutter present in the considered scenes. A deeper investigation in this direction, extending the tracking to take into account other cues such as local features, could bring to a unified, general framework.

Object tracking vs SLAM: Finally, another related research direction is dictated by the impressive advances in the field of localization and mapping techniques, thanks to the recent introduction of extremely powerful SLAM systems [86, 87] and of commodity devices for Augmented Reality applications [6] that offer localization routines based on different sensing technologies. It is our intention to push forward the efficient integration of our object recognition and tracking algorithms and these localization tools, as described in Section 6.7.2. This would enable new, exciting possibilities for Augmented Reality applications in a wide number of domains.

A

Appendix A: Additional Results for Dense Image Alignment Methods

In this Appendix we report the proof of some of the results claimed in Chapter 4. After giving a practical example of a simple warp in order to better illustrate the differences among the different methods in Appendix A.1, in Appendix A.2 we report the proof of the equivalence of the first-order methods described in Chapter 4. Then, in Appendix A.4 we demonstrate that if hypothesis of Equation (4.39) does not hold, as it is the case for many commonly employed warps, then the ESM method becomes a first order method. Finally, some comparative tables are shown in Appendix A.6 for the reader's convenience.

A.1 An Example of Warp: 2D Rigid Deformation

In order to illustrate the differences among the alignment methods introduced in Chapter 4, we show here a practical example of warp, a rigid 2D deformation. In Section 4.6 we describe a more complex warp epmloyed for 3D pose estimation. Other examples (affine warps, homographies) are reported in [7].

Let $\mathcal{F}$ be the family of rigid transforms of the plane parametrized by a vector of 3 parameters $\mathbf{p} \in \mathbb{R}^3$. At iteration c, given the current parameters $\mathbf{p}_c = (\theta, t_1, t_2)^\top$, the warp for a pixel $\mathbf{x}$ is computed as:

$$\mathbf{W}(\mathbf{x}, \mathbf{p}_c) = \mathbf{R}_{\mathbf{p}_c}\mathbf{x} + \mathbf{t}_{\mathbf{p}_c} = \begin{bmatrix} \cos(\theta) & -sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}. \tag{A.1}$$

the family of warps is closed with respect to the composition and to the inversion. The warps are differentiable with respect to all the arguments; the derivative with respect to the parameters is given by:

$$\frac{\partial}{\partial \mathbf{p}}\mathbf{W}(\mathbf{x},\mathbf{p}) = \begin{bmatrix} -\sin(\theta)u + \cos(\theta)v & 1 & 0 \\ -\cos(\theta)u - \sin(\theta)v & 0 & 1 \end{bmatrix}, \tag{A.2}$$

while $\frac{\partial}{\partial \mathbf{x}}\mathbf{W}(\mathbf{x},\mathbf{p}) = \mathbf{R}_{\mathbf{p}}$. Finally, we note that $\mathbf{W}(\mathbf{x},\mathbf{0}) = \mathbf{x}$.

Forward Additive Algorithm: The assumptions for FA algorithm hold. Once the parameters increment $\delta\mathbf{p} = (\delta\theta, \delta t_1, \delta t_2)$ has been computed, the updated warp is given by:

$$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1}) = \begin{bmatrix} \cos(\theta+\delta\theta) & -\sin(\theta+\delta\theta) \\ \sin(\theta+\delta\theta) & \cos(\theta+\delta\theta) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} t_1+\delta t_1 \\ t_2+\delta t_2 \end{bmatrix}, \tag{A.3}$$

Forward Compositional Algorithm: Assumptions of FC algorithm are satisfied. Once the parameters increment $\delta\mathbf{p} = (\delta\theta, \delta t_1, \delta t_2)$ has been computed, the updated warp is given by:

$$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1}) = \mathbf{R}_{\mathbf{p}_c}(\mathbf{R}_{\delta\mathbf{p}}\mathbf{x} + \mathbf{t}_{\delta\mathbf{p}}) + \mathbf{t}_{\mathbf{p}_c}; \tag{A.4}$$

After computing the updated rotation matrix $\mathbf{R}_{\mathbf{p}_{c+1}} = \mathbf{R}_{\mathbf{p}_c}\mathbf{R}_{\delta\mathbf{p}}$ and and the updated translation vector $\mathbf{t}_{\mathbf{p}_{c+1}} = \mathbf{R}_{\mathbf{p}_c}\mathbf{t}_{\delta\mathbf{p}} + \mathbf{t}_{\mathbf{p}_c}$, we can explicitly estimate the updated parameters as, for example:

$$\mathbf{p}_{c+1} = \begin{bmatrix} \cos^{-1}((\mathbf{R}_{\mathbf{p}_{c+1}})_{11}) \\ \mathbf{t}_{\mathbf{p}_{c+1}} \end{bmatrix}, \tag{A.5}$$

Inverse Compositional Algorithm: the inverse of a rigid warp is a rigid warp; the inverse warp is given by:

$$\mathbf{W}^{-1}(\mathbf{x},\mathbf{p}) = \mathbf{R}_{\mathbf{p}}^{T}(\mathbf{x} - \mathbf{t}_{\mathbf{p}}). \tag{A.6}$$

At iteration c, once the parameters increment $\delta\mathbf{p} = (\delta\theta, \delta t_1, \delta t_2)$ has been computed, the updated warp is given by:

$$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1}) = \mathbf{R}_{\mathbf{p}_c}(\mathbf{R}_{\delta\mathbf{p}}^{\top}(\mathbf{x} - \mathbf{t}_{\delta\mathbf{p}})) + \mathbf{t}_{\mathbf{p}_c}, \tag{A.7}$$

and we can explicitly update the parameters in an analogous way as the FC algorithm.

Inverse Additive Algorithm: In order to apply Inverse Additive algorithm, we have to explicitly compute decomposition of Equation (4.28). Since:

$$\left(\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p}_c)}{\partial \mathbf{x}}\right)^{-1} \left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c} = \begin{bmatrix} \cos(\theta) & sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} -\sin(\theta)u + \cos(\theta)v & 1 & 0 \\ -\cos(\theta)u - \sin(\theta)v & 0 & 1 \end{bmatrix} \tag{A.8}$$

it is not possible to decompose the above Equation in the product of 2 matrices $\Gamma(\mathbf{x})\Sigma(\mathbf{p})$, and thus the Inverse Additive algorithm can not be employed, even with this very simple warp.

Efficient Second-order Method: As can be observed comparing the warp updated with the additional rule (Equation (A.3)) and that updated with the compositional rule (Equation (A.4)), assumption (4.39) does not hold in this case, so ESM does not minimize a second-order approximation of the objective function. Nonetheless, as described in Section A.4, since the warps are differentiable and form a group, ESM will provide a first-order method.

A.2 Equivalence of First-order Methods

The equivalence of the 4 first-order methods described in Chapter 4 (FA, FC, IC and IA) has been shown in [7], in the sense that, at a given iteration, all the algorithms provide the same update for the warp, up to a first order development in $\delta\mathbf{p}$.

We show here only the equivalence of FA and FC algorithms, and that of FC and IC algorithms, therefore showing that the 3 algorithms are all equivalent. Even if the same result holds for IA algorithm, we don't report demonstration here since IA is of little interest for most part of applications (the interested reader may refer to [7] for further details).

A.2.1 Equivalence of FA and FC Algorithms

We can approximate the updated warp sought at each iteration of the FA algorithm with the following first-order development:

$$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1}) = \mathbf{W}(\mathbf{x},\mathbf{p}_c + \delta\mathbf{p}) \approx \mathbf{W}(\mathbf{x},\mathbf{p}_c) + \left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c} \delta\mathbf{p}. \tag{A.9}$$

As for the FC algorithm, an analogous approximation gives:

$$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1}) = \mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c) \approx \mathbf{W}(\mathbf{x},\mathbf{p}_c) + \left.\frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c)}{\partial \mathbf{q}}\right|_{\mathbf{q}=\mathbf{0}} \delta\mathbf{p}. \tag{A.10}$$

So, the values of $\delta\mathbf{p}$ minimizing $F_{FA}(\delta\mathbf{p})$ and $F_{FC}(\delta\mathbf{p})$ are the same (up to the first order) if

$\left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c}$ and $\left.\frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c)}{\partial \mathbf{q}}\right|_{\mathbf{q}=\mathbf{0}}$ share the same linear space, that is, if there is an invertible matrix $A \in \mathbb{R}^{2\times 2}$ such that:

$$\left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c} = A \left.\frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c)}{\partial \mathbf{q}}\right|_{\mathbf{q}=\mathbf{0}}. \tag{A.11}$$

Actually, such matrix exists if the warps are differentiable and closed with respect to inversion and composition (see Appendix A.3 for a proof). Therefore, since the optimal update is sought in the same linear space, the updates of the warps computed by FA and FC at a given iteration are the same up to a first-order development in $\delta\mathbf{p}$.

A.2.2 Equivalence of FC and IC Algorithms

We start by interpreting the sum Equation (4.11) as an integral over the domain of the template $\mathcal{D}$:

$$F_{FC}(\delta\mathbf{p}) = \int_{\mathcal{D}} \Big(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p})) - T(\mathbf{x}) \Big)^2 d\mathbf{x}. \tag{A.12}$$

The change of variables $\mathbf{y} = \mathbf{W}(\mathbf{x},\delta\mathbf{p})$ gives:

$$F_{FC}(\delta\mathbf{p}) = \int_{\mathbf{W}(\mathcal{D},\delta\mathbf{p})} \Big(I(\mathbf{W}(\mathbf{y},\mathbf{p})) - T(\mathbf{W}^{-1}(\mathbf{y},\delta\mathbf{p})) \Big)^2 \left| \frac{\partial \mathbf{W}^{-1}(\mathbf{y},\delta\mathbf{p})}{\partial \mathbf{y}} \right| d\mathbf{y}. \tag{A.13}$$

We observe that $\mathbf{W}(\mathcal{D},\delta\mathbf{p}) \approx \mathcal{D}$ up to a zero-th order, and that

$$\left| \frac{\partial \mathbf{W}^{-1}(\mathbf{y},\delta\mathbf{p})}{\partial \mathbf{y}} \right| = 1 + \mathcal{O}(\delta\mathbf{p}), \tag{A.14}$$

since $\mathbf{W}(\mathbf{x},\mathbf{0}) = \mathbf{x}$. Making the assumption that $(I(\mathbf{W}(\mathbf{y},\mathbf{p})) - T(\mathbf{W}^{-1}(\mathbf{y},\delta\mathbf{p})))$ (or, equivalently, $(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p})) - T(\mathbf{x}))$) is $\mathcal{O}(\delta\mathbf{p})$, we can approximate Equation (A.13) up to the first order ignoring the higher order terms in $\delta\mathbf{p}$:

$$F_{FC}(\delta\mathbf{p}) \approx \int_{\mathcal{D}} \Big(I(\mathbf{W}(\mathbf{y},\mathbf{p})) - T(\mathbf{W}^{-1}(\mathbf{y},\delta\mathbf{p})) \Big)^2 d\mathbf{y}; \tag{A.15}$$

this expression is formally identical to $F_{IC}(\delta\mathbf{p})$ of Equation (4.17) (interpreting the sum as an integral over the domain of the template), except for the inverse warp in the template $T(\mathbf{W}^{-1}(\mathbf{y},\delta\mathbf{p}))$. Since in the IC update rule of Equation (4.20) the warp of the template is inverted before composing it with the current warp, we conclude that the updated warps computed by FC and IC are equivalent up to a first order development in $\delta\mathbf{p}$.

A.3 A Theorem about Groups of Differentiable Warps

We prove here the following theorem, needed for demonstrating the equivalence of FA and FC algorithms in Appendix A:

Theorem 1. *Let $\mathcal{F}$ be a group of differentiable warps $\mathbf{W} : \mathbb{R}^2 \times \mathbb{R}^n \to \mathbb{R}^2$. Then, for any pixel $\mathbf{x} \in \mathbb{R}^2$, an invertible matrix $A \in \mathbb{R}^{2\times 2}$ exists (eventually depending on $\mathbf{x}$), such that:*

$$\left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c} = A(\mathbf{x}) \left.\frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c)}{\partial \mathbf{q}}\right|_{\mathbf{q}=\mathbf{0}}. \tag{A.16}$$

In order to prove Theorem 1, we make use of the following:

Theorem 2. *Let $\mathbf{x} \in \mathbb{R}^2$ some fixed pixel, and $\mathcal{F}$ be a group of differentiable warps $\mathbf{W} : \mathbb{R}^2 \times \mathbb{R}^n \to \mathbb{R}^2$. Then a function $\phi : \mathbb{R}^n \to \mathbb{R}^n : \quad \delta\mathbf{p} \mapsto \delta\mathbf{p}' = \phi(\delta\mathbf{p})$ can be defined in some open ball aroud the origin $\mathcal{B}_\delta(\mathbf{0})$, such that:*

- $\phi(\mathbf{0}) = \mathbf{0}$;
- *ϕ is differentiable and invertible in $\mathcal{B}_\delta(\mathbf{0})$;*
- $\mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}) = \mathbf{W}(\mathbf{W}(\mathbf{x},\phi(\delta\mathbf{p})),\mathbf{p}) \qquad \forall \mathbf{p} \in \mathbb{R}^n$;

Proof. First, we observe that there is a $\epsilon > 0$ such that, for all $\delta\mathbf{p} \in \mathbb{R}^n,\ \delta\mathbf{p} \in \mathcal{B}_\epsilon(\mathbf{0})$, there is $\delta\mathbf{p}' \in \mathbb{R}^n$ such that:

$$\mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}) = \mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}'),\mathbf{p}) \qquad \forall \mathbf{p} \in \mathbb{R}^n; \tag{A.17}$$

Since $\mathcal{F}$ is closed under inversion and composition, $\mathbf{W}^{-1}(\mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}),\mathbf{p}) \in \mathcal{F}$, so there must be some $\delta\mathbf{p}'$ so that $\mathbf{W}(\mathbf{x},\delta\mathbf{p}') := \mathbf{W}^{-1}(\mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}),\mathbf{p})$.

Then, we observe that,under the same assumptions, the inverse statement is also true, that is, there is a $\tilde{\epsilon} > 0$ such that, for all $\delta\mathbf{p} \in \mathbb{R}^n,\ ||\delta\mathbf{p}|| < \tilde{\epsilon}$, there is $\delta\mathbf{p}' \in \mathbb{R}^n$ such that:

$$\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}) = \mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}') \quad \forall \mathbf{p} \in \mathbb{R}^n. \tag{A.18}$$

Applying the Generalized Implicit Function Theorem ([151]) to the continuously differentiable function $F(\delta\mathbf{p},\delta\mathbf{p}') = \mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}) - \mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}')$, we deduce that a function $\phi : \mathbb{R}^n \mapsto \mathbb{R}^n : \quad \delta\mathbf{p} \mapsto \phi(\delta\mathbf{p}) = \delta\mathbf{p}'$ exists, which is differentiable and invertible in some open ball around $\mathbf{0}$ and such that $\phi(\mathbf{0}) = \mathbf{0}$. □

Now, we notice that, given a $\mathbf{p},\mathbf{q},\mathbf{r} \in \mathbb{R}^n$, we have :

$$\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p}+\mathbf{q})}{\partial \mathbf{p}} = \frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p}+\mathbf{q})}{\partial \mathbf{q}}; \tag{A.19}$$

and:

$$\frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x}, \mathbf{r}+\mathbf{q}), \mathbf{p})}{\partial \mathbf{r}} = \frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x}, \mathbf{r}+\mathbf{q}), \mathbf{p})}{\partial \mathbf{q}}. \tag{A.20}$$

Now, let $\delta\mathbf{p} \in \mathbb{R}^n$ small enough, so that we can define a function $\phi(\delta\mathbf{p})$ as in Theorem 2. We have:

$$\left.\frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p}+\delta\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c} = \frac{\partial \mathbf{W}(\mathbf{x}, \mathbf{p}_c+\delta\mathbf{p})}{\partial \delta\mathbf{p}} = \tag{A.21}$$

$$\frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}'), \mathbf{p}_c)}{\partial \delta\mathbf{p}'} \frac{\partial \delta\mathbf{p}'}{\partial \delta\mathbf{p}} \approx \nabla\phi(\delta\mathbf{p}) \left.\frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x}, \mathbf{q}+\delta\mathbf{p}'), \mathbf{p}_c)}{\partial \mathbf{q}}\right|_{\mathbf{q}=\mathbf{0}}. \tag{A.22}$$

Finally, the statement of Theorem 1 is obtained by evaluating the above expression for $\delta\mathbf{p} = \mathbf{0}$. So, the invertible matrix A of Equation (A.16) is given by $\nabla\phi(\mathbf{0})$: this shows that $\left.\frac{\partial \mathbf{W}(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c}$ and $\left.\frac{\partial \mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c)}{\partial \mathbf{q}}\right|_{\mathbf{q}=\mathbf{0}}$ share the same linear space, the tangent space of the manifold $\mathbf{W}(\mathbf{x}, \mathbf{p}_c)$.

A.4 Relaxing Hypothesis of ESM

In order to apply ESM to a family of warps $\mathcal{F}$, the cumbersome hypothesis of Equation (4.39) must hold for all the warps in $\mathcal{F}$:

$$\begin{aligned} &\exists \epsilon > 0 \text{ such that } \forall \delta\mathbf{p} \in \mathbb{R}^n, ||\delta\mathbf{p}|| < \epsilon, \text{ then:} \\ &\mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p}) = \mathbf{W}(\mathbf{x}, \mathbf{p}+\delta\mathbf{p}) \qquad \forall \mathbf{p} \in \mathbb{R}^n; \end{aligned}$$

This seriously limits the practical applications of ESM, since only a restricted set of warps respects this assumption. In this section we show that, if this hypothesis does not hold, but $\mathcal{F}$ is a group of differentiable warps, then ESM looses the second-order accuracy and becomes first-order method as FC.

As shown in Section 4.4, ESM is built computing a second-order development of the objective function (Equation (4.36)) and replacing the second-order term of this expression with an approximation based on the first-order development of the image jacobian $\mathbf{J}_{FC}$ of Equation (4.40). If assumption (4.39) does not hold, the development of Equation (4.40) is no longer valid; however, it is possible to employ Theorem 2 for a quantitative estimate of the error done in ESM, showing that ESM objective function approximation is still accurate up to the first order.

Let $\mathcal{F}$ a group of differentiable warps and $\delta\mathbf{p} \in \mathbb{R}^n$ such that $||\delta\mathbf{p}||$ is small enough. Then, thanks to Theorem 2, there's $\delta\mathbf{p}' = \phi^{-1}(\delta\mathbf{p})$ such that:

$$\mathbf{W}(\mathbf{x}, \mathbf{p}+\delta\mathbf{p}') = \mathbf{W}(\mathbf{W}(\mathbf{x}, \delta\mathbf{p}), \mathbf{p}) \qquad \forall \mathbf{p} \in \mathbb{R}^n; \tag{A.23}$$

So, the development of Equation (4.40) becomes:

$$
\begin{aligned}
T(\mathbf{x}) \;&=\; I(\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c)) \quad \Leftrightarrow \\
T(\mathbf{x}) \;&=\; I(\mathbf{W}(\mathbf{x},\mathbf{p}_c+\delta\mathbf{p}')) \quad \Leftrightarrow \\
T(\mathbf{W}(\mathbf{x},\mathbf{0})) \;&=\; I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{0}),\mathbf{p}_c+\delta\mathbf{p}')) \quad \Leftrightarrow \\
\left[\nabla T\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{0})}{\partial\mathbf{p}}\right] \;&=\; \frac{\partial}{\partial\mathbf{q}}\Big(T(\mathbf{W}(\mathbf{x},\mathbf{q}))\Big)\Big|_{\mathbf{q}=\mathbf{0}} = \frac{\partial}{\partial\mathbf{q}}\Big(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c+\delta\mathbf{p}')))\Big|_{\mathbf{q}=\mathbf{0}} \quad \Leftrightarrow \\
\left[\nabla T\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{0})}{\partial\mathbf{p}}\right] \;&\approx\; \frac{\partial}{\partial\mathbf{q}}\Big(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p}_c))\Big)\Big|_{\mathbf{q}=\mathbf{0}} + \frac{\partial}{\partial\mathbf{p}}\frac{\partial}{\partial\mathbf{q}}\Big(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\mathbf{q}),\mathbf{p})\Big)\Big|_{\mathbf{p}=\mathbf{p}_c,\ \mathbf{q}=\mathbf{0}}\delta\mathbf{p}' \quad \Leftrightarrow \\
\left[\nabla T\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{0})}{\partial\mathbf{p}}\right] \;&\approx\; \mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c)+\mathbf{M}\delta\mathbf{p}' \Leftrightarrow \\
\left[\nabla T\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{0})}{\partial\mathbf{p}}\right] \;&\approx\; \mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c)+\mathbf{M}\nabla\phi^{-1}(\mathbf{0})\delta\mathbf{p}.
\end{aligned}
$$

That is, using the approximation $\mathbf{M}\delta\mathbf{p} \approx \left[\nabla T\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{0})}{\partial\mathbf{p}}\right] - \mathbf{J}_{FC}(\mathbf{x},\mathbf{p}_c)$ in Equation (4.36), we introduce an error:

$$\mathbf{e} = (\nabla\phi^{-1}(\mathbf{0}) - \mathbb{I}))\delta\mathbf{p} \tag{A.24}$$

in the second-order term, so that the approximation of the objective function is correct only up to the first order. If Assumption (4.39) holds, then ϕ is the identity function and the error is null.

A.5 Alignment of Multi-Channel Images

For sake of simplicity, in all the iterative methods described in Chapter 4 the images $T,\ I$ are considered to be mono-channel, for instance grayscale images. When appropriate, the same methods can be straightforwardly applied to multi-channel images, extending all the sums to all the channels of the image.

SSD Distance Let $\mathbf{L}$, $\mathbf{T}$ be two images with K channels[1], respectively $L_{k,k=1,\ldots,K}$ and $T_{k,k=1,\ldots,K}$: the minimization of Equation (4.4) becomes:

$$\mathbf{p}_L = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \Big(\mathbf{L}(\mathbf{W}(\mathbf{x},\mathbf{p})) - \mathbf{T}(\mathbf{x})\Big)^\top \Big(\mathbf{L}(\mathbf{W}(\mathbf{x},\mathbf{p})) - \mathbf{T}(\mathbf{x})\Big), \tag{A.25}$$

where $\mathbf{L}(\cdot) = [L_1(\cdot), \ldots, L_K(\cdot)]^\top$ and $\mathbf{T}(\cdot) = [T_1(\cdot), \ldots, T_K(\cdot)]^\top$, are $K \times 1$ arrays storing all the channel values for a pixel. At implementation level, this can easily be achieved for images of size $W \times H$ with K channels, for example creating tiled mono-channel images of size $KW \times H$, where each tile of size $W \times H$ contains the values for the corresponding channel, and conveniently adjusting the range of the sum over the pixels. Of course some care should be taken, in order to discard the spurious derivatives created at the borders between consecutive tiles.

Mahalanobis Distance A further variant of Equation (A.25) is given by the use of Mahalanobis distance; in this case the optimization problem becomes:

$$\mathbf{p}_L = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \Big(\mathbf{L}(\mathbf{W}(\mathbf{x},\mathbf{p})) - \mathbf{T}(\mathbf{x})\Big)^\top \mathbf{Z} \Big(\mathbf{L}(\mathbf{W}(\mathbf{x},\mathbf{p})) - \mathbf{T}(\mathbf{x})\Big), \tag{A.26}$$

where $\mathbf{Z}$ is a $K \times K$, positive definite matrix $\mathbf{Z}$, usually taken as the inverse of the covariance matrix of the images channels.

This case can be implemented similarly to the case of the SSD distance: in fact, since $\mathbf{Z}$ is symmetric and positive definite, a matrix $\mathbf{U}$ exists, such that $\mathbf{Z} = \mathbf{U}^\top\mathbf{U}$. So, the above problem is equivalent to minimizing the following SSD problem:

$$\mathbf{p}_L = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \Big(\tilde{\mathbf{L}}(\mathbf{W}(\mathbf{x},\mathbf{p})) - \tilde{\mathbf{T}}(\mathbf{x})\Big)^\top \Big(\tilde{\mathbf{L}}(\mathbf{W}(\mathbf{x},\mathbf{p})) - \tilde{\mathbf{T}}(\mathbf{x})\Big), \tag{A.27}$$

where $\tilde{\mathbf{L}} = \mathbf{UL}$ and $\tilde{\mathbf{T}} = \mathbf{UT}$. This choice is also computationally efficient since $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{T}}$ can be precomputed, so that the computational cost for one iteration is the same as for problem (A.25).

A.6 Comparative Tables of Dense Alignment Methods

Table A.1 resumes all the algorithms described in Chapter 4, along with their computational complexity, the order of approximation of the objective function and the hypothesis made on the family of parametrized warps.

Table A.2 shows the formulas employed by each algorithm. All the algorithms seek for a parameters increment $\delta\mathbf{p}$, approximately minimizing an objective function $F(\delta\mathbf{p})$. The parameters increment

---

[1] We denote the first multi-channel image $\mathbf{L}$ for avoiding confusion with the identity matrix $\mathbf{I}$ previously employed.

Algo	Order of approx.	Computational Complexity	Assumptions on $\mathcal{F}$
FA	I	$\mathcal{O}(n^2N + n^3)$	$\mathbf{W}(\mathbf{x}, \mathbf{p})$ differentiable wrt $\mathbf{p}$
FC	I	$\mathcal{O}(n^2N + n^3)$	$\mathbf{W}(\mathbf{x}, \mathbf{p})$ differentiable. $\mathcal{F}$ is a semi-group
IC	I	$\mathcal{O}(n^2N + n^2)$	$\mathbf{W}(\mathbf{x}, \mathbf{p})$ differentiable. $\mathcal{F}$ is a group
IA	I	$\mathcal{O}(n^2N + n^2)$	$\mathbf{W}(\mathbf{x}, \mathbf{p})$ differentiable. Decomposition of Equation (4.28)
ESM	II	$\mathcal{O}(n^2N + n^3)$	$\mathbf{W}(\mathbf{x}, \mathbf{p})$ differentiable. $\mathcal{F}$ is a semi-group. Hyp. of Equation (4.39)

Table A.1 – Computational complexity and assumptions on the family of warps for the algorithms described in Chapter 4. The computational complexity for one iteration of each algorithm is given, as a function of the number N of pixels of the template and the number n of parameters of the warps. For common applications, $N \in [10^3, 10^5]$ and $n < 10$.

$\delta\mathbf{p}$ is computed by all the methods with formula of Equation (4.53):

$$\delta\mathbf{p} = \alpha\, \mathbf{H}^{-1} \sum_{\mathbf{x}} \mathbf{J}(\mathbf{x})^\top \, (T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}, \mathbf{p}_c))), \tag{A.28}$$

where $\mathbf{H} = \sum_{\mathbf{x}}(\mathbf{J}(\mathbf{x})^\top\, \mathbf{J}(\mathbf{x}))$, and

$$\alpha = \begin{cases} 1 & \text{for forward algorithms (FA, FC, ESM)} \\ -1 & \text{for inverse algorithms (IA, IC).} \end{cases}$$

Algo	**Objective Function** $F(\delta\mathbf{p})$	**Approx. order**	**J**	**Update**
FA	$\sum_{\mathbf{x}}(I(\mathbf{W}(\mathbf{x},\mathbf{p}_c+\delta\mathbf{p}))-T(\mathbf{x}))^2$	I	$\mathbf{J}_{FA}=\nabla I(\mathbf{W}(\mathbf{x},\mathbf{p}_c))\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{p}_c)}{\partial\mathbf{p}}$	$\mathbf{p}_{c+1}=\mathbf{p}_c+\delta\mathbf{p}$
FC	$\sum_{\mathbf{x}}(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c))-T(\mathbf{x}))^2$	I	$\mathbf{J}_{FC}=\nabla I(\mathbf{W}(\mathbf{x},\mathbf{p}_c))\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{p}_c)}{\partial\mathbf{x}}\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{0})}{\partial\mathbf{p}}$	$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1})=\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c)$
IC	$\sum_{\mathbf{x}}(T(\mathbf{W}(\mathbf{x},\delta\mathbf{p}))-I(\mathbf{W}(\mathbf{x},\mathbf{p}_c)))^2$	I	$\mathbf{J}_{IC}=\nabla T(\mathbf{x})\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{0})}{\partial\mathbf{p}}$	$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1})=\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p})^{-1},\mathbf{p}_c).$
IA	$\sum_{\mathbf{x}}(I(\mathbf{W}(\mathbf{x},\mathbf{p}_c+\delta\mathbf{p}))-T(\mathbf{x}))^2$	I	$\left[\nabla I\frac{\partial\mathbf{W}(\mathbf{x},\mathbf{p})}{\partial\mathbf{p}}\right]$	$\mathbf{p}_{c+1}=\mathbf{p}_c-\delta\mathbf{p}$
ESM	$\sum_{\mathbf{x}}(I(\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c))-T(\mathbf{x}))^2$	II	$\mathbf{J}_{ESM}=\frac{\mathbf{J}_{FC}+\mathbf{J}_{IC}}{2}$	$\mathbf{W}(\mathbf{x},\mathbf{p}_{c+1})=\mathbf{W}(\mathbf{W}(\mathbf{x},\delta\mathbf{p}),\mathbf{p}_c)$

Table A.2 – Formulas for the update of the warp estimate for the algorithms described in this chapter. The sums are extended to all pixels $\mathbf{x}$ of the template, while $\mathbf{p}_c$ is the current parameters estimate at iteration c.

B

Appendix B: 3D Tracking with Dense Image Alignment and Different Internal Matrices

A fundamental hypothesis for employing the iterative algorithms introduced in Chapter 4, more in particular the FC, the IC and the ESM algorithms, is that $\mathbf{W}(\mathbf{x}, \mathbf{0}) = \mathbf{x}$. When we use these methods for 3D pose estimation as described in Chapter 5, employing the warp introduced in Section 4.6, this is true only if the image and the template have the same internal calibration matrix. [1] If T and I have different internal calibration matrices (say, respectively, $\mathbf{K}_T$ and $\mathbf{K}_{IM}$), one can pre-warp the image and use $\tilde{I}(\mathbf{x}) = I(\mathbf{K}_T \mathbf{K}_{IM}^{-1} \mathbf{x})$. Alternatively, as shown in Figure B.1, it is possible to split the global warp in 2 parts, the warp $\mathbf{W}(\mathbf{x}, \mathbf{p})$ estimated through image alignment, that employs exclusively the template internal calibration matrix $\mathbf{K}_T$ (in the red box in Figure B.1), and an additional transformation $\widehat{\mathbf{W}(\mathbf{x}, \mathbf{p})} = \mathbf{K}_{IM} \mathbf{K}_T^{-1} \mathbf{W}(\mathbf{x}, \mathbf{p})$ for reading intensity values on the image. From the implementation point of view, this means there is no need of pre-warping the image, but only:

1. employ $\mathbf{K}_T$ to compute the warp derivatives and the jacobian matrices of the 3D warp;
2. use $I(\widehat{\mathbf{W}(\mathbf{x}, \mathbf{p})})$ instead of $I(\mathbf{W}(\mathbf{x}, \mathbf{p}))$ for retrieving the luminous intensity values of the image pixels. Notice that

$$I(\mathbf{K}_{IM} \mathbf{K}_T^{-1} \mathcal{P}_{\mathbf{K}_T}(\mathbf{X}, \mathbf{p}_T + \mathbf{p})) = I(\mathcal{P}_{\mathbf{K}_{IM}}(\mathbf{X}, \mathbf{p}_T + \mathbf{p})). \tag{B.1}$$

---

[1] For sake of simplicity, in this section we employ a slight abuse of notation employing the same notation for 2D arrays such as $\mathbf{x}$ or $\mathbf{W}(\mathbf{x}, \mathbf{p})$ and the corresponding quantities in homogeneous coordinates, that is 3D arrays obtained appending 1 to the corresponding 2D quantity, for instance $\tilde{\mathbf{x}} = [\mathbf{x}^\top \; 1]^\top$, $\tilde{\mathbf{W}}(\mathbf{x}, \mathbf{p}) = [\mathbf{W}(\mathbf{x}, \mathbf{p})^\top \; 1]^\top$, since this does not lead to confusion. We refer to [94] for an exhaustive description about the use of homogeneous coordinates and their properties.

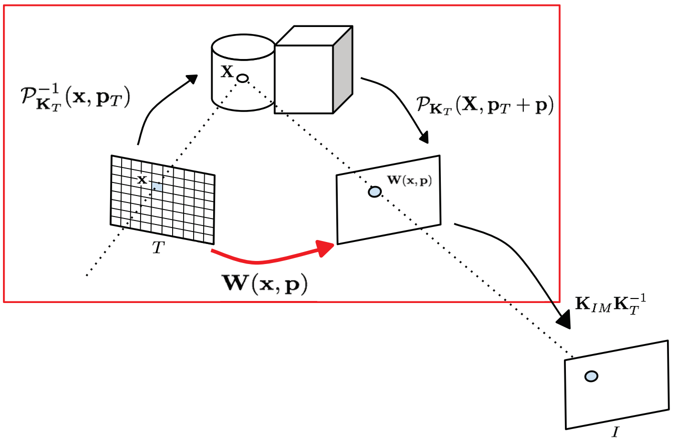

$\mathcal{P}^{-1}_{\mathbf{K}_T}(\mathbf{x},\mathbf{p}_T)$
$\mathbf{X}$
$\mathcal{P}_{\mathbf{K}_T}(\mathbf{X},\mathbf{p}_T+\mathbf{p})$
$\mathbf{x}$
$\mathbf{W}(\mathbf{x},\mathbf{p})$
T
$\mathbf{W}(\mathbf{x},\mathbf{p})$
$\mathbf{K}_{IM}\mathbf{K}_T^{-1}$
I

Figure B.1 – Warp between a template T with internal calibration matrix $\mathbf{K}_T$ and pose $\mathbf{p}_T$, and an image I, with pose $\mathbf{p}_T+\mathbf{p}$ and internal calibration matrix $\mathbf{K}_{IM}$. Notice the difference with Figure 4.6. Dependence of the projective transforms on the internal calibration matrix has been highlighted: $\mathcal{P}_{\mathbf{K}}(\mathbf{X},\mathbf{p}) = \mathbf{K}(\mathbf{R}(\mathbf{p})\mathbf{X}+\mathbf{t}(\mathbf{p}))$

Bibliography

[1] D. Demir, S. Birecik, F. Kurugöllü, M. Sezgin, B. Bucak, B. Sankur, and E. Anarim, "Quality inspection in pcbs and smds using computer vision techniques," in *Industrial Electronics, Control and Instrumentation, 1994. IECON'94., 20th International Conference on*, vol. 2. IEEE, 1994, pp. 857–861.

[2] E. Saldaña, R. Siche, M. Luján, and R.Quevedo, "Review: computer vision applied to the inspection and quality control of fruits and vegetables," *Brazilian Journal of Food Technology*, vol. 16, no. 4, pp. 254–272, 2013.

[3] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *CoRR*, vol. abs/1603.02199, 2016. [Online]. Available: http://arxiv.org/abs/1603.02199

[4] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of intelligent and robotic systems*, vol. 53, no. 3, pp. 263–296, 2008.

[5] S. Henderson and S. Feiner, "Augmented reality in the psychomotor phase of a procedural task," in *ISMAR*. IEEE, 2011, pp. 191–200.

[6] "Microsort Hololens," https://www.microsoft.com/microsoft-hololens/en-us, 2006, accessed: 2016-31-07.

[7] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," *IJCV*, pp. 221–255, March 2004.

[8] E. Malis, "Improving Vision-Based Control Using Efficient Second-Order Minimization Techniques," in *ICRA*, 2004, pp. 1843–1848.

[9] A. Crivellaro and V. Lepetit, "Robust 3D Tracking with Descriptor Fields," in *CVPR*, 2014.

[10] A. Crivellaro, M. Rad, Y. Verdie, K. Yi, P. Fua, and V. Lepetit, "A novel representation of parts for accurate 3d object detection and tracking in monocular images," in *ICCV*, 2015.

[11] C. Harris and C. Stennett, "RAPID-a Video Rate Object Tracker," in *BMVC*, 1990.

[12] D. G. Lowe, "Fitting Parameterized Three-Dimensional Models to Images," *PAMI*, vol. 13, no. 5, pp. 441–450, June 1991.

[13] T. Drummond and R. Cipolla, "Real-Time Visual Tracking of Complex Structures," *PAMI*, vol. 27, no. 7, pp. 932–946, July 2002.

[14] G. Klein and D. Murray, "Full-3D Edge Tracking with a Particle Filter," in *BMVC*, 2006.

[15] M. Armstrong and A. Zisserman, "Robust Object Tracking," in *ACCV*, 1995.

[16] L. Vacchetti, V. Lepetit, and P. Fua, "Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking," in *ISMAR*, 2004.

[17] I. Skrypnyk and D. G. Lowe, "Scene Modelling, Recognition and Tracking with Invariant Image Features," in *ISMAR*, November 2004.

[18] L. Vacchetti, V. Lepetit, and P. Fua, "Stable Real-Time 3D Tracking Using Online and Offline Information," *PAMI*, vol. 26, no. 10, October 2004.

[19] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose Tracking from Natural Features on Mobile Phones," in *ISMAR*, September 2008.

[20] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *IJCV*, vol. 20, no. 2, 2004.

[21] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *ECCV*, 2006.

[22] E. Rublee, V. Rabaud, K. Konolidge, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," in *ICCV*, 2011.

[23] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," *PAMI*, vol. 34, no. 7, pp. 1281–1298, 2012.

[24] P. Alcantarilla, P. Fernández, A. Bartoli, and A. J. Davidson, "KAZE Features," in *ECCV*, 2012.

[25] T. Trzcinski, M. Christoudias, and V. Lepetit, "Learning Image Descriptors with Boosting," *PAMI*, vol. 37, no. 3, pp. 597–610, 2015.

[26] A. Collet, M. Martinez, and S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research*, 2011.

[27] E. Rosten and T. Drummond, "Fusing Points and Lines for High Performance Tracking," in *ICCV*, 2005.

[28] C. Choi, A. Trevor, and H. Christensen, "RGB-D Edge Detection and Edge-Based Registration," in *IROS*, 2013.

[29] K. Pauwels, L. Rubio, J. Diaz, and E. Ros, "Real-Time Model-Based Rigid Object Pose Estimation and Tracking Combining Dense and Sparse Visual Cues," in *CVPR*, 2013.

[30] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes," in *ACCV*, 2012.

[31] V. Prisacariu, A. Segal, and I. Reid, “Simultaneous Monocular 2D Segmentation, 3D Pose Recovery and 3D Reconstruction,” in *ACCV*, 2012.

[32] V. Prisacariu and I. Reid, “PWP3D: Real-Time Segmentation and Tracking of 3D Objects,” *IJCV*, vol. 98, pp. 335–354, 2012.

[33] G. Chliveros, M. Pateraki, and P. Trahanias, “Robust Multi-Hypothesis 3D Object Pose Tracking,” in *ICCV*, 2013.

[34] B. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision,” in *IJCAI*, 1981, pp. 674–679.

[35] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active Appearance Models,” *PAMI*, vol. 23, no. 6, June 2001.

[36] F. Jurie and M. Dhome, “Hyperplane Approximation for Template Matching,” *PAMI*, vol. 24, no. 7, pp. 996–100, July 2002.

[37] C. Mei, S. Benhimane, E. Malis, and P. Rives, “Efficient Homography-Based Tracking and 3D Reconstruction for Single-Viewpoint Sensors,” *TRA*, vol. 24, no. 6, pp. 1352–1364, 2008.

[38] S. Lucey, Y. Wang, and J. F. Cohn, “Non-Rigid Face Tracking with Enforced Convexity and Local Appearance Consistency Constraint,” *IJCV*, vol. 28, no. 5, pp. 781–789, 2010.

[39] R. Newcombe, S. Lovegrove, and A. Davison, “DTAM: Dense Tracking and Mapping in Real-Time,” in *ICCV*, 2011.

[40] P. Besl and N. Mckay, “A Method for Registration of 3D Shapes,” *PAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[41] D. Meger, C. Wojek, J. Little, and B. Schiele, “Explicit occlusion reasoning for 3d object detection.” in *BMVC*, 2011.

[42] B. Drost, M. Ulrich, N. Navab, and S. Ilic, “Model Globally, Match Locally: Efficient and Robust 3D Object Recognition,” in *CVPR*, 2010.

[43] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, “SLAM++: Simultaneous Localisation and Mapping at the Level of Objects,” in *CVPR*, 2013.

[44] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, “Gradient Response Maps for Real-Time Detection of Textureless Objects,” *PAMI*, vol. 34, no. 5, pp. 876–888, 2012.

[45] R. Rios-cabrera and T. Tuytelaars, “Discriminatively Trained Templates for 3D Object Detection: A Real Time Scalable Approach,” in *ICCV*, 2013.

[46] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit, “Online Learning of Patch Perspective Rectification for Efficient Object Detection,” in *CVPR*, 2008.

[47] K. Lai, L. Bo, X. Ren, and D. Fox, “A Scalable Tree-Based Approach for Joint Object and Pose Recognition,” in *AAAI*, 2011.

[48] D. Tan and S. Ilic, “Multi-Forest Tracker: A Chameleon in Tracking,” in *CVPR*, 2014.

[49] J. Tan, F. Tombari, S. Ilic, and N. Navab, “A versatile learning-based 3d temporal tracker: Scalable, robust, online,” in *ICCV*, 2015.

[50] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, “Latent-Class Hough Forests for 3D Object Detection and Pose Estimation,” in *ECCV*, 2014.

[51] A. Doumanoglou, V. Balntas, R. Kouskouridas, S. Malassiotis, and T. Kim, “6d object detection and next-best-view prediction in the crowd,” in *CVPR*, 2016.

[52] U. Bonde, V. Badrinarayanan, and R.Cipolla, “Robust instance recognition in presence of occlusion and clutter,” in *ECCV*, 2014.

[53] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, “KinectFusion: Real-Time Dense Surface Mapping and Tracking,” in *ISMAR*, 2011.

[54] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, “Learning 6d object pose estimation using 3d object coordinates,” in *ECCV*, 2014.

[55] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon, “The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation,” in *CVPR*, 2012.

[56] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, “Scene coordinate regression forests for camera relocalization in rgb-d images,” in *CVPR*, 2013, pp. 2930–2937.

[57] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, “Learning Analysis-By-Synthesis for 6D Pose Estimation in RGB-D Images,” in *ICCV*, 2015.

[58] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother, “6-dof model based tracking via object coordinate regression,” in *ACCV*, 2014.

[59] E. Brachmann, F. Michel, A. Krull, M. M. Yang, S. Gumhold, and C. Rother, “Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image,” in *CVPR*, 2016.

[60] A. Kendall, M. Grimes, and R. Cipolla, “Posenet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization,” in *ICCV*, 2015.

[61] P. Wohlhart and V. Lepetit, “Learning Descriptors for Object Recognition and 3D Pose Estimation,” in *CVPR*, 2015.

[62] A. Doumanoglou, V. Balntas, R. Kouskouridas, and T. Kim, “Siamese Regression Networks with Efficient mid-level Feature Extraction for 3D Object Pose Estimation,” *ARXIV*, 2016.

[63] U. Bonde, V. Badrinarayanan, R. Cipolla, and M. Pham, “TemplateNet for Depth-Based Object Instance Recognition,” *ARXIV*, 2015.

[64] J. Koenderink and A. van Doorn, “The internal representation of solid shape with respect to vision,” *Biological cybernetics*, vol. 32, no. 4, pp. 211–216, 1979.

[65] A. Kushal, C. Schmid, and J. Ponce, “Flexible Object Models for Category-Level 3D Object Recognition,” in *CVPR*, 2007.

[66] J. Liebelt and C. Schmid, “Multi-View Object Class Detection with a 3D Geometric Model,” in *CVPR*, 2010.

[67] S. Savarese and L. Fei-fei, “3D Generic Object Categorization, Localization and Pose Estimation,” in *ICCV*, 2007.

[68] H. Su, M. Sun, L. Fei-fei, and S. Savarese, “Learning a Dense Multi-View Representation for Detection, Viewpoint Classification and Synthesis of Object Categories,” in *ICCV*, 2009.

[69] M. Sun, H. Su, S. Savarese, and L. Fei-Fei, “A multi-view probabilistic model for 3d object classes,” in *CVPR*, 2009.

[70] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, “Object Detection with Discriminatively Trained Part Based Models,” *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[71] C. Gu and X. Ren, “Discriminative Mixture-Of-Templates for Viewpoint Classification,” in *ECCV*, 2010.

[72] B. Pepik, M. Stark, P. Gehler, and B. Schiele, “Teaching 3D Geometry to Deformable Part Models,” in *CVPR*, 2012.

[73] A. Shrivastava and A. Gupta, “Building Part-Based Object Detectors via 3D Geometry,” in *ICCV*, 2013.

[74] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese, “Monocular Multiview Object Tracking with 3D Aspect Parts,” in *ECCV*, 2014.

[75] N. Payet and S. Todorovic, “From Contours to 3D Object Detection and Pose Estimation,” in *ICCV*, 2011.

[76] J. Lim, A. Khosla, and A. Torralba, “FPM: Fine Pose Parts-Based Model with 3D CAD Models,” in *ECCV*, 2014.

[77] K. Koser and R. Koch, “Perspectively Invariant Normal Features,” in *ICCV*, 2007.

[78] S. Gupta, P. Arbelaez, R. Girshick, and J. Malik, “Aligning 3d models to rgb-d images of cluttered scenes,” in *CVPR*, 2015.

[79] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, “Learning Rich Features from RGB-D Images for Object Detection and Segmentation,” in *ECCV*, 2014.

[80] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic, “Seeing 3D Chairs: Exemplar Part-Based 2D-3D Alignement Using a Large Dataset of CAD Models,” in *CVPR*, 2014.

[81] H. Su, C. Qi, Y. Li, and L. J. Guibas, “Render for CNN: Viewpoint estimation in images using cnns trained with rendered 3d model views,” in *ICCV*, 2015.

[82] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *CVPR*, 2015.

[83] X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning deep object detectors from 3d models," in *ICCV*, 2015.

[84] F. Massa, B. Russell, and M. Aubry, "Deep exemplar 2d-3d detection by adapting from real to rendered views," in *"CVPR"*, "2016".

[85] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *ISMAR*, 2007.

[86] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *ECCV*, 2014.

[87] R. Mur-Artal, J. Montiel, and J. Tardós, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[88] Y. Xu and A. Roy-Chowdhury, "Inverse Compositional Estimation of 3D Pose and Lighting in Dynamic Scenes," *PAMI*, vol. 30, no. 7, pp. 1300–1307, 2008.

[89] G. Silveira and E. Malis, "Real-Time Visual Tracking Under Arbitrary Illumination Changes," in *CVPR*, 2007.

[90] P. Lagger, M. Salzmann, V. Lepetit, and P. Fua, "3D Pose Refinement from Reflections," in *CVPR*, 2008.

[91] A. Netz and M. Osadchy, "Recognition using specular highlights," *PAMI*, vol. 35, no. 3, pp. 639–652, 2013.

[92] Z. Zia, M. Stark, and K. Schindler, "Explicit occlusion modeling for 3d object class representations," in *CVPR*, 2013.

[93] E. Hsiao and M. Hebert, "Occlusion Reasoning for Object Detection Under Arbitrary Viewpoint," *PAMI*, vol. 36, no. 9, pp. 1803–1815, 2014.

[94] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[95] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2011.

[96] F. Grassia, "Practical Parameterization of Rotations Using the Exponential Map," *Journal of graphics tools*, vol. 3, no. 3, pp. 29–48, 1998.

[97] S. Benhimane and E. Malis, "Homography-Based 2D Visual Tracking and Servoing," *The International Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, 2007.

[98] H.-Y. Shum and R. Szeliski, "Systems and Experiment Paper: Construction of Panoramic Image Mosaics with Global and Local Alignment," *IJCV*, vol. 36, no. 2, pp. 101–130, 2000.

[99] S. Baker and I. Matthews, "Equivalence and Efficiency of Image Alignment Algorithms," in *CVPR*, 2001.

[100] G. Hager and P. Belhumeur, “Efficient Region Tracking with Parametric Models of Geometry and Illumination,” *PAMI*, vol. 20, no. 10, pp. 1025–1039, 1998.

[101] X. Xiong and F. D. la Torre, “Supervised descent method and its applications to face alignment,” in *CVPR*, 2013.

[102] ——, “Global supervised descent method,” in *CVPR*, 2015.

[103] C. Lin, R. Zhu, and S. Lucey, “The conditional lucas & kanade algorithm,” *ARXIV*, 2016.

[104] G. Scandaroli, M. Meilland, and R. Richa, “Improving NCC-Based Direct Visual Tracking,” in *ECCV*, 2012.

[105] A. Dame and E. Marchand, “Second-Order Optimization of Mutual Information for Real-Time Image Registration,” *IEEE Transactions on Image Processing*, vol. 21, no. 9, pp. 4190–4203, 2012.

[106] G. Panin and A. Knoll, “Mutual Information-Based 3D Object Tracking,” *IJCV*, vol. 78, no. 1, pp. 107–118, 2008.

[107] N. Dowson and R. Bowden, “A Unifying Framework for Mutual Information Methods for Use in Non-Linear Optimisation,” in *ECCV*, 2006.

[108] P. Viola and W. Wells, “Alignment by Maximization of Mutual Information,” *IJCV*, vol. 24, no. 2, pp. 134–154, 1997.

[109] R. Brooks and T. Arbel, “Generalizing inverse compositional and esm image alignment,” *IJCV*, vol. 87, no. 3, pp. 191–212, 2010.

[110] D. Ngo, S. Park, A. Jorstad, A. Crivellaro, C. Yoo, and P. Fua, “Dense Image Registration and Deformable Surface Reconstruction in Presence of Occlusions and Minimal Texture,” in *ICCV*, 2015.

[111] M. Nguyen and F. D. la Torre, “Metric Learning for Image Alignment,” *IJCV*, vol. 88, no. 1, pp. 69–84, 2010.

[112] K. Arya, P. Gupta, P. Kalra, and P. Mitra, “Image Registration Using Robust M-Estimators,” *PR*, vol. 28, no. 15, pp. 1957–1968, 2007.

[113] E. Tola, V. Lepetit, and P. Fua, “A Fast Local Descriptor for Dense Matching,” in *CVPR*, 2008.

[114] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” in *CVPR*, 2005.

[115] L. Sevilla-lara and E. Learned-miller, “Distribution Fields for Tracking,” in *CVPR*, 2012.

[116] H. Alismail and B. Browning and S. Lucey, “Bit-planes: Dense subpixel alignment of binary descriptors,” *ARXIV*, 2016.

[117] S. Oron, A. Bar-hillel, and S. Avidan, “Extended Lucas-Kanade Tracking,” in *ECCV*, September 2014, pp. 142–156.

[118] Y. Park, V. Lepetit, and W. Woo, “Handling motion-blur in 3d tracking and rendering for augmented reality,” *IEEE transactions on visualization and computer graphics*, vol. 18, no. 9, pp. 1449–1459, 2012.

[119] A. Crivellaro, Y. Verdie, K. Yi, P. Fua, and V. Lepetit, “[demo] tracking texture-less, shiny objects with descriptor fields,” in *ISMAR*, 2014.

[120] L. Florack, B. Romeny, M. Viergever, and J. Koenderink, “The Gaussian Scale-Space Paradigm and the Multiscale Local Jet,” *IJCV*, vol. 18, pp. 61–75, 1996.

[121] C. Schmid and R. Mohr, “Local Grayvalue Invariants for Image Retrieval,” *PAMI*, vol. 19, no. 5, pp. 530–534, May 1997.

[122] I. Laptev and T. Lindeberg, “Local Descriptors for Spatio-Temporal Recognition,” in *Spatial Coherence for Visual Motion Analysis, Lecture Notes in Computer Science*, 2006.

[123] A. Larsen, S. Darkner, A. Dahl, and K. Pedersen, “Jet-Based Local Image Descriptors,” in *ECCV*, 2012.

[124] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab, “A Dataset and Evaluation Methodology for Template-Based Tracking Algorithms,” in *ISMAR*, 2009.

[125] V. Lepetit, F. Moreno-noguer, and P. Fua, “EPnP: An Accurate $o(n)$ Solution to the PnP Problem,” *IJCV*, 2009.

[126] C. Shannon, “A mathematical theory of communication,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[127] P. Thévenaz and M.Unser, “Optimization of mutual information for multiresolution image registration,” *IEEE transactions on image processing*, vol. 9, no. 12, pp. 2083–2099, 2000.

[128] G. Caron, A. Dame, and E. Marchand, “Direct model based visual tracking and pose estimation using mutual information,” *Image and Vision Computing*, vol. 32, no. 1, pp. 54–63, 2014.

[129] M. Rad, “Robust 3D Face Registration in Video Stream,” Master’s thesis, Ecole Polytechnique Fédérale de Lausanne, Switzerland, 2014.

[130] S. Song and J. Xiao, “Sliding Shapes for 3D Object Detection in Depth Images,” in *ECCV*, 2014.

[131] N. Kyriazis and A. Argyros, “Scalable 3D Tracking of Multiple Interacting Objects,” in *CVPR*, 2014.

[132] D. Damen, P. Bunnun, A. Calway, and W. Mayol-cuevas, “Real-Time Learning and Detection of 3D Texture-Less Objects: A Scalable Approach,” in *BMVC*, 2012.

[133] F. Tombari, A. Franchi, and L. D. Stefano, “BOLD Deatures to Detect Texture-Less Objects,” in *ICCV*, 2013.

[134] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” *IEEE*, 1998.

[135] A. Crivellaro, M. Rad, Y. Verdie, K. Yi, P. Fua, and V. Lepetit, “Robust 3d object tracking from monocular images using stable parts,” *PAMI*, 2016, "Submitted for publication".

[136] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *NIPS*, 2012.

[137] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated Recognition, Localization and Detection Using Convolutional Networks,” in *International Conference on Learning Representations*, 2014.

[138] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *ICLR*, 2015.

[139] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *CVPR*, 2016.

[140] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, “Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks,” in *ICIP*, 2013.

[141] S. Umeyama, “Least-Squares Estimation of Transformation Parameters Between Two Point Patterns,” *PAMI*, vol. 13, no. 4, 1991.

[142] G. Welch and G. Bishop, “An Introduction to Kalman Filter,” Department of Computer Science, University of North Carolina, Technical Report, 1995.

[143] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow., A. Bergeron, N. Bouchard, and Y. Bengio, “Theano: New Features and Speed Improvements,” in *NIPS*, 2012.

[144] F. Moreno-noguer, V. Lepetit, and P. Fua, “Pose Priors for Simultaneously Solving Alignment and Correspondence,” in *ECCV*, 2008.

[145] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A Benchmark for the Evaluation of RGB-D SLAM Systems,” in *IROS*, 2012.

[146] A. Ude, “Filtering in a unit quaternion space for model-based object tracking,” *Robotics and Autonomous Systems*, vol. 28, no. 2–3, 1999.

[147] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, “Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion,” *PR*, vol. 47, no. 6, pp. 2280–2292, 2014.

[148] F. Markley, Y. Cheng, J. Crassidis, and Y. Oshman, “Averaging quaternions,” *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.

[149] D. Eggert, A. Lorusso, and R. Fisher, “Estimating 3D Rigid Body Transformations: A Comparison of Four Major Algorithms,” *Machine Vision and Applications*, vol. 9, no. 5-6, pp. 272–290, 1997.

[150] F. Michel, A. Krull, E. Brachmann, M. Y. Yang, S. Gumhold, and C. Rother, “Pose estimation of kinematic chain instances via object coordinate regression,” in *BMVC*, 2015.

[151] H. Glöckner, “Implicit functions from topological vector spaces to banach spaces,” *Israel Journal of Mathematics*, vol. 155, no. 1, pp. 205–252, 2006.

Crivellaro Alberto

Born in 1986.
Nationality: Italian.
Tel. +41 21 69 37518
alberto.crivellaro@epfl.ch

Computer Vision PhD Candidate with 4 years working experience, determined and goal-oriented. Demonstrated ability to generate novel ideas, carry out research projects and deliver functional prototypes for both academic and industrial projects. Motivated to carry out cutting-edge R&D projects in challenging, dynamic environments.

Professional Experience

2012-Present: **Research Scientist for EDUSAFE Marie Curie ITN Project at Computer Vision Lab, Ecole Polytechnique Fédérale de Lausanne** (Switzerland). As part of a multi-national team of 12 researchers, developed an Augmented Reality based personnel safety device for technical operations in extreme environments. Proposed innovative 3D tracking algorithms leading to 3 scientific publications in the world's top Computer Vision conferences (CVPR – ICCV); deployed a working AR prototype at CERN. *Real time object tracking; 3D pose estimation; machine learning*.

2012: **Simulation Engineer at Ferrari Gestione Sportiva SpA,** Maranello, Italy (6 months). Working in the Vehicle Modeling and Simulation Team, developed a multi-variate interpolation library (C++) for fast lap-time simulations for analysis of Formula 1 vehicles. *Agile and test-driven development; fast algorithm prototyping; design patterns*.

Education

2012-Present: **PhD Program in Computer Vision at the Ecole Polytechnique Fédérale de Lausanne**, Switzerland. Thesis title: "Robust 3D Tracking from monocular images for Augmented Reality in Industrial Environments". Expected graduation: Sept. 2016.

2011: MSc in **Mathematical Engineering** at the **Politecnico di Milano** (Italy), majoring in Scientific Calculus. Grade: 110 *cum Laude*/110.

2011: MSc.: Diplôme d'Ingénieur Généraliste at **École Centrale de Lyon** (France), with the 2 years *T.I.M.E.* double degree program (Top Industrial Managers for Europe).

2005: Literal high school diploma, Liceo "*A. Racchetti*", Crema, Italy. Grade: 100/100.

Skills and Qualifications

- **Italian**: mother tongue.
- **French**: fluent. Lived and studied 2 years in France.
- **English:** fluent. TOEFL IBT, score: 107/120.

IT: **C/C++**; **Matlab; FreeFem**; **Bash scripting**; HTML; Javascript; frameworks for parallel computing and GPGPU (**OpenMPI**, **OpenMP, CUDA)**; version control (**SVN, GIT**). Wide knowledge of Linux/Windows.

Grants & Honors

2013: **Qualcomm Innovation Fellowship,** for innovation proposal “A Robust Dense Approach for Marker-less Camera Pose Estimation in Industrial Contexts”.
2012: **Departmental fellowship** of the School of Computer and Communication Sciences at the Ecole Polytechnique Fédérale de Lausanne (Switzerland).

Academic Projects and Internships

2014: **2 months R&D internship** at S&H, Peschiera Borromeo, Italy. Topic: «*Visual tracking system for a pipe welding machine*». Developed and built an embedded system for autonomous welding machine guidance in harsh environments.
2009: **2 months research internship** at CEDRAT, Meylan, France. Topic: «*New effective methods for the solution of multi-systems in magnetodynamics*».
2008: **4 weeks practical work** at Seyfert Provence, Sorgues, FR. Worker in a paper factory.

Publications

2015: A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua and V. Lepetit. **«A Novel Representation of Parts for Accurate 3D Object Detection and Tracking in Monocular Images»**. ICCV 2015.

2015: T. D. Ngo, S. Park, A. A. Jorstad, A. Crivellaro, C. Yoo and P. Fua. **«Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture »**. ICCV 2015.

2014: A. Crivellaro and V. Lepetit. **«Robust 3D Tracking with Descriptor Fields»**. CVPR 2014.

2014: A. Crivellaro, Y. Verdie, K. Moo Yi, P. Fua, and V. Lepetit. **«[DEMO] Tracking texture-less, shiny objects with descriptor fields».** ISMAR 2014.

2014: A. Crivellaro, P.Fua, and V. Lepetit. **«Dense Methods for Image Alignment with an Application to 3D Tracking».** EPFL Technical Report No. 197866.

2011: A. Crivellaro. **«Effective algorithms for surface reconstruction from scattered data».** Master thesis, Politecnico di Milano. Novel algorithm for multivariate function interpolation based on compactly supported radial basis functions.

Other Activities

Sports and hobbies: sailing; climbing; guitar; travelling.
2015: Handball player for 15 years in clubs (regional leagues) in Italy, France and Switzerland.
2011: Volounteer **Italian teacher** at the Scuola Popolare per Stranieri di Italiano, Milano.
2009: Author of the 3D pictures show **« MILANO 3D »**, at the *Institut Italien de Culture*, Lyon, FR.
2008: Responsible of the student union **Bureau International** at the École Centrale de Lyon, in charge of the reception and the aid to the foreign students.