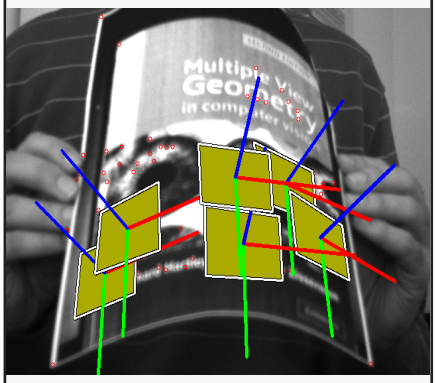
Computer Aided Medical Procedures
Prof. Dr. Nassir Navab



Dissertation

Real-Time Object Detection and Pose Estimation of Low-Textured and Texture-Less Objects

Stefan Hinterstoißer

Fakultät für Informatik
Technische Universität München

TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik
Computer Aided Medical Procedures & Augmented Reality / I16

Real-Time Detection and Pose Estimation of Low-Textured and Texture-Less Objects

Stefan Hinterstoißer

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. M. Beetz, Ph.D.
Prüfer der Dissertation:

1. Univ.-Prof. Dr. N. Navab
2. Univ.-Prof. Dr. B. Schiele, Technische Universität Darmstadt
3. Prof. K. Konolige, Ph.D., Stanford University, USA

Die Dissertation wurde am 07.04.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 09.02.2012 angenommen.

Abstract

Real-Time detection and pose estimation are two key components in various areas of computer vision, e.g. in industrial inspection, augmented reality and robotics. The task is to find the object of interest in an image and to recover its pose without having *a priori* knowledge of it. For many applications this has to work robustly and in real-time in order to be fully operational. While real-time detection of well textured objects has already reached a high level of maturity, its application on low-textured or texture-less objects is still an open issue. Unfortunately, those kinds of objects play an important role in man made environments which makes it necessary and unavoidable to deal with them in an efficient manner. In this thesis, we therefore present four novel methods for the efficient and reliable detection of low-textured and texture-less objects.

The first two approaches — made for low-textured objects — are based on fast perspective patch rectification. Although both methods follow the same general approach, one favors fast run-time performance while the other is designed for robustness and fast real-time learning. Given a reference view of a planar patch, they can quickly recognize it in new images and accurately estimate the homography between the reference and the new view. Our methods are more memory-consuming than affine region detectors and currently limited to a few tens of patches in practice. However, if the reference image is fronto-parallel to the object and the internal parameters are known, one single patch is often enough to precisely estimate the pose. As a result, we can efficiently deal with objects that are significantly less textured than the ones required by state-of-the-art approaches.

To further extend our detection capabilities to texture-less objects, we additionally propose two novel template matching methods for real-time 3D object detection that both do not require a time consuming training stage. They allow to consider thousands of templates in real-time and to represent a 3D object with only a limited set of templates. While having similar properties, the two methods come in two different flavors: while the first approach is purely gradient based and extremely fast for small templates, the second one easily incorporates different modalities and visual cues, can handle differently sized templates and is much more robust with respect to background clutter. Both approaches make use of novel representations, either of the template or of the image. Both representations are invariant to small image deformations which let us test only a small subset of all possible pixel locations when parsing the image. In combination with discretized values and their binary representation, they allow us to efficiently scan the image. As a result, both methods deal with texture-less objects in real-time while being much faster and much more robust than state-of-the-art approaches.

Keywords: Real-Time Detection, Markerless Tracking, Textureless Objects, Image based Tracking, Pose Estimation

Zusammenfassung

Detektion und Lageschätzung von Objekten sind zwei Schlüsselkomponenten in unterschiedlichen Bereichen des Computer Sehens, wie z.B. in der industriellen Qualitätssicherung, in der erweiterten Realität und in der Robotik. Ihre Aufgabenstellung besteht darin, ein Objekt in einem Bild ohne *a priori* Wissen zu finden und seine Lage zu schätzen. Für viele Anwendungen muss dies robust und in Echtzeit geschehen. Während die Echtzeiterkennung für gut texturierte Objekte schon ziemlich ausgereift ist, treten bei weniger texturierten oder texturlosen Objekten immer noch Probleme auf. Leider spielen gerade diese Objekte in menschlichen Umgebungen eine große Rolle, welche es notwendig macht, den Umgang mit ihnen effizient zu gestalten. Aufgrund dessen präsentieren wir in dieser Dissertation vier neue Methoden, um gering texturierte und texturlose Objekte effizient und verlässlich zu detektieren.

Die ersten beiden Ansätze wurden speziell für weniger texturierte Objekte entwickelt und basieren auf effizienter, perspektivischer Patch Rektifizierung. Obwohl beide Methoden den selben generellen Ansatz verfolgen, favorisiert die eine ein schnelleres Laufzeitverhalten, während die andere für Robustheit und Echtzeitlernen ausgelegt ist. Hat man eine Referenzansicht eines planaren Patches, so kann man mit den beiden Methoden schnell und genau die Homographie zwischen dem Referenzbild und der neuen Ansicht schätzen. Beide Methoden sind speicherintensiver als *Affine Region* Detektoren und beschränken sich in der Praxis noch auf wenige Dutzend Patches. Jedoch reicht meistens schon ein einziger Patch aus, um die Lage des Objektes genau genug zu bestimmen, wenn das Referenzbild fronto-parallel zum Objekt aufgenommen wurde und die internen Kameraparameter bekannt sind. Dadurch kann man u.a. Objekte detektieren, die signifikant weniger texturiert sind als jene, auf welchen *state-of-the-art* Methoden arbeiten.

Desweiteren führen wir zwei Template Matching Methoden zur Echtzeiterkennung von texturlosen 3D Objekten ein, welche keinen zeitaufwendigen Trainingsschritt benötigen. Sie erlauben es, tausende von Templates in Echtzeit zu behandeln und 3D Objekte mit einer limitierten Anzahl von Templates zu repräsentiern. Obwohl beide Methoden ähnliche Eigenschaften vorweisen, existieren sie in zwei unterschiedlichen Varianten: während der erste Ansatz Gradienten basiert und besonders für kleine Templates extrem effizient ist, kann die zweite Methode viele verschiedene Modalitäten miteinbeziehen, ist robuster in Bezug auf Hintergrundrauschen und kann verschieden große Templates effizient behandeln. Beide Ansätze basieren auf einer neuartigen Template- bzw. Bilddarstellung, welche sie invariant gegenüber kleinen Bilddeformationen macht. Dies erlaubt es uns, nur eine kleine Untermenge aller möglichen Pixel Positionen zu betrachten. In Kombination mit diskretisierten Werten und ihrer binären Darstellung ermöglicht dies uns, das Bild effizient abzutasten. Dadurch können texturlosen Objekte in Echtzeit detektiert werden, wobei beide Methoden schneller und robuster sind als alle bisherigen Ansätze.

Schlagwörter: Real-Time Detektierung, Markerloses Tracking, Texturlose Objekte, Bildbasiertes Tracking, Posenschätzung

Acknowledgements

First of all, I would like to thank my PhD advisor Prof. Dr. Nassir Navab for making me part of his chair and for providing the best and most uncomplicated PhD environment I can imagine. In particular, I thank Nassir that he always gave me the trust and the freedom to do research in the field I was most interested in. His patience, his thoughts and his philosophy shaped my understanding of research, and I am very grateful to this.

I specially owe many thanks to Dr. Vincent Lepetit for being my main supervisor during the past few years. Besides allowing me to spend three month in his lab, I thank him for all the good discussions, for his invaluable advice, for his interest in my ideas, for not giving up teaching me how to write good papers and for his very important advice of not relying too much on the expertise of reviewers. It was really a privilege for me to work with you, Vincent!

I also would like to thank Dr. Selim Benhimane and Dr. Slobodan Ilic who were accompanying and co-supervising my PhD in the first and the second half, respectively. They always gave me valuable input, discussed new ideas with me and helped me a lot writing my papers. Thank you both very much for that!

In addition, I would like to thank all the people from CAMP, which (some more, some less) helped me to achieve my research goals. In particular, I want to thank Stefan Holzer, Pierre Georgel and Jürgen Sotke for being my supportive office mates who were always open to discuss current problems and ideas. Moreover, I would like to thank Oliver Kutter, Cedric Cagniart, Prof. Dr. Pascal Fua, Prof. Dr. Peter Sturm and Nicolas Alt for fruitful collaborations. Many thanks also to the rest of the vision team (especially to Diana Mateus who proof read some of my publications). In addition, I would like to thank Martin Groher, Ben Glocker, Hauke Heibel, Olivier Pauli, Christian Wachinger, Darko Zikic, Andreas Keil, Max Baust, Jose Gardiazabal, Selen Atasoy, Jörg Traub, Marco Feuerstein, Stefanie Demirci, Tobias Lasser, Nicolas Brieu, Richard Brosig and many more for working in the most collaborative and fun environment. I may not forget to mention Martin Horn and Martina Hilla who both helped me to find my way through the technical and administrative jungle, and to whom I am also very grateful.

I also would like to thank WillowGarage, and in particular Dr. Kurt Konolige, Dr. Gary Bradski and Dr. Radu Rusu for their joint collaboration and for the possibility to spend some time in their amazing labs.

I am very much obliged to my parents Eva and Hans Hinterstoißer who always supported me during my whole life, letting me grow up in a loving environment, never put pressure on me concerning my future plans, and thus made this thesis possible in the first place. I also thank my sister Kathrin who is a little sunshine to me — if she wants. Thanks a lot also to all my friends who always showed me that there is a life beyond work. I also have to mention Edeltraud Weidemann who gave me the feeling that she is proud of what I am doing and who gave me some good advice in the past.

Finally, I want to thank Anja — for nothing special as she recently said — but indeed for much more as she might believe.

CONTENTS

THESIS OUTLINE

While real-time detection and pose estimation of well-textured objects have been subject to extensive studies in the past years, relatively little effort was spent to improve the real-time detection of low-textured or texture-less objects. However, especially these kinds of objects appear quite often in human environments. So it becomes evident that the success or failure of many vision based applications depends on the capability to handle these kinds of objects well.

In this context, we present four novel algorithms to provide a solution to that problem and to initiate the further development of new applications. In the following, we give a brief outline of the single chapters of this thesis.

Chapter 1: Introduction In our first chapter we introduce real-time object detection and pose estimation as two key components in computer vision and motivate their importance on a variety of fundamental vision-based applications like augmented reality, industrial inspection, robotics and man-machine interfaces. We describe fundamental directions that have been followed and show what kind of objects are still problematic to detect. Furthermore, we point out some inherent challenges which have to be tackled in order to guarantee robust real-time behavior. We conclude this chapter by listing our contributions.

Chapter 2: Real-Time Detection of Low-Textured Objects by Patch Based Rectification Our second chapter is dedicated to the first part of our main contribution: the real-time detection and pose estimation of low-textured objects. In this context, we introduce two different methods — called LEOPAR and GEPARD. Although both methods have the same general design they come in two different flavors: one emphasizes runtime efficiency while the other is made for robustness and online training. Before explaining the single steps of the two methods in detail, we give an introduction and an overview of related work. This is followed by an explanation of how patch based rectification is done — starting with the initial coarse pose estimation, continuing with the refinement method and finishing with the final self-verification. We also show how incremental learning is performed and give qualitative and quantitative experiments. The chapter is concluded by discussing possible applications and showing exemplary augmented images.

Chapter 3: Real-Time Detection of Texture-Less Objects by Template Matching Chapter three covers the second part of our main contribution: the detection of texture-less objects. The two different methods we present in this chapter — called DOT and LINE — are both based on real-time template matching. After giving a short introduction on the detection of texture-less objects and an overview of related work, we start explaining our two methods.

We first derive the gradient based similarity measure for DOT. We show how this measure can be speeded up at runtime by making the template invariant to small deformations and by using a binary representation of the gradient orientations. The efficiency of this approach can be further increased by applying SSE instructions and by introducing clustering. We test DOT against current state-of-the-art approaches and conclude by showing possible fields of application.

In the second part of this chapter, we show the development of LINE. In contrast to DOT, we derive a general framework to enable the simultaneous integration of many different modalities and visual cues. Again, we first derive the similarity measure and show how this measure introduces invariance to small image deformations. We explain how our method can be speeded up by linearizing the memory and by applying SSE instructions. Furthermore, we show how to preprocess data to incorporate it into our framework and demonstrate it on the color image and the dense depth map. In a theoretical part we compute the speed improvement of LINE in contrast to its brute force version. This is followed by an experimental section in which we compare LINE against current state-of-the-art approaches and show its advantages. We finish this chapter by demonstrating the robustness of LINE on exemplary images showing the detection of texture-less objects in highly cluttered scenes.

Chapter 4: Outlook Chapter four gives an outlook of future fields of research inspired by the work presented in this thesis. We will discuss open problems like partial occlusion, scalability, accurate pose estimation for 3D objects, automatic learning from 3D models and object class detection.

Chapter 5: Conclusion We conclude in chapter five and give some major insights of this doctoral thesis.

CHAPTER

ONE

INTRODUCTION

In this thesis, we tackle the problem of fast and robust detection of low-textured and texture-less objects from visual and multimodal data (see Fig. 1.1). The notion *detection* refers to the problem of determining whether or not an image contains a specific object. If the object is detected, *pose estimation* provides the relative position and orientation between the camera and the object. The combination of detection and pose estimation is also often called tracking-by-detection. In this case, no a-priori information about the current input image is available. This is a much harder problem than the so called "tracking" approaches [3, 52, 8] which use the a-priori information computed in the previous images and thus, restrict the search space drastically. However, once these methods loose track (e.g. due to too fast motions, image blur or simply because the object is not in the image any more), they fail in giving the correct pose of the object. In such a case, they have to be reinitialized by above mentioned tracking-by-detection approaches.

(a) (b)

Figure 1.1: Detection of two texture-less objects with the method introduced in Sec. 3.3. **(a)** Detection of an hole-punch. **(b)** Detection of a toy monkey. Note that the highlighted object contours fit to the current poses of the objects. In both images detection is performed in front of heavy cluttered background.

1.1 Motivation

Nowadays, detection and pose estimation of arbitrary objects play an important role in many application areas. Their main requirements are efficiency, robustness and accuracy.

While for this dissertation we initially focused on the detection of low-textured objects, we also turned in the course of this work to the much harder problem of detecting texture-less objects. Although low-textured and texture-less objects have similar object properties, they fundamentally differ in how they can be robustly detected. In order to understand what makes them different, we first have to make clear what we understand as low-textured and texture-less objects. In order to do so, we first have to have a look on what is called texture.

Although texture is an important term in computer vision, there is no precise definition of it. The main reason is that natural textures often display different yet contradicting properties, such as regularity versus randomness, uniformity versus distortion, which can hardly be described in a unified manner.

For example, the image of the leaves of a tree contains variations of intensities which form certain patterns called visual texture. However, this pattern is simultaneously observed as a random and a regular one.

The patterns themselves can be the result of physical surface properties such as roughness or oriented strands which often have a tactile quality, or they could be the result of reflectance differences such as the color on a surface.

Many authors [100, 94, 37, 86, 113] tried to define texture, but up to now no standard definition has been accepted within the community. Therefore, we fall back to a widely accepted method: we simply display some exemplary images (see Fig. 1.2) to show what we understand as texture. In Fig. 1.2(a), we see four objects which are heavily textured. Informally, textured objects could be well described as objects with accumulated variations in local image areas of intensity or color.

In contrast to Fig. 1.2(a), Fig. 1.2(b) and (c) show no or only few local areas with intensity or color changes. The mainly consist of homogeneous regions (homogeneous either in intensity or in color). Therefore, we define objects like the ones shown in Fig. 1.2(b) as texture-less and the ones shown in Fig. 1.2(c) as low-textured objects. As we will see, especially those kinds of objects are most difficult to handle.

In the following, we will give a short overview of impacted domains, point out some inherent difficulties and motivate the need for efficient and robust algorithms to detect low-textured and texture-less objects (see Fig. 1.2(b) and (c)).

1.1.1 Augmented Reality

One of the most prominent applications for efficient detection and pose estimation is Augmented Reality (AR). The purpose of augmented reality is to enhance the information we naturally receive, by superimposing artificial elements in a way that is seamless to the user. This enables us to bring complementary information and meaning into the real world that may not be perceived by natural means, making the surrounding real world interactive and digitally usable.

(a) (b) (c)

Figure 1.2: In this figure, we show four objects for each object category: **(a)** well-textured objects, **(b)** texture-less objects and **(c)** low-textured objects. In case of the low-textured objects, only the logos provide sufficient texture to detect and estimate the pose of the object.

In order to fulfill this task, real world objects need to be found and tracked, so that the augmentation can be performed. Initially, this was done using artificial markers. Artificial markers make it possible to achieve good tracking results even in difficult situations with illumination change or partial occlusion e.g. by using high contrast colors or by placing redundant markers in the scene.

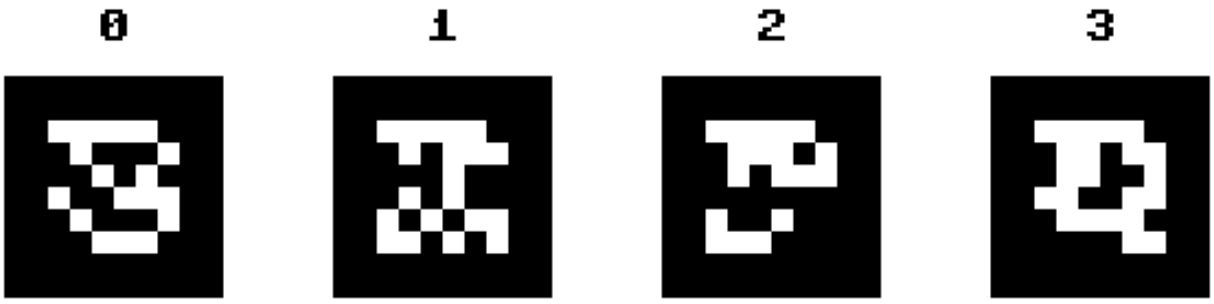
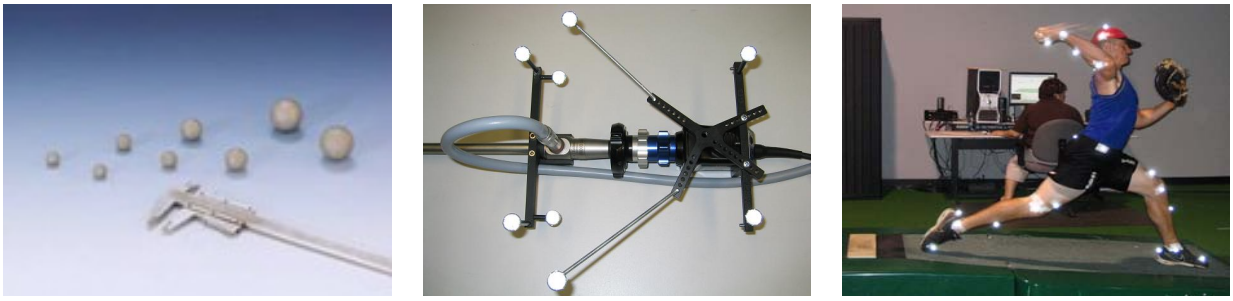


Figure 1.3: Artificial paper-based marker encoding additional information (courtesy of ARTag).

Artificial markers usually come in two different flavors: paper-based (Fig. 1.3) and retro-reflective makers (Fig. 1.4). Paper-based markers are very easy to use and to detect and allow to encode additional information. However, they need relatively large planar areas to be attached to, which constrains their usability in real applications. In Fig. 1.5 we show some examples where real world scenes are augmented using the pose provided by paper-based markers.

In contrast to paper-based markers, retro-reflective markers are smaller, and can be therefore more easily incorporated into the scene. They have a special coating which allows reliable detection in the infrared spectrum. This requires infrared cameras which usually cause additional costs in terms of money and effort. However, once everything is



(a) (b) (c)

Figure 1.4: **(a)** Artificial retro-reflective markers (courtesy of A.R.T.GmbH). **(b)** Retro-reflective markers attached to a medical device. **(c)** Retro-reflective markers attached to a person for action monitoring (courtesy of TMI Sports Performance).

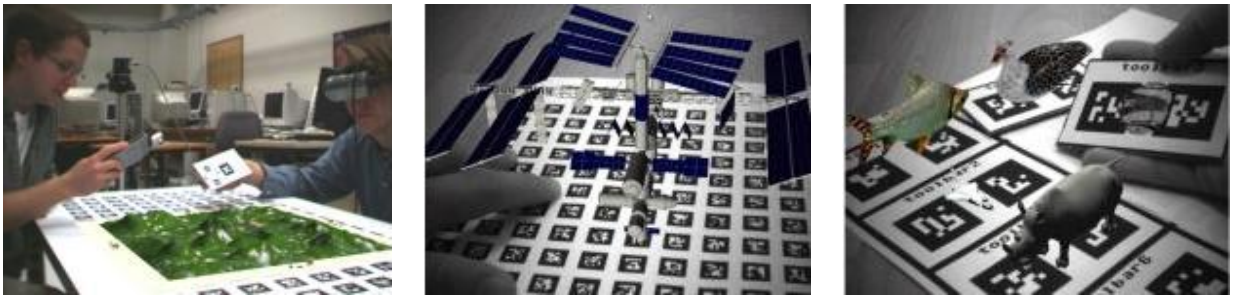


Figure 1.5: Augmentation of real world scenes with virtual objects. The objects are mainly planar and do not look realistic after markers have been attached to them (courtesy of ARTag).

set up, the accuracy and the efficiency of retro-reflective markers is usually so good that they are even applied in medical systems.

Unfortunately, all artificial markers need to be manually placed into the scene which is not always possible or desirable. Additionally, these markers occlude real world objects, and it is expensive to revert this occlusion for a realistic impression of the scene.

At this point markerless tracking comes into play: markerless tracking is non intrusive, i.e. no manual placement of artificial markers is required, and thus no special treatment for occluded objects is needed. Instead, objects are tracked by using their characteristic appearance only. While markerless tracking was initially less robust, less reliable and much slower than maker-based tracking, it has been tremendously improved over the last few years. Especially in case of well-textured objects, markerless tracking can be performed fairly satisfactorily now (e.g. [63, 68, 101, 34, 59, 81, 15, 6, 8, 51, 54]).

However, this does not hold for all kind of objects. While markerless tracking of well-textured objects (see Fig. 1.2(a)) is pretty robust, reliable and efficient nowadays, low-textured or texture-less objects (see Fig. 1.2(b) and (c)), as they often appear in man-made environments, still cause a lot of problems: reduced speed or lack of robustness are only some of the challenges that have to be solved (see Sec. 2.1 and Sec. 3.1 for a more detailed discussion).

This is one of the main reasons why the impact of augmented reality in our daily live is currently not very strong and often limited to fun applications (see Fig. 1.6). However, we believe that augmented reality can easily be pushed to the next level by developing methods for the robust and efficient detection of low-textured and texture-less objects. In this way, we will be able to handle all kinds of objects in human environments which will significantly extend the possible field of future augmented reality applications.

(a) (b) (c)

Figure 1.6: In this figure we show various products of Augmented Reality companies. **(a)** Lego box augmented with a Lego house (courtesy of Metaio GmbH). **(b)** Baseball card augmented with a baseball player (courtesy of Total Immersion and Topps). **(c)** Two augmented robots fighting against each other (courtesy of Qualcomm). Note that all tracked objects are well textured.

1.1.2 Industrial Inspection

Another prominent domain for real-time detection and pose estimation is industrial image processing. Here, tracking-by-detection is typically used to find special objects in order to analyze them, e.g. for visual inspection, visual measuring or visual surveying (Fig. 1.7). For these applications, the major requirements are robustness, speed and accuracy: to recognize damaged parts, for instance, the object of interest has to be robustly found and its pose has to be accurately estimated to allow reliable and precise measurements. This has to be done as fast as possible to enable a high throughput rate, and thus to reduce the production costs.

To meet these demands, industrial environments very often allow supportive and constrained surroundings. Some of the constraints that can be utilized are lighting conditions, camera calibration, search space or noise. Using artificial markers, as done in AR applications (see Sec. 1.1.1), is nevertheless no option because it is most of the time not possible or efficient to attach them to the objects of interest.

In contrast to marker-based approaches, marker-less tracking provides efficient means to detect the object reliably without any additional help. However, since industrial objects usually show only little or almost no texture, are very often non-planar and highly reflective, state-of-the-art texture based approaches can not be applied. Also, current methods for the detection of low-textured and texture-less objects exhibit serious problems in terms of speed and robustness (see Sec. 2.1 and Sec. 3.1 for a more detailed

discussion). Therefore, alternative solutions for the detection of those kinds of 3D objects have to be developed.

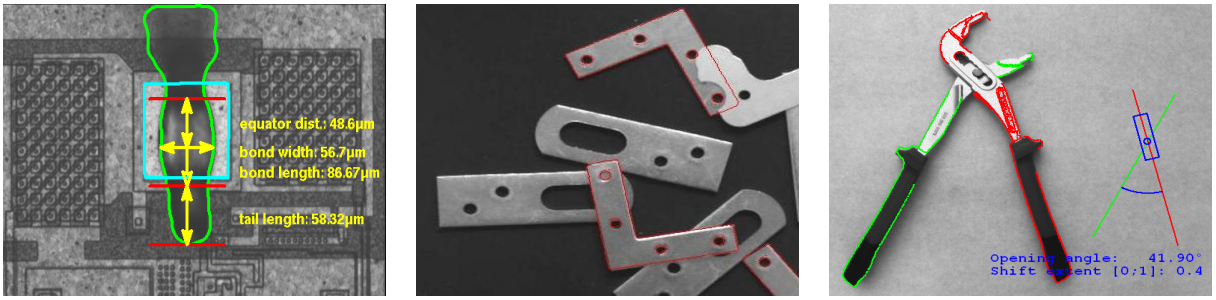


Figure 1.7: Mechanical components recognition and measurement. Measurements (e.g. euclidean distances, areas etc.) have to be precise in order to avoid using defect components. One can see that the objects of interest do not exhibit much texture (courtesy of MVTec).

1.1.3 Robotic Applications

Due to the high computational power of modern computers and the ongoing improvements in computer vision, artificial intelligence and mechanics, the number of robots used in daily life has grown during the last decades.

The purpose of robots is to fulfill tasks that humans are not able or willing to do: the job may be boring, such as domestic cleaning, exhausting or difficult, such as assembling cars on an automated production line, or dangerous, such as removing landmines. Other jobs are physically inaccessible, such as exploring the deep sea, cleaning the inside of narrow tunnels, or performing minimally invasive surgeries.

Robots are often divided into two main categories: dedicated and general-purpose robots. Dedicated robots are designed to efficiently perform one particular task extremely well and are used in a wide field of applications, for instance in car production, packaging and electronics.

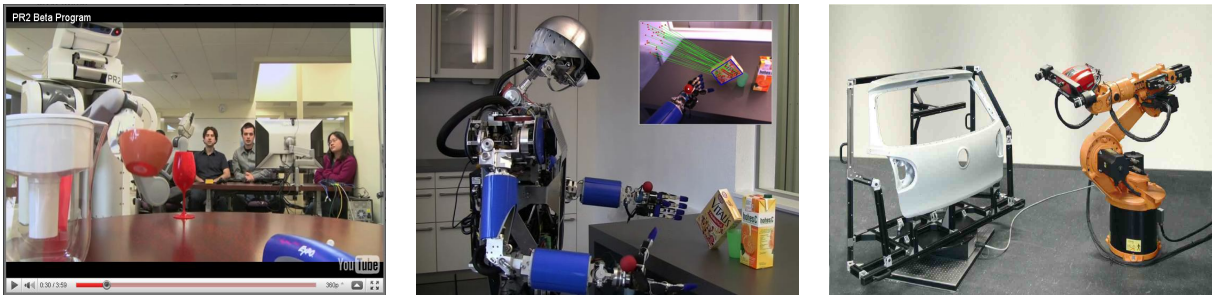
On the contrary, general-purpose robots are systems that can perform a variety of functions independently, however less efficient. Their applications include various services such as home use, health care, disaster rescue, transportation and many more.

While dedicated robots are already intensively used in industrial environments, interactive and general-purpose robots are not yet part of our daily life.

One of the main reasons for it is that general-purpose robots have to interact with their environment to complete a specific task. This makes it necessary to perceive the surrounding space quickly and to interpret it reliably. In other words, general-purpose robots need to efficiently detect and recognize arbitrary objects in their direct neighborhood and estimate their pose. In addition, it is often of great importance to quickly learn new objects online, since general-purpose robots usually live in fast changing environments in which never seen objects show up.

Due to the unconstrained human surroundings, this results in handling not only well-textured but also low-textured and texture-less objects, which is currently one of the main

problems in robotics and also one of the reasons why autonomous systems are not yet used in our daily life.



(a) (b) (c)

Figure 1.8: In this figure we show some exemplary robotic applications. **(a)** The PR2 robot picks a homogeneous bowl (courtesy of WillowGarage). **(b)** Visual servoing using a well-textured box (courtesy of Karlsruher Institut für Technologie). **(c)** Industrial robot in the car industry detecting the backdoor of a car (courtesy of Université de Saint Etienne).

1.1.4 Man Machine Interfaces

Another important application for tracking-by-detection is its integration into man-machine interfaces. Such user interfaces allow humans to operate machines in a more natural, efficient and intuitive way. This is often achieved by manipulating familiar objects, and thus taking advantage of their everyday experience [58]. In this context, marker-less tracking-by-detection is the appropriate technique for seamless interaction with physical objects, without the need of changing them. For instance, by continuously tracking the pose of a special hand-held object, the object could be used as an alternative 3D pointer [58]. It could then serve as what is known as *Tangible User Interface* [70, 17, 74]. A Tangible User Interface is defined as a new interface type that interlinks the digital and physical world [93].

Since many man-made objects in human environments show only little or no texture at all, there is again the need for efficient and robust algorithms to detect those items in an efficient and reliable way, as discussed in the previous sections.

1.2 Challenges

In contrast to methods based on artificial markers, marker-less approaches do not have any additional support in the form of attached markers. Therefore, the visual appearance of the object is the only information that can be used for detection. In case of low-textured and texture-less items, this visual information is pretty limited.

For instance, low-textured objects, as displayed in Fig. 1.2(c), show texture only in spatially limited areas. With current state-of-the-art algorithms [59, 15, 63, 6, 34, 101] it is quite difficult to extract enough information from these local areas to efficiently compute

(a) (b) (c)

Figure 1.9: Possible tangible interfaces are shown in **(a)** (courtesy of Tangible Media Group, MIT Media Lab). In **(b)** and **(c)** real game board pawns (courtesy of EPFL) and a PDA are used as a tangible interfaces (courtesy of the University of Cambridge).

the object pose. This is because they usually make extensive use of what is called a *consensus set* [28] of multiple matched features. A consensus set is defined as a set of data points that obey the same model (transformation and object model). Since low-textured objects do not exhibit enough features to form such a reliable consensus set, we have to find an alternative solution to detect the object reliably.

In contrast to textured or low-textured objects, texture-less objects do not show any texture at all and consist of homogeneous areas only (see Fig. 1.2(b)). As a result, only few visual cues are left which are robust to illumination change and noise. One of them is gradient information. However, although gradients can be robustly extracted, they are often ambiguous and in case of occluding contours even not attached to a specific object location. Additionally, gradients are sparse features and only discriminative if seen in a global context.

Another robust visual cue, well suited for the detection of texture-less objects, is the dense depth map generated by structured light. However, due to the dense character of the data it is not obvious how to use, process and integrate it to obtain a robust but very efficient performance.

As one can see, it is quite challenging to do detection on texture-less objects. In addition, special care has to be taken to robustly handle the background as strong clutter or noise might lead to many false positives or false negatives.

While robustness is one of the main topics to be covered for the detection of texture-less objects, efficiency is another: opposite to texture-based methods, texture-less approaches can not make use of keypoints to reduce the runtime. Since there is also no a priori information available about the object pose, the whole image has to be analyzed and all possible poses and viewpoints have to be considered to recognize the object from different views. This leads to a combinatorial explosion in the search space which makes it hard to speed-up the processing without loosing robustness.

While efficiency at runtime is very important, efficiency at learning is another key feature that many applications require (e.g. see Sec. 1.1.3). Most desirable is real-time learning — also referred to as online learning — since it allows to adapt to changing situ-

ations. Yet, it is usually not easy to guarantee online learning and real-time performance at the same time, as they are oppositional constraints.

To summarize, the challenges of this thesis lie in making the detection of low-textured and texture-less objects robust and efficient, both at learning and at runtime, while having only limited visual information available.

1.3 Contributions

In this thesis, we introduce four new real-time detection and pose estimation approaches for low-textured and texture-less objects which significantly improve currently available solutions for these kinds of objects. In this line of work, we also focus on making the learning of our proposed approaches efficient enough to allow their use in online applications. In the following, we shortly list the different contributions of this dissertation.

Contributions to Real-Time Detection of Low-Textured Objects: we propose two new real-time detection and pose estimation methods for low-textured objects. Both methods are designed to handle each single feature separately and independently from all the others, and compute the pose of the object only based on the local surrounding of each feature point. Thus, they differ from current state-of-the-art approaches which need to extract and match [59, 15, 63, 6, 34, 101] multiple feature points in order to compute the pose with robust estimators [28].

To our best knowledge, we are the first to make use of this concept — which we call *perspective patch rectification* — in real-time. Applying it on a single keypoint is then often enough to estimate the 3-D pose of the object where the keypoint lies on, given that a fronto-parallel view of the keypoint is provided for training. As a result, both approaches need significantly less texture than usually required by current state-of-the-art-approaches. This is because our two new methods operate on spatially limited regions, and thus only the local areas and not the whole object has to be sufficiently textured.

Both approaches make use of a fast pre-classification, in which — contrary to previous classification methods — not only the identity but also a coarse pose of the object is estimated. An additional refinement step based on efficient linear predictors gives then the exact local pose and enables self-verification by similarity computation. False positives are reliably removed after this step.

Although both approaches work in real-time and use the same general pipeline, they come in two different flavors: one method favors runtime speed using a Ferns based classifier while the other supports robustness and real-time learning by exploiting mean patches and a precomputed principal component analysis basis for fast learning. Choosing between the two methods then depends on the application at hand.

Contributions to Real-Time Detection of Texture-Less Objects: we also propose two novel real-time approaches for the detection of texture-less objects. Since texture-less objects usually do not show feature points, only methods based on the dense analysis

of the image can be used. One prominent representative of this class of algorithms is template matching using a sliding window approach. While being well suited in theory, current state-of-the-art template matching methods exhibit severe problems in practice: they are either inefficient [97, 1] or prone to misdetection [12, 30, 50, 90, 48].

This made us developing two new robust approaches based on real-time template matching, where the templates can both be built and matched very quickly. As we will show, this makes it very easy and virtually instantaneous to learn new incoming objects by simply adding new templates to the database while maintaining reliable real-time recognition. As a result, both approaches are suitable for online applications.

Our novel methods make use of gradient orientations, and thus are much more robust than efficient binary state-of-the-art approaches [12, 30, 50, 90, 48]. In order to increase efficiency at runtime, we discretize the gradient orientations which allows us to introduce invariance to small deformations. This enables us to skip pixel positions during runtime without loosing robustness, and to represent a 3D object with only a limited number of templates. In addition, discretization permits the use of bitwise operations and SSE instructions for a further speed-up. As a result, both approaches presented in this chapter are significantly more efficient than previous gradient based methods [97, 1].

While the first proposed approach is purely based on image gradients, we generalize the second one to also incorporate other visual cues like e.g. dense depth data. By using several different cues, we increase robustness and significantly decrease the number of false positives.

Our two methods differ in the way how they introduce invariance to small deformations: while the first approach makes the template invariant, the second introduces invariance to the image. Following the first method turns out to be highly efficient, especially for small objects where all templates have equal size. The second method, although less efficient, is much more robust with respect to strong background clutter and can handle differently sized templates during runtime without loosing efficiency.

Thanks to our novel template and image representation the two proposed methods show a combination of robustness and efficiency which is superior to current state-of-the-art algorithms. To our best knowledge, this is the first time that real-time template matching becomes tractable in a large scale environment.

CHAPTER

TWO

REAL-TIME DETECTION OF LOW-TEXTURED OBJECTS BY PATCH BASED RECTIFICATION

For several years retrieving the poses of patches around keypoints in addition to matching them has been an essential task for many applications such as vision-based robot localization [32], object recognition [88] or image retrieval [19, 85] to constrain the problem at hand. It is usually done by decoupling the matching process from the keypoint pose estimation: the standard approach is to first use some affine region detector [68] and then rely on SIFT [63] or SURF [6] descriptors on the rectified regions to match them.

Recently, it has been shown that taking advantage of a training phase, when possible, greatly improves the speed and the rate of keypoint recognition tasks [34, 80]. Such a training phase is possible when the application relies on some *database* of keypoints, such as object detection or SLAM. By contrast with Mikolajczyk *et al.* [68], these learning-based approaches usually do not rely on the extraction of local patch transformations in order to handle larger perspective distortions but on the ability to generalize well from training data. The drawback is that they only provide a 2–D location, while using an affine region detector provides additional constraints that proved to be useful [88, 19].

Such constraints are especially important if we have to deal with poorly textured objects which are often found in human environments as seen in Chapter 1. They consist of large homogeneous areas and show texture only in spatially limited areas. As a result, state-of-the-art feature point approaches are often not applicable and a correct object pose can not be computed.

To overcome this problem we introduce an approach illustrated in Fig. 2.1 that is real-time thanks to a learning stage and that can provide not only an affine transformation but the full perspective patch rectification based on a limited spatial region. We show that this is very useful for object detection and SLAM applications: applying our approach on a single keypoint [38, 63, 95, 87, 57] is often enough to estimate the 3–D pose of the object that the keypoint lies on, provided that a fronto-parallel view of the keypoint is given for training. As a result, we can robustly handle strongly occluded or very poorly textured objects.

More specifically, we propose two variants of this approach. The first method [41] is called Leopar. It is the faster of the two methods and much more accurate than

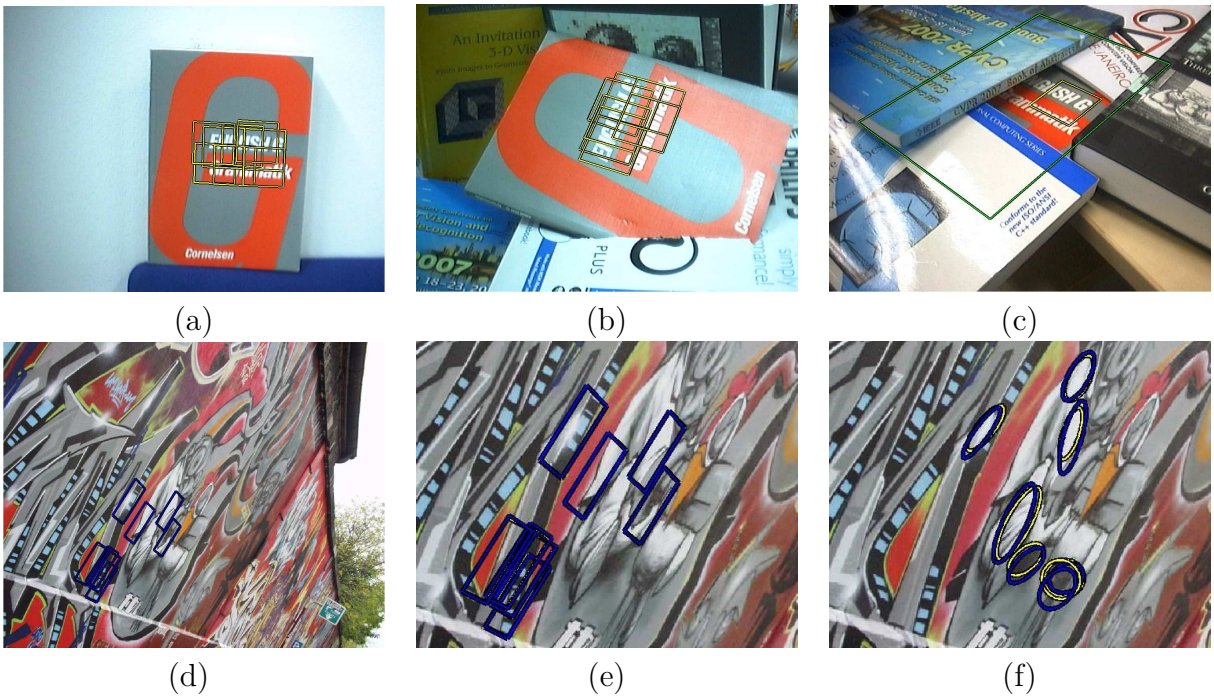


Figure 2.1: The advantages of learning for patch recognition and pose estimation. **(a)** Given a training images or a video sequence, our method learns to recognize patches and in the same time to estimate their transformation. **(b)** The results are very accurate and mostly exempt of outliers. Note we get the full perspective pose, and not only an affine transformation. **(c)** Hence a single patch is often sufficient to detect objects and estimate their pose very accurately. **(d)** To illustrate the accuracy, we use the 'Graffiti 1' image and the ICCV booklet cover respectively to train our method and detect patches in the 'Graffiti 6' image and in the real scene respectively. We then superimpose the retrieved transformations with the original patches warped by the ground truth homography. **(e)** Even after zooming, the errors are still barely visible. **(f)** By contrast, the standard methods retrieve comparatively inaccurate transformations, which are limited to the affine transformation group.

affine region detectors. However, Leopar needs a long training stage of approximately one second per patch that avoids its use in online applications. The second method [44] is called Gepard. Gepard produces more reliable results than Leopar and requires only a very short training period. Choosing between the two methods depends on the application at hand.

Both methods are made of two stages. The first stage relies on a classifier to quickly recognize the keypoints and to provide a first estimate of their poses to the second stage. This second stage then uses slower but much more accurate template matching techniques to refine the pose estimate. The difference between the two methods lies in the way the first stage proceeds.

Leopar first retrieves the patch identity and then a coarse pose using an extended version of the Ferns classifier [80]: to each keypoint in our database we correspond several classes, where each class covers its possible appearances for a restricted range of poses. Due to the Fern structure the computations can be done in almost no time which results in a very fast runtime performance.

Unfortunately the Ferns require a long training stage and a large amount of memory. Gepard dramatically decreases the training time and the memory requirements and is even more robust; however, it is slower than Leopar at runtime. In Gepard, the Ferns classifier is replaced by a simple nearest-neighbor classifier, as new classes can be added quickly to such a classifier. To retrieve the incoming keypoint identities and approximate poses, each keypoint in the database is characterized in the classifier by a set of "mean patches", each of them being the average of the keypoint appearances over a restricted range of poses. Our mean patches are related to Geometric Blur [9], but we show how to very quickly compute them, making our approach more efficient.

To retrieve an accurate full perspective transformation, the second stage uses linear regressors similar to the one described in [52] for template matching. We made this choice because our experiments proved that for this purpose they converge faster and more accurately than other least-squares optimizations such as Gauss-Newton. In addition, we show that these regressors can be trained efficiently. The final pose estimate is typically accurate enough to allow a final check by simple cross-correlation and a rejection of incorrect results.

Compared to affine region detectors, the closest competitors in the state-of-the-art, our two methods have one important limitation: They do not scale very well with the size of the keypoints database, and our current implementation is limited to a few tens of keypoints to keep the applications real-time capable. Moreover, they need a frontal training view and the camera internal parameters to compute the camera pose with respect to the keypoint. However, as our experiments show, our two methods are not only much faster but they also provide an accurate 3–D pose for each keypoint, by contrast with an approximate affine transformation. In practice, a single keypoint is often enough to compute the camera or target pose, which compensates this limitation on the database size for the applications we present in this chapter and makes our approaches especially suitable for low-textured objects as described above.

In the remainder of the first chapter, we first discuss related work. Then, we describe our two methods, and compare them against affine region detectors [68]. Finally, we present applications of tracking-by-detection and SLAM using our methods.

2.1 Related Work

Many different approaches often called "affine region detectors" have been proposed to recognize keypoints under large perspective distortion. For example, [105] generalized the Förstner-Harris approach, which was designed to detect keypoints stably under translation, small similarities and affine transformations. However, this method does not provide the transformation itself. Other approaches attempt to retrieve a canonical affine transformation without *a priori* knowledge. This transformation is then used to rectify the image around the keypoint which makes it easier to recognize it. Baumberg [5] was among the first who proposed such an approach. His method extracts keypoints at several scales using the Harris detector [38] and then adapts the shape of the point neighborhood to the local image structure using the iterative procedure proposed by Lindeberg [61].

Since then, many affine region detectors have been developed, e.g. the Intensity-Based-

Region detector (IBR) [106], the Edge-Based-Region detector (EBR) [107], the Salient-Region detector [53] and the Harris-Affine region detector [65]. The IBR detector makes use of the shape constructed by the maxima of intensity profiles on rays originating from the center of local intensity extrema. Although firing on similar regions, the EBR detector provides a different solution that is geometric-based and uses corner points and nearby edges to extract affine regions. In contrast to them, the salient region detector describes each regions based on the probability density function of intensity values computed over an elliptical region. Its final outcome is then ranked with respect to scale and with respect to the magnitude of the derivative of this function. This enumeration is concluded with the Harris-Affine detector which is based on the approach of Baumberg [5] and extended by a scale dependent iterative affine shape adaption algorithm.

Although these detectors work reasonably well, Mikolajczyk *et al.* showed in [68] that the Hessian-Affine detector of [66] and the MSER detector of [64] are the most reliable ones. In the case of the Hessian-Affine detector, the retrieved affine transformation is based on the image second moment matrix. It normalizes the region up to a rotation, which can then be estimated, for example, by considering the peaks of the histogram of gradient orientations over the patch as in SIFT [63] or SURF [6]. In the case of the MSER detector, other approaches exploiting the region shape are also possible [78], and a common approach is to compute the transformation from the region covariance matrix and solve for the remaining degree of freedom using local maxima of curvature and bitangents.

While all these detectors are based on the gray value image, Stöttinger *et al.* [98] recently introduced a detector based on color saliency which allowed a reduction in the number of extracted regions while maintaining a high recognition performance.

After rectification, each patch usually needs a proper description of its content in order to be correctly identified. For that reason descriptors like GLOH [67] and SIFT [63] were developed: once a region is extracted it is first normalized to a canonical representation and then described by a robust method. Of the many proposed descriptors, SIFT is probably the one which is most commonly used. However, SIFT is pretty slow and for a faster description several approximations have been proposed: SURF [6], for instance, was among the first approximations of SIFT. It makes intensively use of integral images and haar features which allow to compute the feature vectors on different scales in constant time. Another more recent approximation is CHoG [18] which uses a bitwise compression scheme in order to lower the memory consumption for mobile devices. Its gradient histogram is represented as a tree structure which can be efficiently compressed using Gagie-Entropy-Tree coding. As a result, it is 10 times faster than SIFT and reduces the memory consumption about 20 times.

While all these approaches were developed for a sparse computation, DAISY [103] was designed for a dense description of the whole image: for example, computing DAISY on a SVGA image for tasks like 3D reconstruction [104] takes only five seconds.

Once the description process is finished, all image patches need to be properly matched. For this reason, several efficient data structures have been proposed: the approximate best-bin-first algorithm [7] on kd-trees [29], for instance, is used on large SIFT feature databases, while vocabulary trees [77] are often applied on very large image databases. The latter enables an efficient preliminary search for a small subset of the closest database

images using hierarchical k-means clustering and an inverted file system. Pairwise comparison of the query image with the images in this subset is then computationally tractable.

Since many different data structures have been proposed for fast matching, Muja *et al.* [72] recently presented an automatic framework for providing guidance on selecting the best algorithm and its parameters for any given dataset.

Above mentioned affine region detectors, feature descriptors and matching data structures have proven to be the key ingredients for many computer vision tasks. In this context, the affine region detectors are mainly used to rectify patches to help the recognition. However, they can also provide some other very useful constraints: [88], for example, uses them to build and recognize 3–D objects in stereoscopic images, [19] needs them to add constraints between the different regions to help the matching and [55] applies them to directly estimate the parameters for conjugate rotation.

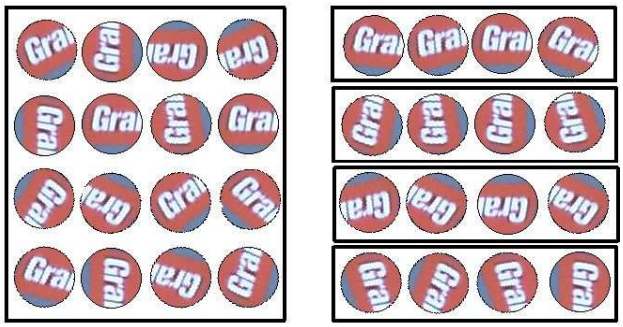
Unfortunately, as the experiments presented in this chapter show, the retrieved affine transformations are often not accurate enough and the algorithms too slow for real-time processing. As we will show in this thesis, we can reach a much better accuracy at a much higher speed by exploiting learning-based methods

Learning-based methods to recognize keypoints became quite popular recently, however most of them provide only the identity of the points, not their pose. For example, in [59, 81], Randomized Trees are trained with randomly warped patches to estimate a probability distribution over the classes for each leaf node. The non-terminal nodes contain decisions based on pairwise intensity comparisons which are very fast to compute. Once the trees are trained an incoming patch is classified by adding up the probability distributions of the leaf nodes that were reached and by identifying the class with the maximal probability. As shown in [82], training can be performed incrementally which simplifies their use on arbitrary well-textured 3D objects. However, it is not real-time and therefore has to be performed offline, which is problematic for applications such as SLAM. [111] replaced the Randomized Trees by a simpler list structure and binary values instead of a probability distribution. These modifications allow it to learn new features online in real-time. Another approach that enables online feature learning in real-time was proposed in [34]: it is based on the boosting algorithm presented in [33] and can adapt to changing scenes. Recently, Calonder *et al.* presented the concept of *Signatures* [14] which uses Randomized Trees, pre-trained on a random set of images, and exploits the returned probability distributions for an image patch as feature vector. As such, extraction is very fast which can be even more speeded up by using the bitwise representation of the binary comparisons directly [15].

Recently, Taylor *et al.* [101, 102] introduced another learning-based approach that is closer to the approach presented in this chapter. It is based on what is called "Histogrammed Intensity Patches" (HIP). The link with our approach is that the HIPs are reminiscent of our "mean patches" used in Gepard: Each keypoint in the database is represented by a set of HIPs, each of them computed over a small range of poses. For fast indexing, an HIP is a binarized histogram of the intensities of a few pixels around the keypoint. However, while this is in theory possible, estimating the keypoint pose has not been evaluated nor demonstrated, and this method provides only the keypoint identities, too.

Another work related to our two methods is [71], which exploits the perspective transformation of patches centered on landmarks in SLAM applications. However, it is still very dependent on the tracking prediction to match the landmarks and to retrieve their transformations, while we do not need any prior on the pose. Moreover, in [71], these transformations are recovered using a Jacobian-based method while, in our case, a linear predictor can be trained very efficiently for faster convergence.

In short, to the best of our knowledge, there is no method in the literature that attempts to reach the exact same goal as ours. Our two methods can estimate quickly and accurately the pose of keypoints in a database, thanks to a learning-based approach.



(a) (b)

Figure 2.2: Examples of patches used for classification. (a) To estimate the keypoint identity, patches from the same keypoint are grouped in a single class. (b) To estimate the patch transformation, several classes for different transformations are created for each keypoint in the database.

2.2 Leopar: A Classifier Favoring runtime Performance

In this section we present the first stage of Leopar. It provides the identity and an approximate pose of a keypoint, given an image patch centered on this keypoint, using an extension of [80]. The second stage which consists of the keypoint pose refinement and checking steps is common to our two methods, and will be presented in Section 2.4.

2.2.1 Finding the Keypoint's Identity

Leopar first recognizes the keypoint to which the patch corresponds to by using the Ferns classifier presented in [80]. Ferns are trained with patches centered on the keypoints in the database and seen under different viewing conditions as in Fig. 2.2(a). Formally, for a given patch $\mathcal{P}$ centered on a keypoint we want to recognize, it estimates:

$$\widehat{id} = \underset{id}{\operatorname{argmax}}\, P(Id = id \mid \mathcal{P}) \,, \tag{2.1}$$

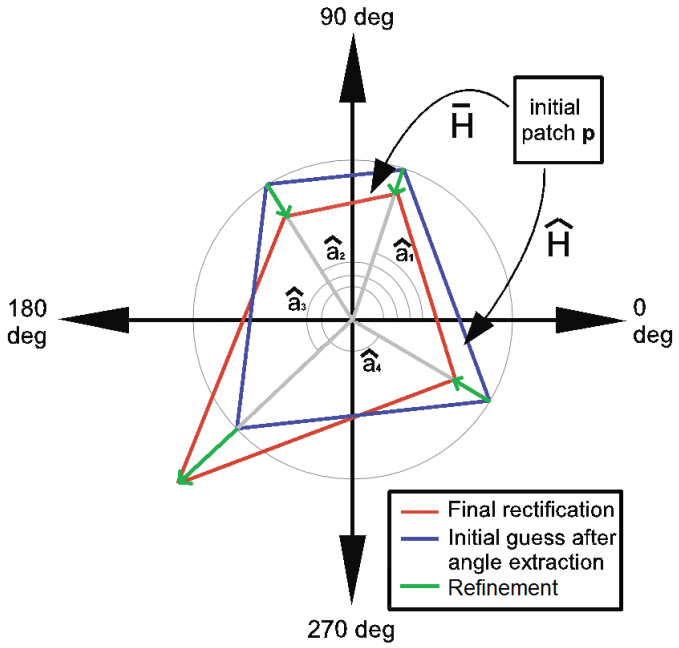


Figure 2.3: A first estimate of the patch transformation is obtained using a classifier that provides the values of the angles $\mathbf{a}_i$ defined as the angles between the lines that go through the patch center and each of the four corners. We also tried to recover the length of these four lines by the classifier, however, this gave no good results. The final rectification is then performed by applying the approach of [52] as described in Sec. 2.4.1 using the first estimate of the patch transformation as initial guess.

where Id is a random variable representing the identity of the keypoint. The identity is simply the index of the corresponding keypoint in the database. As described in [80], the classifier represents the patch $\mathcal{P}$ as a set of simple image binary features that are grouped into subsets, and Id is estimated following a semi-Naive Bayesian scheme that assumes the feature subsets are independent. This classifier is usually able to retrieve the patch identity Id under scale, perspective and lighting variations.

2.2.2 Discretizing and Estimating the Keypoint's Pose

Once Id is estimated, our objective is then to get an estimate of the transformation of the patch around the keypoint. Because we also want to use a Fern classifier for that due to its great efficiency, we first need a way to quantize the transformations. We tried various approaches, and the best results were obtained with the parametrization described in Fig. 2.3. It is made of the four anti-clockwise arranged angles $\mathbf{a}_i$ between the horizontal axis and the semi-lines going from the patch center and passing through the patch corners. Thus, we parametrize a quadrangle which represents an initial homography. Each angle is quantized into 36 values (so each angle increases in 10 degree steps), and to both reduce the required amount of memory and increase the speed at runtime, we estimate each angle independently as:

$$\forall i = 1 \dots 4 \quad \widehat{\mathbf{a}}_i = \underset{\mathbf{a}_i}{\operatorname{argmax}} \, P(A_i = \mathbf{a}_i \mid Id = id, \mathcal{P}) \,, \tag{2.2}$$

using four Ferns classifiers specific to the keypoint of identity Id.

Leopar will be evaluated in Section 2.5. Before that, we present our second method and their common second stage.

2.3 Gepard: A Classifier Favoring Real-Time Learning and Robustness

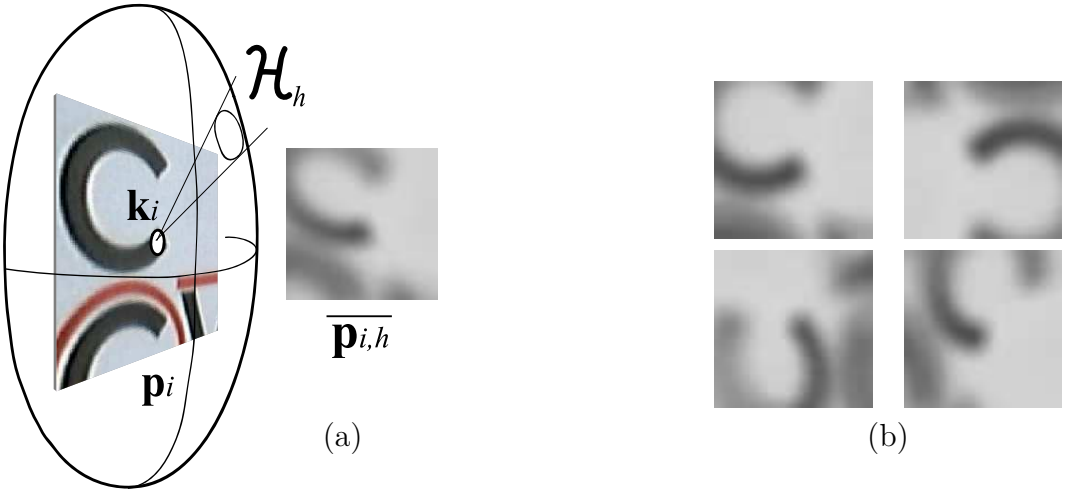


Figure 2.4: The Gepard descriptor. (a) For a feature point $\mathbf{k}_i$, this descriptor is made of a set of mean patches $\{\overline{\mathbf{p}_{i,h}}\}$, each computed for a small range of poses $\mathcal{H}_h$ from a reference patch $\mathbf{p}_i$ centered on the feature point $\mathbf{k}_i$. (b) Some other mean patches for the same feature point. The examples shown here are full resolution patches for visibility, in practice we use downscaled patches.

Leopar was designed for runtime speed, and it requires a slow training phase: It takes about 1 second for Leopar to learn one keypoint, and this makes it unsuitable for online applications like SLAM. We therefore propose a second method — called Gepard in the following, which is slower at runtime but can learn new keypoints much faster. It is also more robust.

2.3.1 Finding the Keypoint's Identity and Pose

As depicted by Fig. 2.4, our starting idea to estimate the keypoint's identity and pose is to first build a set of "mean patches". Each mean patch is computed as the average of the keypoint appearance when the pose varies in the neighborhood of a reference pose. In praxis, two neighboring poses are approximately 15 degree apart. Then, we can use nearest-neighbor classification: We assign to an incoming keypoint the pose of the most similar mean patch as a first estimate of its pose.

Of course, computing a single mean patch over the full range of poses would result in a blurred irrelevant patch. Because we compute these mean patches over only a small range of poses, they are meaningful and allow for reliable recognition. Another advantage

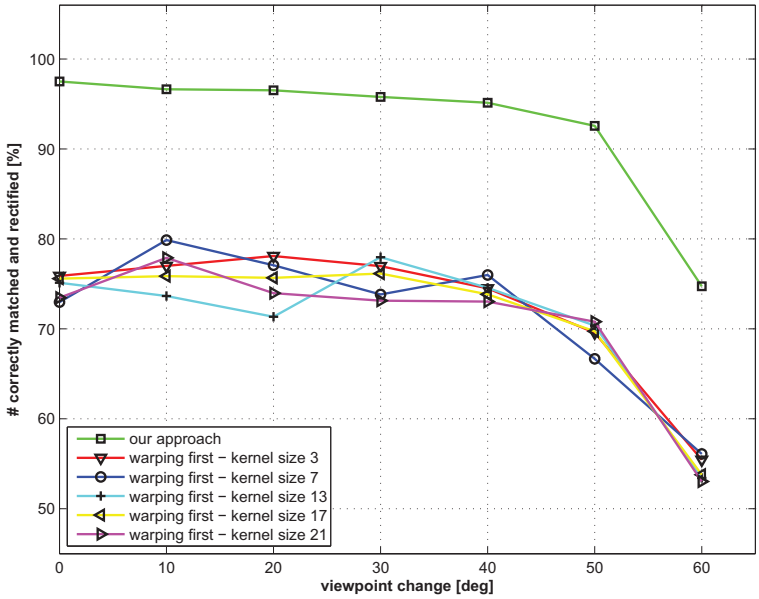


Figure 2.5: Computing a set of our mean patches clearly outperforms simple blurring of a set of warped patches. Different Gaussian smoothing kernels were tried and we show that our approach improves the matching rate constantly of about 20%.

is that they are robust to image noise and blur. As the mean patches in Fig. 2.4 look like blurred image patches, one may wonder if using a uniform blur on warped patches would be enough. The answer is no: As Fig. 2.5 shows, using mean patches substantially improves the matching rate by about 20% compared to using blurred warped patches.

Compared to standard approaches [68], we do not have to extract an estimate of the keypoint's pose, nor compute a descriptor for the incoming points, and that makes the approach faster at runtime. The set of means that characterizes a keypoint in the database can be seen as a descriptor, which we refer to as a "one-way descriptor" since it does not have to be computed for the new points. This approach increases the number of vectors that have to be stored in the database, but fortunately, efficient methods exist for nearest-neighbor search in large databases of high-dimensional vectors [7].

More formally, given a keypoint $\mathbf{k}$ in a reference image, we compute a set of mean patches $\{\overline{\mathbf{p}_h}\}$ where each mean $\overline{\mathbf{p}_h}$ can be expressed as

$$\overline{\mathbf{p}_h} = \int_{H \in \mathcal{H}_h} \mathbf{w}(\mathcal{P}^*, H) p(H) dH, \tag{2.3}$$

where

- H represents a pose, in our case a homography,
- $\mathcal{P}^*$ is the *reference patch*, the image patch centered on the keypoint $\mathbf{k}$ in the reference image. To be robust to light changes, the pixel intensities in $\mathcal{P}^*$ are normalized so their sum is equal to 0, and their standard deviation to 1,

- $\mathbf{w}(\mathcal{P}, H)$ returns the patch $\mathcal{P}$ seen under pose H,
- $p(H)$ is the probability that the keypoint will be seen under pose H, and
- $\mathcal{H}_h$ is a range of poses, as represented in Fig. 2.4. The $\mathcal{H}_h$'s are defined so that they cover small variations around a fixed pose H_h but together they span the set of all possible poses $\bigcup_h \mathcal{H}_h$.

In practice, the integrals in Eq. (2.3) are replaced by finite sums, the distribution over the transformations H is assumed uniform and expression (2.3) becomes

$$\overline{\mathbf{p}_h} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\mathcal{P}^*, H_{h,j}), \tag{2.4}$$

where the $H_{h,j}$ are N poses sampled from $\mathcal{H}_h$.

Once the means $\overline{\mathbf{p}_h}$ are computed, it is easy to match incoming keypoints against the database, and get a coarse pose. Given the normalized patch $\mathbf{p}$ centered on such an incoming point with assumed identity $\widehat{id}$, its coarse pose $\widehat{\mathbf{H}}_{i=\widehat{id},h=\hat{h}}$ indexed by $\hat{h}$ is obtained by finding:

$$\hat{h} = \underset{i=\widehat{id},h}{\operatorname{argmax}} \, \mathbf{p}^\top \cdot \overline{\mathbf{p}_h} \, . \tag{2.5}$$

However, computing the $\overline{\mathbf{p}_h}$ using Eq. (2.4) is very inefficient because it would require the generation of too many samples $\mathbf{w}(\mathcal{P}^*, H_{h,j})$. In practice, to reach decent results, we have to use at least 300 samples, and this takes about 1.1 seconds to generate[1], which was not acceptable for interactive applications. We show below that the mean patches can actually be computed very quickly, independent of the number of samples used.

2.3.2 Fast Computation of the Mean Patches

We show here that we can move most of the computation cost of the mean patches to an offline stage, so that computing the mean patches at runtime can be done very efficiently. To this end, we first approximate the reference patch $\mathcal{P}^*$ as:

$$\mathcal{P}^* \approx \overline{\mathcal{V}} + \sum_{l=1}^{L} \alpha_l \mathcal{V}_l \tag{2.6}$$

where $\overline{\mathcal{V}}$ and the $\mathcal{V}_l$'s are respectively the mean and the L first principal components of a large set of image patches centered on keypoints. The α_l are therefore the coordinates of $\mathcal{P}^*$ in this eigenspace, and can be computed as $\alpha_l = \mathcal{V}_l^\top \mathcal{P}^*$.

Computing $\overline{\mathcal{V}}$ and the $\mathcal{V}_i$'s takes time but this can be done offline once and for all. Because we consider normalized patches, the mean $\overline{\mathcal{V}}$ is equal to 0, and Eq. (2.3) becomes

$$\overline{\mathbf{p}_h} \approx \frac{1}{N} \sum_{j} \mathbf{w}(\sum_{l=1}^{L} \alpha_l \mathcal{V}_l, H_{j,h}) \, . \tag{2.7}$$

[1] All times given in this chapter were reached on a on a standard notebook (Intel(R) Centrino Core(TM)2 Duo with 2.6GHz and 3GB RAM and an NVIDIA quadro FX3600M with 512MB).

This expression can be simplified by using the fact that the warping function $\mathbf{w}(\cdot, H)$ is a linear function: Warping is mostly a permutation of the pixel intensities between the original patch and the patch after warping, and therefore a linear transformation. This fact was used, for example, in [10] for transformation-invariant data modeling. In our case, because we use perspective transformations, parts that are not visible in the original one could appear in the generated patch. To solve this issue, we simply take the original patch larger than the warped patch, and large enough so that there are never parts in the warped patch which were not present in the original patch. The function $\mathbf{w}(., H)$ then becomes a permutation followed by a projection, and this composition remains a linear transformation.

Thanks to this property, Eq. (2.7) simplifies easily:

$$\begin{aligned}
\overline{\mathbf{p}_h} &\approx \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}(\sum_{l=1}^{L}\alpha_l\mathcal{V}_l, H_{j,h}) && (2.8)\\
&= \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{l=1}^{L}\alpha_l\mathbf{w}(\mathcal{V}_l, H_{j,h})\right) && (2.9)\\
&= \sum_{l=1}^{L}\frac{\alpha_l}{N}\sum_{j=1}^{N}\mathbf{w}(\mathcal{V}_l, H_{j,h}) && (2.10)\\
&= \sum_{l=1}^{L}\alpha_l\overline{\mathbf{v}_{l,h}} && (2.11)
\end{aligned}$$

where the $\overline{\mathbf{v}_{l,h}}$'s are patches obtained by warping the eigenvectors $\mathcal{V}_l$ under poses in $\mathcal{H}_h$:

$$\overline{\mathbf{v}_{l,h}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}(\mathcal{V}_l, H_{j,h})\,. \qquad (2.12)$$

Like the $\mathcal{V}_l$, the $\overline{\mathbf{v}_{l,h}}$'s patches can be computed offline. The number of samples N therefore does not matter for the runtime computations, and we can use a very large number.

In summary, when we have to insert a new keypoint in the database, we simply have to project it into the eigenspace, and compute its associated mean patches by linear combinations. The complete process can be written in matrix form:

$$\begin{aligned}
\boldsymbol{\alpha} &= \mathbf{P}_{\mathrm{PCA}}\mathcal{P}^*\,, \text{ and} && (2.13)\\
\forall h\ \ \overline{\mathbf{p}_h} &= \mathbf{V}_h\boldsymbol{\alpha}\,, && (2.14)
\end{aligned}$$

where $\mathcal{P}^*$ is the reference patch seen as a vector, $\mathbf{P}_{\mathrm{PCA}}$ the projection matrix into the eigenspace, $\boldsymbol{\alpha}$ the vector of the α_l coefficients, and the $\mathbf{V}_h$'s are matrices. The rows of $\mathbf{P}_{\mathrm{PCA}}$ are the $\mathcal{V}_l$ vectors, and the columns of the $\mathbf{V}_h$'s matrices are the $\overline{\mathbf{v}_{l,h}}$ vectors. This approach saves significant computation time with respect to the naive way to compute Eq. (2.7).

To speed-up computation even further, mostly in the evaluation of the similarity between an incoming patch $\mathbf{p}$ and a mean patch $\overline{\mathbf{p}_h}$ as in Eq. (2.5), we downsample these patches. We keep the reference patch $\mathcal{P}^*$ and the eigenvectors $\mathcal{V}_l$ at the original resolution

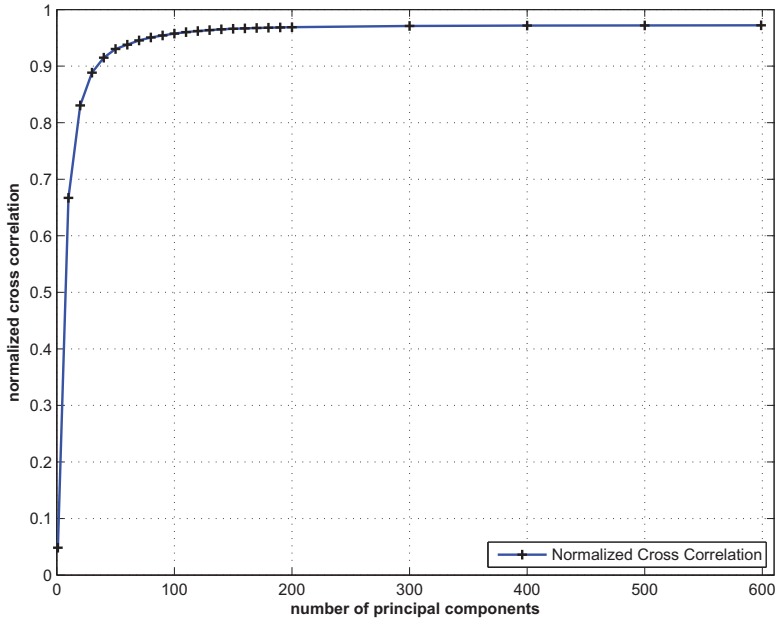


Figure 2.6: Normalized cross-correlation between approximated mean patches and their exact computation as a function of the number of principal components. The values are averaged over 100 patches from the Graffiti image set [68]. In this experiment, the patches are 120×120 pixels but we need only a small percentage of the principal components to get a good approximation.

to avoid losing important details. Downscaling is applied only on the result of the sum in Eq. (2.12), which is performed off-line anyway.

To handle scale efficiently, we compute the mean patches only on one scale level. Then, the evaluation of the similarity in Eq. (2.5) is done several times for each mean patch with different versions of the incoming patch $\mathbf{p}$, each version extracted at a different scale.

In practice, as shown in Fig. 2.6, we can keep only a small percentage of the first principal components and still get a good approximation of the mean patches. The graph of Fig. 2.7 shows that using only $L = 150$ principal components and 192 mean patches over 3 scales—giving a total of 576 mean patches—already gives reasonably good results. The computation time is then 15 milliseconds for patches downsampled from 71×71 to 12×12 including the computation of $\boldsymbol{\alpha}$ while using Eq. (2.4) directly takes 1.1 seconds. By using the GPU to compute the matrix form expressions in Eqs. (2.13) and (2.14), we can reduce the processing time even further to only 5.5 milliseconds[1].

2.3.3 Discretizing the Pose Space for Gepard

The quantized pose space used for Leopar represented in Fig. 2.3 was designed to keep the number of discrete homographies small because a finer discretization does not improve the matching score while slowing down the recognition and increasing the required memory. In Gepard, however, we can afford a finer discretization and that results in better recognition rate (see Fig. 2.8).

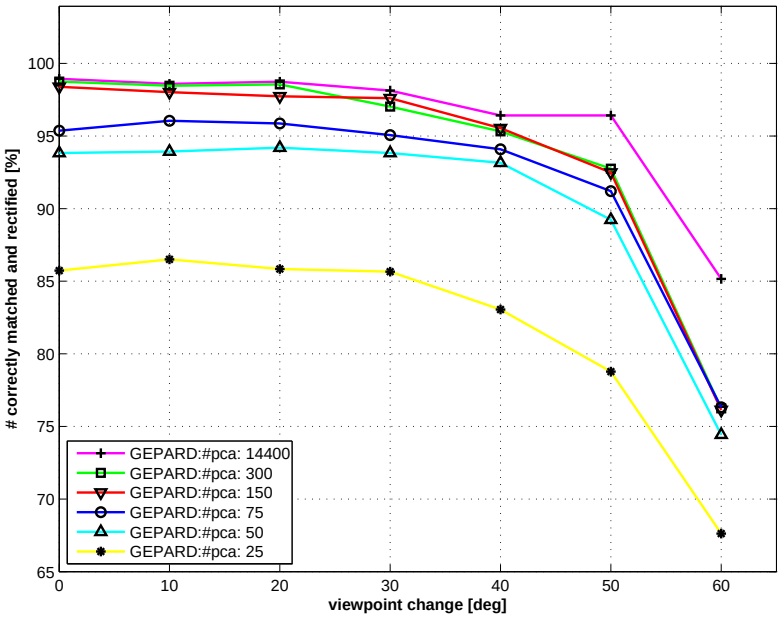


Figure 2.7: Recognition rates as a function of the viewpoint angle for different number of principal components. The values are averaged over 100 patches from the Graffiti image set [68]. We use synthesized images to generate more views than the original Graffiti image sequence. In this experiment, the patches are 120×120 pixels but using only 150 principal components over 14400 gives results comparable to the full method up to 40 degrees and is more than 70 times faster.

This discretization is done based on the formula:

$$\mathbf{H} = \mathbf{K}\left(\Delta\mathbf{R} + \frac{\delta\mathbf{t}\cdot\mathbf{n}^\top}{d}\right)\mathbf{K}^{-1}\,, \tag{2.15}$$

which is the expression of the homography $\mathbf{H}$ relating two views of a 3–D plane, where $\mathbf{K}$ is the matrix of the camera internal parameters, $[\mathbf{n}^\top,\ d]^\top$ the parameters of the plane in the first view, and $\Delta\mathbf{R}$ and $\delta\mathbf{t}$ the camera displacement between the two views. For simplification, we assume that we have a frontal view of the reference patches.

We first tried discretizing the motion between the views by simply discretizing the rotation angles around the three axes. However, for the nearest-neighbor classification to work well, it must be initialized as close as possible to the correct solution. We provide a better solution: as shown by the left image of Fig. 2.9, we found that the vertices of (almost) regular polyhedrons provide a more regular sampling that is useful to discretize the angle of the second view includes with the patch plane in Eq. (2.15).

However, there exists only a few convex regular polyhedrons —the Platonic solids— with the icosahedron the one with the largest number of vertices, 12. As the right image of Fig. 2.9 illustrates, we obtain a finer sampling by recursively substituting each triangle into four almost equilateral triangles. The vertices of the created polyhedron give us the two out-of-plane rotation angles for the sampled pose, that is around the x- and y-axes of Fig. 2.9. We discretize the in-plane rotation angle with 10° steps to cover the 360° range.

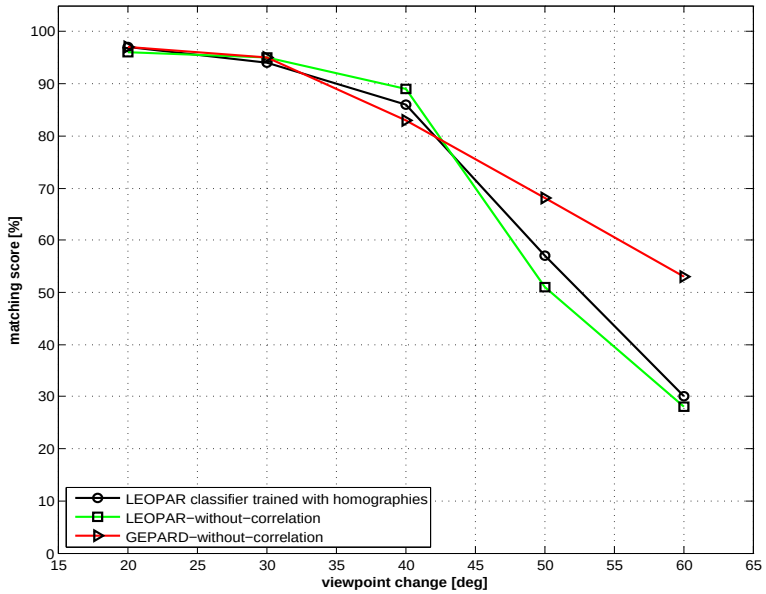


Figure 2.8: We tried to use in Leopar the homography discretization used in Gepard. As the graph shows, it does not result in any improvement in terms of matching score compared to the discretization described in Fig. 2.3. Since it requires more computation time and memory because the number of discrete homographies is larger, we used the method of Fig. 2.3 for Leopar. The experiment was performed on the standard Graffiti test set [68].

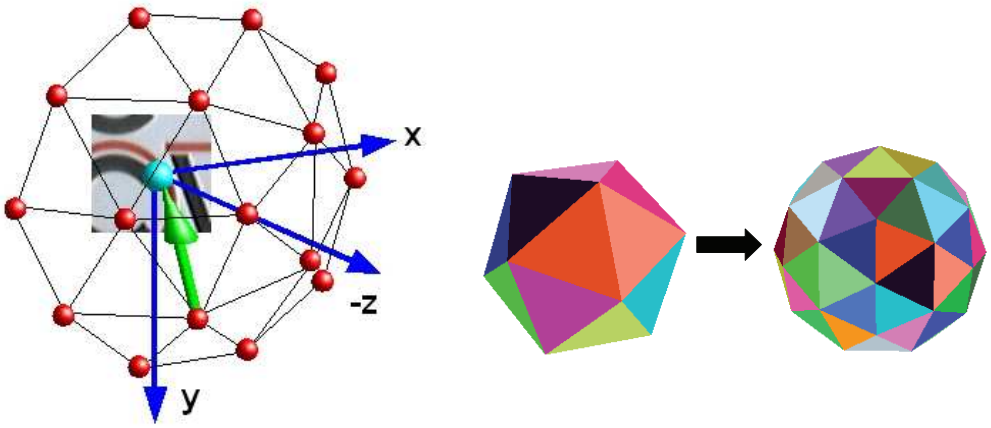


Figure 2.9: Pose space sampling using almost regular polyhedrons. Left: The red dots represent the vertices of an (almost) regular polyhedron generated by our recursive decomposition and centered on a planar patch. The sampled directions of views are given by vectors starting from one of the vertices and pointing toward the patch center. The green arrow is an example of such a vector. Right: The initial icosahedron and the result of the first triangle substitution.

2.4 Pose Refinement and Final Check

Having the output of the first stage of Leopar or Gepard, the keypoint's identity and approximate pose, we want to compute a better estimate of the pose in the form of a homography without quantization. This refinement is based on linear regression, and we show how the linear predictors can be computed incrementally and how we can improve the training speed. The refinement is followed by a final check, to suppress keypoints that were incorrectly recognized.

2.4.1 Linear Prediction for Refinement

The homography $\widehat{\mathbf{H}}$ computed in the first stage is an initial estimate of the true homography $\overline{\mathbf{H}}$. We use the method presented in [52] which is based on linear predictors to obtain the parameters $\tilde{\mathbf{x}}$ of a corrective homography:

$$\tilde{\mathbf{x}} = \mathbf{B}\left(\mathbf{w}(\mathcal{P}, \widehat{\mathbf{H}}) - \mathbf{p}^*\right) , \tag{2.16}$$

where

- $\mathbf{B}$ is the matrix of our linear predictor and depends on the retrieved patch identity $\widehat{id}$;
- $\mathcal{P}$ is the patch in the incoming image, centered on the keypoint to recognize;
- $\mathbf{w}(\mathcal{P}, \widehat{\mathbf{H}})$ is the patch $\mathbf{p}$ warped by the current estimate $\widehat{\mathbf{H}}$ and downscaled for efficiency. Note that we do not actually warp the patch, we simply warp back the sampled pixel locations;
- $\mathbf{p}^*$ is the reference patch $\mathcal{P}^*$ after downscaling. $\mathcal{P}^*$ is the image patch centered on the keypoint $\widehat{id}$ in a reference image as in Section 2.3.

This equation gives us the parameters $\tilde{\mathbf{x}}$ of the incremental homography that updates $\widehat{\mathbf{H}}$ to produce a better estimate of the true homography $\overline{\mathbf{H}}$:

$$\widehat{\mathbf{H}} \longleftarrow \widehat{\mathbf{H}} \circ \mathbf{H}(\tilde{\mathbf{x}}) . \tag{2.17}$$

For more accuracy, we iterate Eqs. (2.16) and (2.17) using a series of linear predictors $\mathbf{B}_i$, each matrix being dedicated to smaller errors than its predecessor: Applying these matrices successively remains fast and gives a more accurate estimate than with a single level. In order to do an ultimate refinement the ESM algorithm [8] can be applied.

In practice, our vectors $\mathbf{w}(\mathcal{P}, \widehat{\mathbf{H}})$ and $\mathbf{p}^*$ contain the intensities at locations sampled on a regular grid of 13×13 over image patches of size 75×75 pixels, and we normalize them to be robust to light changes. We parametrize the homographies by the 2D locations of the patch's four corners since this parametrization has proven to be more stable than others in [2]. In practice, for each patch we train four to ten[2] matrices $\mathbf{B}$ with different

ranges of variation from coarse to fine, using downscaled patches of $13 \times 13 = 169$ pixels and 300 to 5000 [2] training samples.

For online applications, the $\mathbf{B}$'s matrices must be computed for each new keypoint inserted in the database at runtime. Thus, learning the $\mathbf{B}_i$'s consists of computing a set of pairs made of small random transformations H_s and the corresponding warped patches $\mathbf{w}(\mathcal{P}, H_s^{-1})$ must be fast enough to fulfill the real-time constraints as discussed below. In order to do so we precompute the transformations H_s and the warped pixel locations in order to obtain very quickly the $\mathbf{w}(\mathcal{P}, H_s^{-1})$ patches at runtime for an arbitrary incoming feature point. The whole process thus can be speed up to 29 milliseconds [1] using 300 samples and four $\mathbf{B}$ matrices.

2.4.2 Incrementally Learning the Linear Predictor

For some applications it is desirable to improve the tracking by performing online learning. Since the classification steps in Leopar as well as in Gepard can easily be extended to do online learning, we only have to concentrate on the linear predictors.

The linear predictor $\mathbf{B}$ in Eq. (2.16) can be computed as the pseudo-inverse of the analytically derived Jacobian matrix of a correlation measure [3, 8]. However, the hyperplane approximation [52] computed from several examples yields a much larger region of convergence. The matrix $\mathbf{B}$ is then computed as:

$$\mathbf{B} = \mathbf{X}\mathbf{D}^\top \left(\mathbf{D}\mathbf{D}^\top\right)^{-1} , \tag{2.18}$$

where $\mathbf{X}$ is a matrix made of $\mathbf{x}_i$ column vectors, and $\mathbf{D}$ a matrix made of column vectors $\mathbf{d}_i$. Each vector $\mathbf{d}_i$ is the difference between the reference patch $\mathbf{p}^*$ and the same patch after warping by the homography parametrized by $\mathbf{x}_i$: $\mathbf{d}_i = \mathbf{w}(\mathbf{p}, \mathbf{H}(\mathbf{x}_i)) - \mathbf{p}^*$.

Eq. (2.18) requires all the pairs $(\mathbf{x}_i, \mathbf{d}_i)$ to be simultaneously available. If it is applied directly, this prevents incremental learning but this can be fixed. Suppose that the matrix $\mathbf{B} = \mathbf{B}_n$ is already computed for n examples, and then a new example $(\mathbf{x}_{n+1}, \mathbf{d}_{n+1})$ becomes available. We want to update the matrix $\mathbf{B}$ into the matrix $\mathbf{B}_{n+1}$ that takes into account all the $n+1$ examples. Let us introduce the matrices $\mathbf{Y}_n = \mathbf{X}_n\mathbf{D}_n^\top$ and $\mathbf{Z}_n = \mathbf{D}_n\mathbf{D}_n^\top$. We then have:

$$\begin{aligned}
\mathbf{B}_{n+1} &= \mathbf{Y}_{n+1}\mathbf{Z}_{n+1}^{-1} \\
&= \mathbf{X}_{n+1}\mathbf{D}_{n+1}^\top \left(\mathbf{D}_{n+1}\mathbf{D}_{n+1}^\top\right)^{-1} \\
&= [\mathbf{X}_n|\mathbf{x}_{n+1}][\mathbf{D}_n|\mathbf{d}_{n+1}]^\top \left([\mathbf{D}_n|\mathbf{d}_{n+1}][\mathbf{D}_n|\mathbf{d}_{n+1}]^\top\right)^{-1} \\
&= \left(\mathbf{X}_n\mathbf{D}_n^\top + \mathbf{x}_{n+1}\mathbf{d}_{n+1}^\top\right)\left(\mathbf{D}_n\mathbf{D}_n^\top + \mathbf{d}_{n+1}\mathbf{d}_{n+1}^\top\right)^{-1} \\
&= \left(\mathbf{Y}_n + \mathbf{x}_{n+1}\mathbf{d}_{n+1}^\top\right)\left(\mathbf{Z}_n + \mathbf{d}_{n+1}\mathbf{d}_{n+1}^\top\right)^{-1}
\end{aligned} \tag{2.19}$$

[2] Depending on the application. Using more $\mathbf{B}_i$ matrices or more training samples improves the accuracy but the computation time for this step increases linearly with the number of matrices and exponentially with the number of training samples.

where $\mathbf{x}_{n+1}$ and $\mathbf{d}_{n+1}$ are concatenated to $\mathbf{X}_n$ and $\mathbf{D}_n$ respectively to form $\mathbf{X}_{n+1}$ and $\mathbf{D}_{n+1}$. Thus, by only storing the *constant size* matrices $\mathbf{Y}_n$ and $\mathbf{Z}_n$ and updating them as:

$$\mathbf{Y}_{n+1} \longleftarrow \mathbf{Y}_n + \mathbf{x}_{n+1}\mathbf{d}_{n+1}^\top \tag{2.20}$$
$$\mathbf{Z}_{n+1} \longleftarrow \mathbf{Z}_n + \mathbf{d}_{n+1}\mathbf{d}_{n+1}^\top\,, \tag{2.21}$$

it becomes possible to incrementally learn the linear predictor without storing the previous examples, and allows for an arbitrary large number of examples.

Since the computation of $\mathbf{B}$ has to be done for many locations in each incoming image and $\mathbf{Z}_n$ is a large matrix in practice, we need to go one step further in order to avoid the computation of $\mathbf{Z}_n^{-1}$ at every iteration. We apply the Sherman-Morrison formula to $\mathbf{Z}_{n+1}^{-1}$ and we get:

$$\begin{aligned}\mathbf{Z}_{n+1}^{-1} &= \left(\mathbf{Z}_n + \mathbf{d}_{n+1}\mathbf{d}_{n+1}^\top\right)^{-1} \\ &= \mathbf{Z}_n^{-1} - \frac{\mathbf{Z}_n^{-1}\mathbf{d}_{n+1}\mathbf{d}_{n+1}^\top\mathbf{Z}_n^{-1}}{1 + \mathbf{d}_{n+1}^\top\mathbf{Z}_n^{-1}\mathbf{d}_{n+1}}\,.\end{aligned} \tag{2.22}$$

Therefore, if we store $\mathbf{Z}_n^{-1}$ instead of $\mathbf{Z}_n$ itself, and update it using Eq. (2.22), no matrix inversion is required anymore, and the computation of matrix $\mathbf{B}_{n+1}$ becomes very fast.

2.4.3 Correlation-based Hypothesis Selection and Verification

In Gepard, for each possible keypoint identity i, we use the method explained above to estimate a fine homography $\widehat{\mathbf{H}}_{i,\text{final}}$. Thanks to the high accuracy of the retrieved transformation, we can select the correct pair of keypoint identity i and pose $\widehat{\mathbf{H}}_{i,\text{final}}$ based on the normalized cross-correlation between the reference patch $\mathcal{P}_i^*$ and the warped patch $\mathbf{w}(\mathcal{P}, \widehat{\mathbf{H}}_{i,\text{final}})$ seen under pose $\widehat{\mathbf{H}}_{i,\text{final}}$. The selection is done by

$$\underset{i}{\operatorname{argmax}}\, \mathcal{P}_i^{*\top} \cdot \mathbf{w}(\mathcal{P}, \widehat{\mathbf{H}}_{i,\text{final}})\,. \tag{2.23}$$

In Leopar, the keypoint identity i is directly provided by the Ferns classifier.

Finally, we use a threshold $\tau_{\text{NCC}} = 0.9$ in order to remove wrong matches:

$$\mathbf{w}(\mathcal{P}, \widehat{\mathbf{H}}_{i,\text{final}})^\top \cdot \mathcal{P}_i^* > \tau_{\text{NCC}}\,. \tag{2.24}$$

Thus, each patch $\mathbf{w}(\mathcal{P}, \widehat{\mathbf{H}}_{i,\text{final}})$ that gives the maximum similarity score, which exceeds τ_{NCC} at the same time, yields an accepted match.

2.5 Experimental Validation

Here, we compare our approach against affine region detectors on the Graffiti image set from [68] towards robustness and accuracy. Although affine region detectors are computed without having a priori information about the target patch, we chose them because they

are in our opinion the closest state-of-the-art approaches. This is because affine region detectors are based on local regions only and return a pose estimation of the underlying structure. While we could have added some pose refinement and final verification to them — similar to our approaches — we did not perform these extra steps since they are not part of the official methods.

At the end of this section, we also evaluate the performance of our algorithms with respect to training time, running time and memory consumption. For each experiment we give a detailed discussion about the specific advantages of each of our two methods.

2.5.1 Evaluation on the Graffiti Image Set

We first built a database of the most stable 100 Harris keypoints from the first image of the Graffiti set [68]. We used Harris keypoints because they have proven their robustness under different viewpoints in various works e.g. in [68]. However, we could easily use other keypoints like [87, 63, 95, 57]. In this case, one has to make sure that a characteristic of the keypoint type is to be surrounded by enough texture for doing the matching and the refinement.

These keypoints were found by synthetically rendering the image under many random transformations, adding artificial image noise and extracting Harris keypoints. We then kept the 100 keypoints detected most frequently.

The Ferns classifiers in Leopar were trained with synthetic images as well, by scaling and rotating the first image for changes in viewpoint angle up to 65 degrees and adding noise. In the case of Gepard only the first image is needed. We then extracted Harris keypoints in the other images of the set, and ran Leopar and Gepard to recognize them and to estimate their poses.

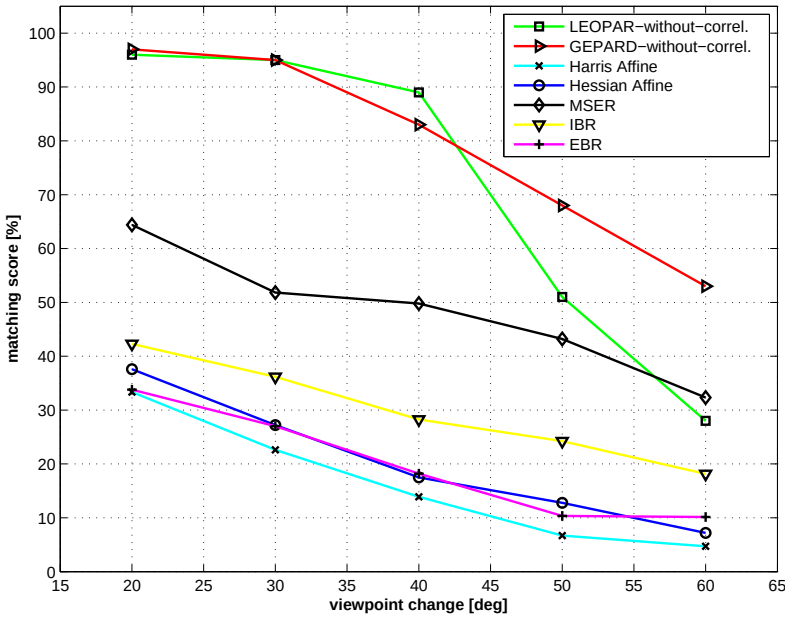
We also ran the different region detectors over the set of images and matched the regions in the first image against the regions in the other images using the SIFT descriptor computed on the rectified regions.

2.5.1.1 Robustness

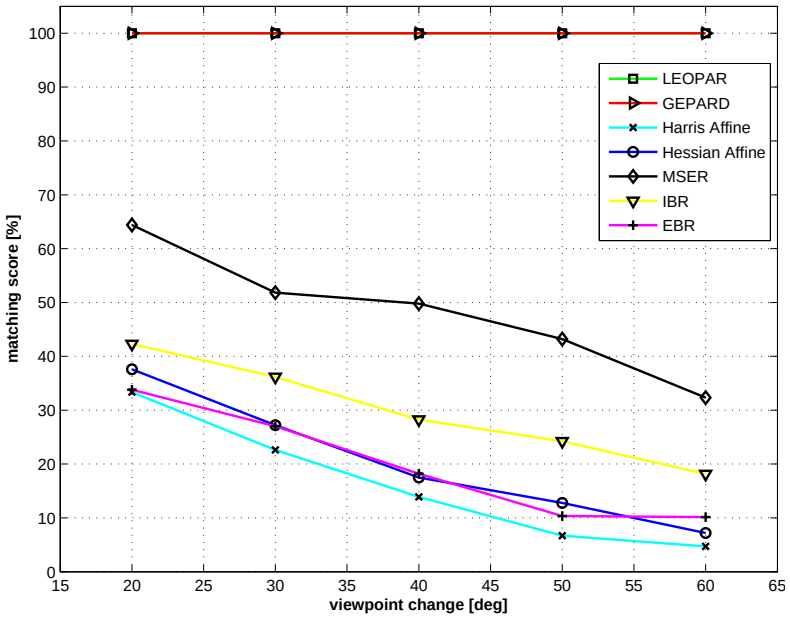
In Fig. 2.10, we compare the matching scores for the different methods. The matching score is computed as the ratio between the number of correct matches and the smaller number of regions detected in one of the two images as defined in [68]. Two regions are said to be correctly matched if the overlap error is smaller than 40%. In our case, the regions are defined as the patch surfaces warped by the retrieved transformations.

For a fair comparison, we first turned off our final check on the correlation since there is no equivalent for the affine regions in [68]. This yields the 'Leopar/Gepard without Correlation' curves. Even then, our methods perform much better, at least up to an angle of 50° in the case of Leopar. When we turn the final check on, not a single outlier is kept in this experiment. For completeness, we also show the actual number of correct matches in Fig. 2.11(a). Note, that we can manually choose how many patches we initially want to track while affine region detectors can not.

Because the Ferns classifiers consume a large amount of memory that grows linearly with the number of classes, it is difficult to handle more than 100 keypoints with Leopar.



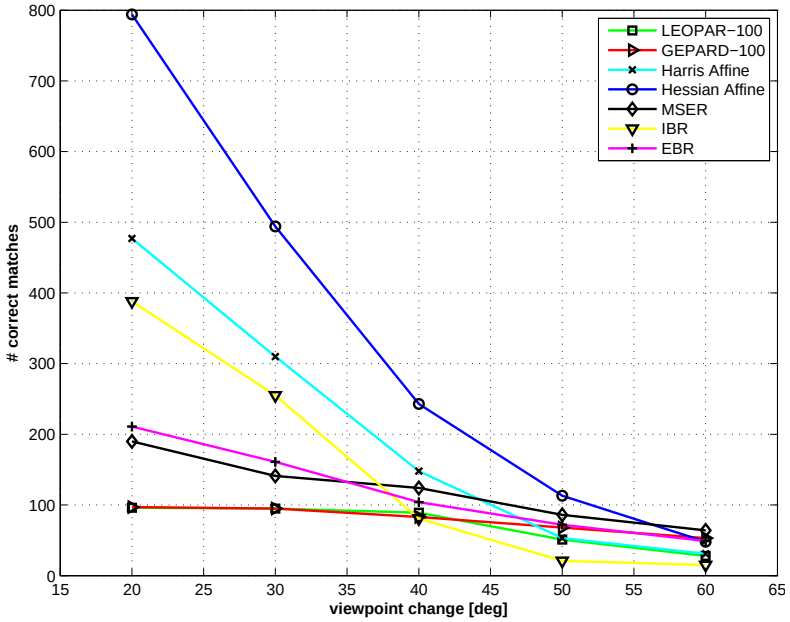
(a)



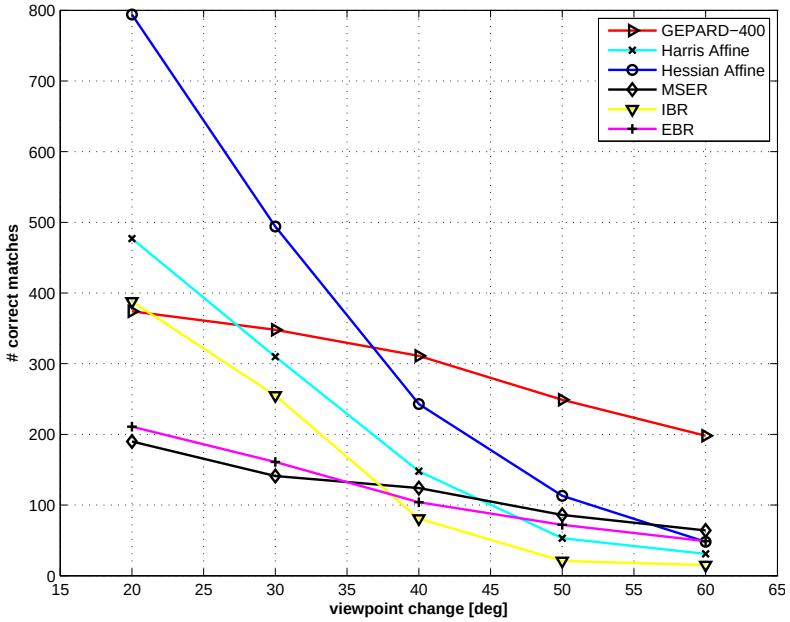
(b)

Figure 2.10: Comparing the robustness of our methods and of affine region detectors on the Graffiti image set. **(a)** Matching score as a function of the viewpoint angle. The results of Leopar and Gepard are shown with the correlation test of Section 2.4.3 disabled. Even then, our methods compare very favorably with the affine region detectors. **(b)** Same with the correlation test turned on. No outlier is produced.

Gepard in contrast uses a nearest-neighbor classifier and can deal with more keypoints. We therefore performed – as shown in Fig. 2.11(b) – the same kind of experiment as



(a)



(b)

Figure 2.11: Comparing our method against affine region detectors on the Graffiti image set. **(a)** Number of correct matches for our approach and the affine region detectors. We trained our methods on 100 keypoints in the first image. **(b)** Gepard can manage more keypoints, and for this experiment we trained it on 400 keypoints. For the largest viewpoint we still obtain a matching rate of about 50%.

in Fig. 2.11(a), but with 400 keypoints for Gepard. In that case, for the large viewpoint angles, we get more correctly matched keypoints than regions with the affine region detectors.

Gepard is also more robust to scale and perspective distortions than Leopar. In practice we found out that the limiting factor is by far the repeatability of the keypoint detector. However, once a keypoint is correctly detected, it is very frequently correctly matched at least by Gepard.

2.5.1.2 2–D Accuracy

In Figs. 2.13(a)-(d), we compare the 2–D accuracy for the different methods. To create these graphs, we proceed as shown in Fig. 2.12. We first fit a square tangent to the normalized region, take into account the canonical orientation retrieved by SIFT and warp these squares back with the inverse transformation to get a quadrangle. To account for different scales, we proceed as in [68]: We normalize the reference patch and the back-warped quadrangle such that the size of the reference patch is the same for all the patches.

Two corresponding regions should overlap if one of them is warped using the ground truth homography. A perfect overlap for the affine regions cannot be expected since their detectors are unable to retrieve the full perspective transformation. Since in SIFT several orientations were considered when ambiguity arises, we decided to keep the one that yields the most accurate correspondence. In the case of our method, the quadrangles are simply taken to be the patch borders after warping by the retrieved transformations.

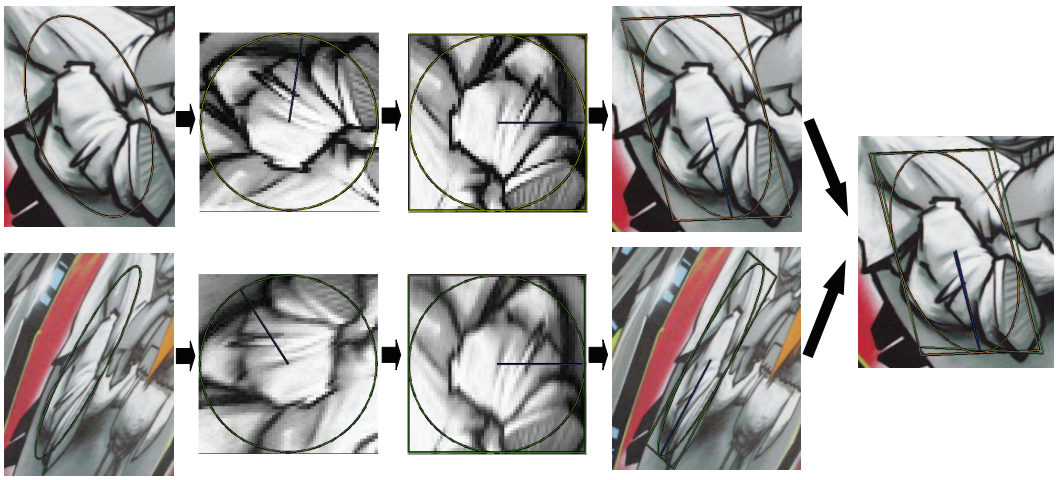
Fig. 2.13(a) evaluates the error based on the overlap between the quadrangles and their corresponding warped versions. This overlap is between 90% and 100% for our methods, about 5-10% better than MSER and about 15-25% better for all the other methods. Fig. 2.13(b) evaluates the error based on the distances between the quadrangle corners. Our methods also perform much better than the other methods. The error of the patch corner is less than two pixels in average for Leopar and slightly more for Gepard. Figs. 2.13(c) and (d) show the same comparisons, this time when taking only the best 100 regions into account. The results are very similar.

2.5.2 3–D Pose Evaluation for Low-Textured Objects

In order to demonstrate the usefulness of our approach especially for low textured objects and for outdoor environments, we did two other quantitative experiments.

For the first experiment we ran different methods to retrieve the camera pose using the power outlet of Fig. 2.14 in a sequence of 398 real images. To obtain ground truth data we attached an artificial marker next to the power outlet and tracked this marker. The marker itself was then hidden in the reference image. We consider errors on the camera center larger than 50 units as not correctly matched. For clarity we do not display these false results. For our approaches we learned only one single patch on the power outlet to track in order to emphasize the possibility to track an object with only a single patch. For learning, we created synthetically warped images of the first picture of the dataset. Gepard achieved a successful matching rate in over 98%, directly followed by Leopar with 97%.

For the affine region detectors we tried two different methods to estimate the pose of the power outlet:



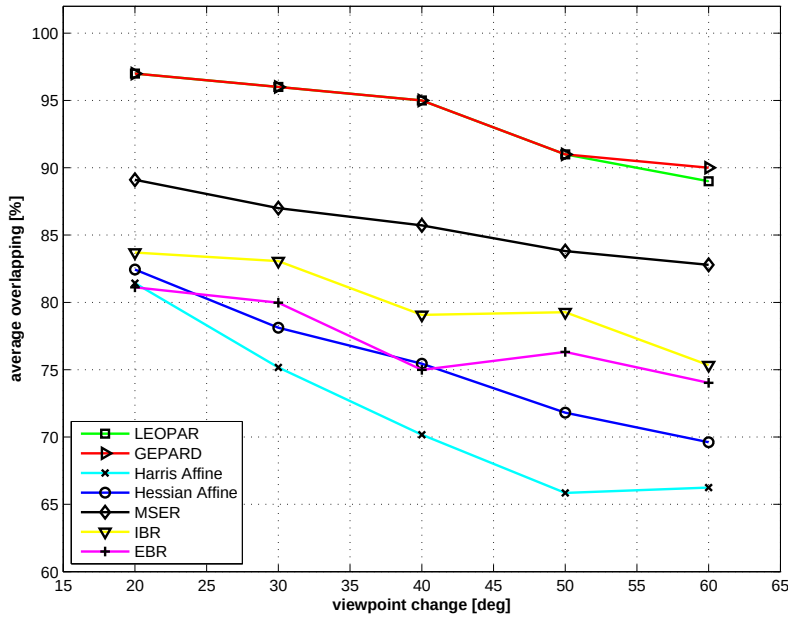
(a) (b) (c) (d) (e)

Figure 2.12: Measuring the overlapping errors and the corners distances. (a) Two matched affine regions. (b) The same regions, after normalization by their affine transformations displayed with their canonical orientations. (c) Squares are fitted to the final normalized regions. (d) The squares are warped back into quadrangles in the original images. (e) The quadrangle of the second region is warped back with the ground truth homography and compared with the quadrangle of the first image. Ideally the two quadrangles should overlap. For comparison of different region detectors we normalize the reference region to a fixed size and scale the warped region correspondingly.

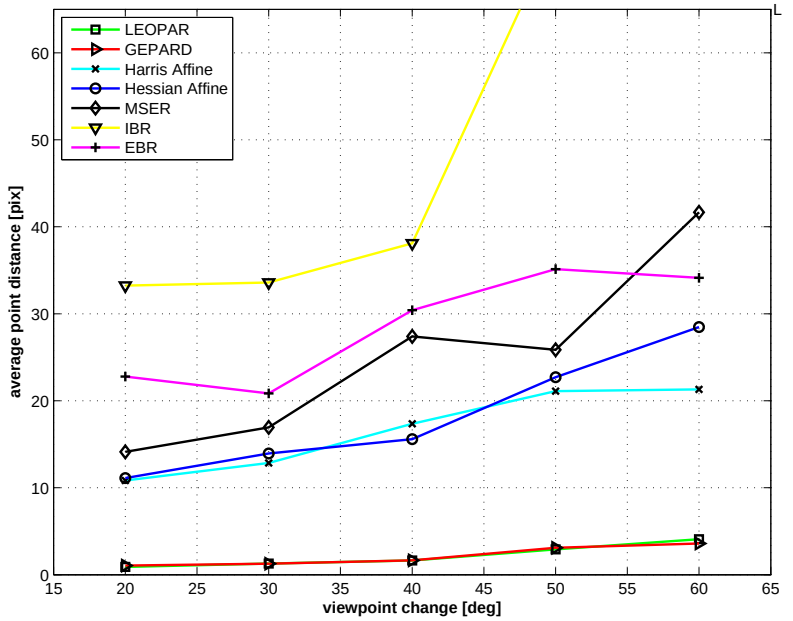
- Method A: For Fig. 2.14(a)-(c) we computed the pose from the 2–D locations of all the correctly matched affine regions. The correct matches were obtained by computing the overlap error between the regions that were matched by SIFT. In order to compute the overlap error we used the ground truth transformations. Note that this gives a strong advantage to the affine region detectors since the ground truth is usually not available. Each pair of regions was labeled as correctly matched if the overlap error was below 40%. The IBR detector obtained the best results with a 18% matching rate.

- Method B: For Fig. 2.14(d)-(f), we used the shape of two matched affine regions in order to determine the current pose of the object. In order to obtain the missing degree of freedom, the orientation was obtained by determining the dominant orientation within the patch [63]. Since for each image several transformations are available due to several extracted affine regions, we chose the transformation that corresponds best to the ground truth. The MSER and the Hessian-Affine detector perform best with a matching rate of 43% and 35%.

For the second experiment, shown in Fig. 2.15, we tracked a foot print in a snowy ground in a sequence of 453 images. The results are very similar to the first experiment's results. The success rates of our algorithms are around 88%. Again, Method A with the IBR detector performs best among the affine region detectors with a matching rate of



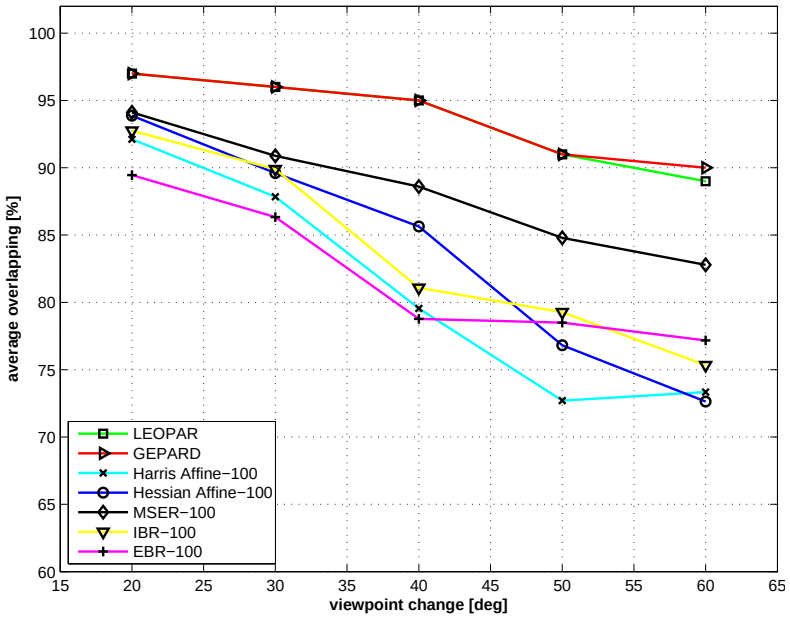
(a)



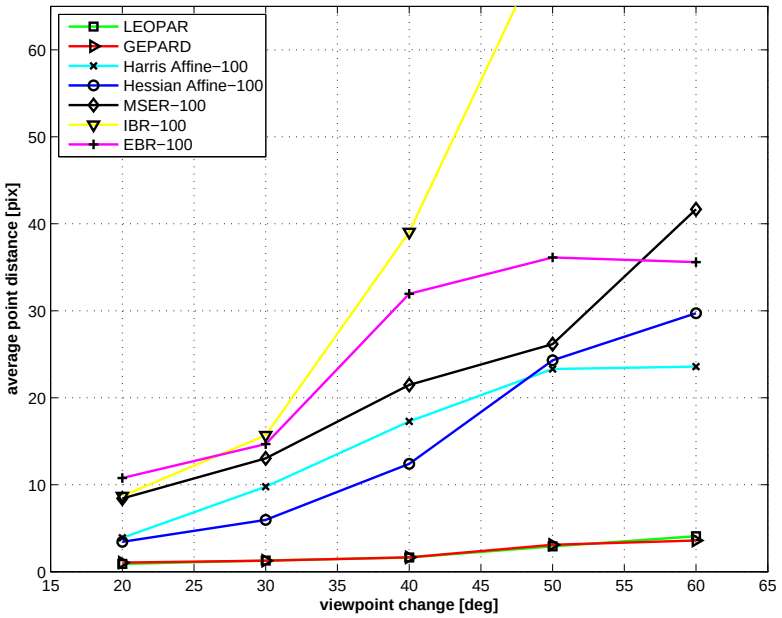
(b)

Figure 2.13: Comparing the accuracy of our methods and of affine region detectors on the Graffiti image set. **(a)** Average overlapping area of all correctly matched regions. Our method is very close to 100% and always more accurate than the other methods. **(b)** Average sum of the distances from the ground truth for the corner points. Our method is also more accurate in that case.

12%. All other detectors had success rates of below 1% (see Fig. 2.15(a)-(c)). For Method B, all affine region detectors performed around 5% except the EBR detector which had a matching rate below 1% (see Fig. 2.15(d)-(f)).



(c)

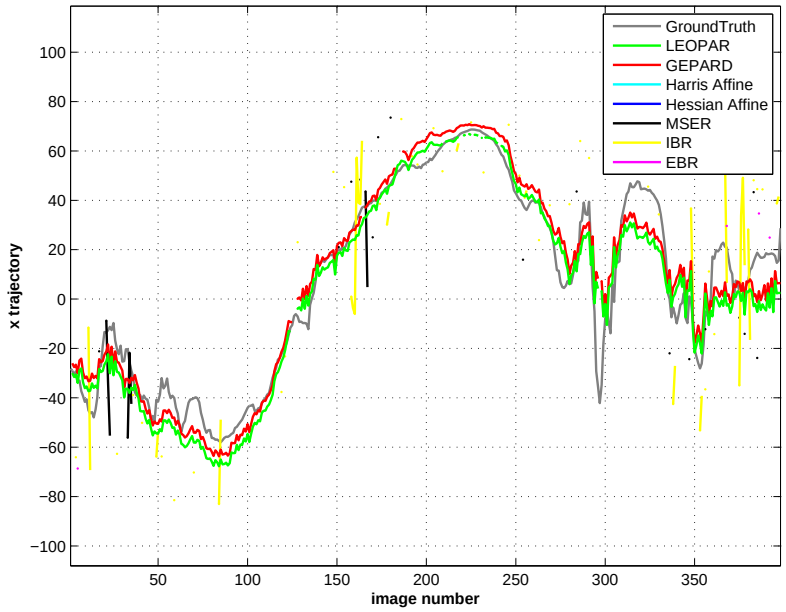


(d)

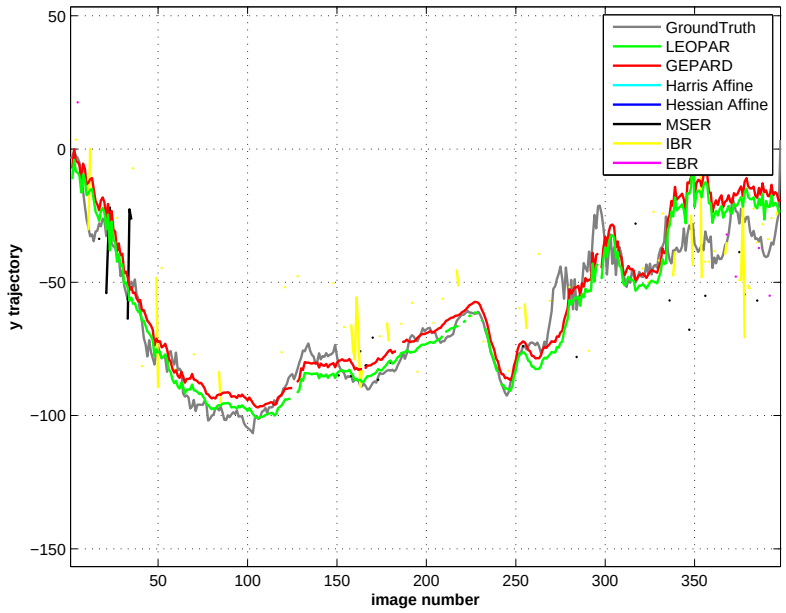
Figure 2.13 (cont.): Comparing the accuracy of our methods and of affine region detectors on the Graffiti image set. **(c)**,**(d)** Same experiments as in **(a)**, **(b)** but with only the best 100 regions kept.

2.5.3 Speed

Below, we give the computation times for training and runtime for both of our methods. All times given were obtained on a standard notebook [1]. Our implementations are written in C++ using the Intel OpenCV and IPP libraries.

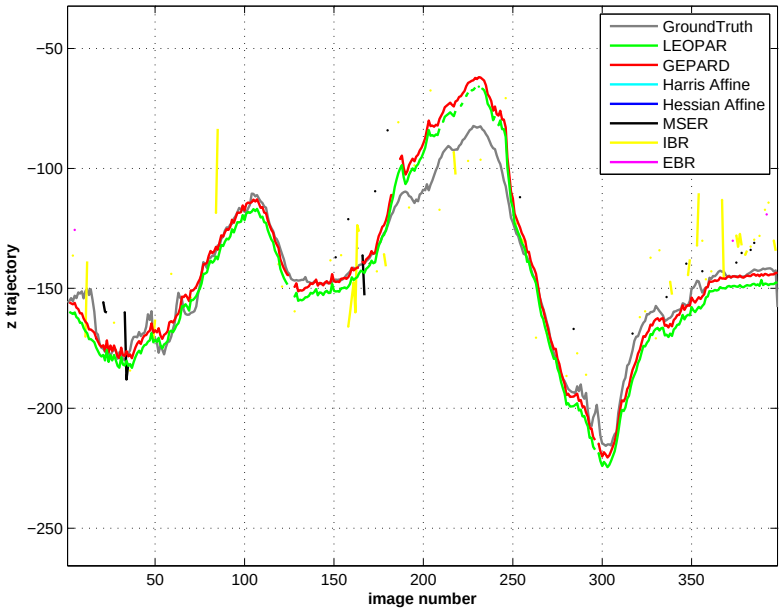


(a)

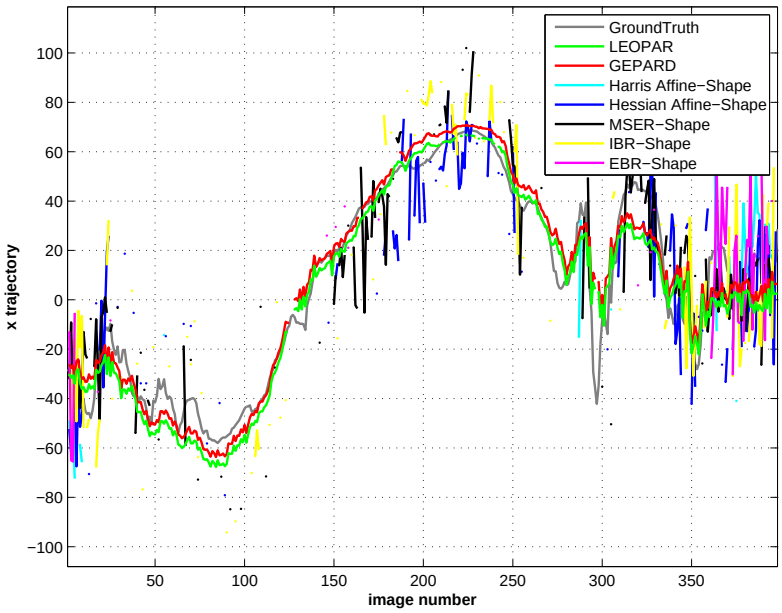


(b)

Figure 2.14: Camera trajectories retrieved by different methods for a video sequence of the power outlet of Fig. 2.23. For clarity we do not display results if the error on the camera center is larger than 50 units. **(a),(b)** X and Y coordinates of the camera center over the sequence in a coordinates system centered on the power outlet. For the affine region detectors, the camera pose was retrieved using Method A as explained in Section 2.5.2.



(c)

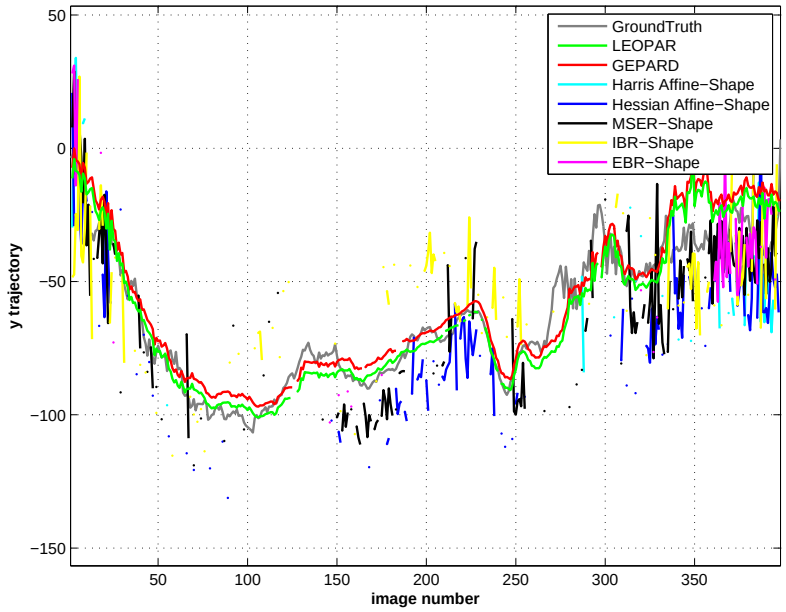


(d)

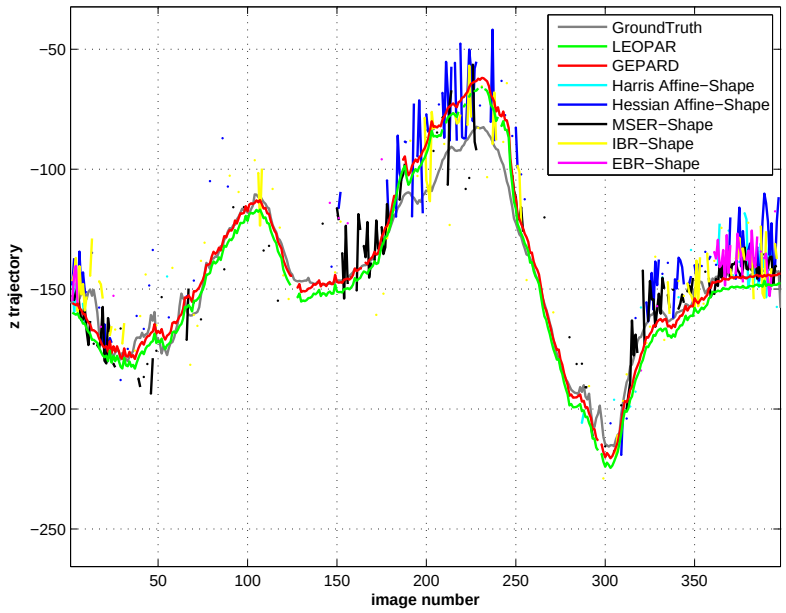
Figure 2.14 (cont.): Camera trajectories retrieved by different methods for a video sequence of the power outlet of Fig. 2.23. For clarity, we do not display results if the error on the camera center is larger than 50 units. **(c)** Z coordinates of the camera center over the sequence in a coordinates system centered on the power outlet. For the affine region detectors, the camera pose was retrieved using Method A as explained in Section 2.5.2. **(d)** Same as **(a)** but using Method B.

2.5.3.1 Training

Table 2.1 shows the advantage of Gepard over Leopar: It is much faster than Leopar. When the GPU is used, learning time drops to 5.5 milliseconds, which is largely fast



(e)



(f)

Figure 2.14 (cont.): Camera trajectories retrieved by different methods for a video sequence of the power outlet of Fig. 2.23. For clarity we do not display results if the error on the camera center is larger than 50 units. **(e),(f)** Same as **(b)**and **(c)** but using Method B for the affine region detectors as explained in Section 2.5.2.

enough for frame rate learning. This is, for instance, important for SLAM applications. Computing the $\mathbf{B}$ matrices for the refinement stage can be done in additional 29 ms on the CPU.

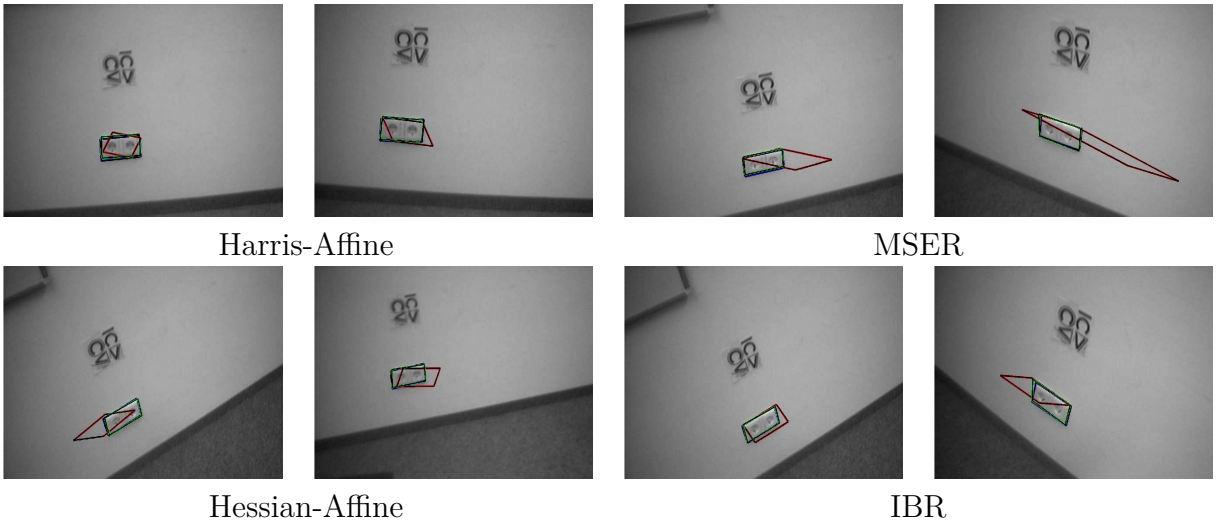


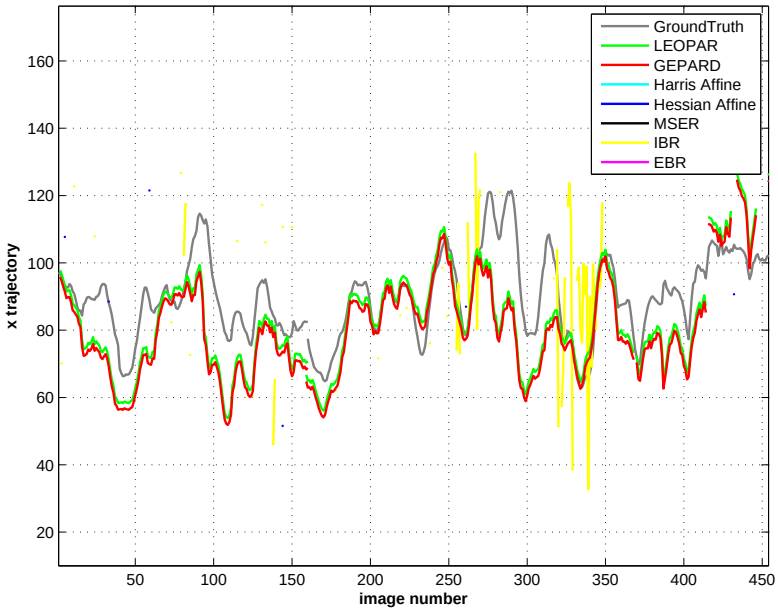
Figure 2.14 (cont.): Typical results for different methods on the example of the power outlet of Fig. 2.23. The blue quadrangle is the ground truth, the green one was retrieved using Gepard, the red one using one of the affine region detectors.

Leopar [41]	1.05 seconds
Gepard (CPU) [44]	15 milliseconds
Gepard (GPU) [44]	5.5 milliseconds

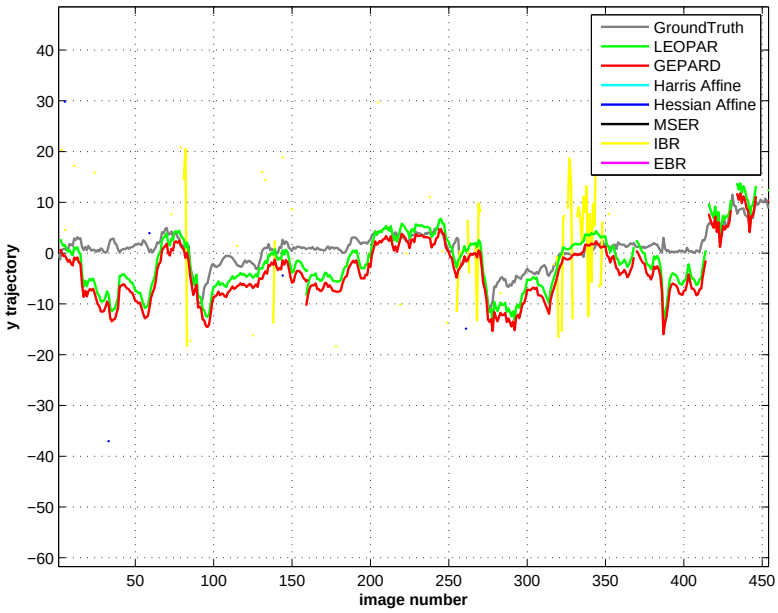
Table 2.1: Average learning time per feature for the first step of our different approaches. Gepard is more than 70 times faster when the GPU is used.

2.5.3.2 runtime

Our current implementation of Gepard runs at about 10 frames per second using 10 keypoints in the database and 70 candidate keypoints in the image. A better runtime performance is achieved with Leopar: Our implementation runs at about 10 frames per second using a database of 50 keypoints and 400 image candidate keypoints. Note that for Leopar the runtime is almost constant with respect to the size of the database and only depends on the number of candidate keypoints. For Gepard the runtime is not only influenced by the number of candidate keypoints but also behaves linearly in the number of patches in the database. The single times for one patch in the database with respect to the number of candidate keypoints in the current image are given in Fig. 2.16. We do not use any special data structure for nearest neighbor search and using, for example, KD-trees [7] would speed it up. However, due to the method's robustness and accuracy, one detected keypoint is already enough to detect the target object and to estimate its pose reliably. This can considerably speed up the processing time if the object is seen in the image and the result of only one extracted patch is enough to start a non-linear optimization process. Thus, the processing of all remaining keypoints in an image can be skipped as soon as one keypoint is extracted.

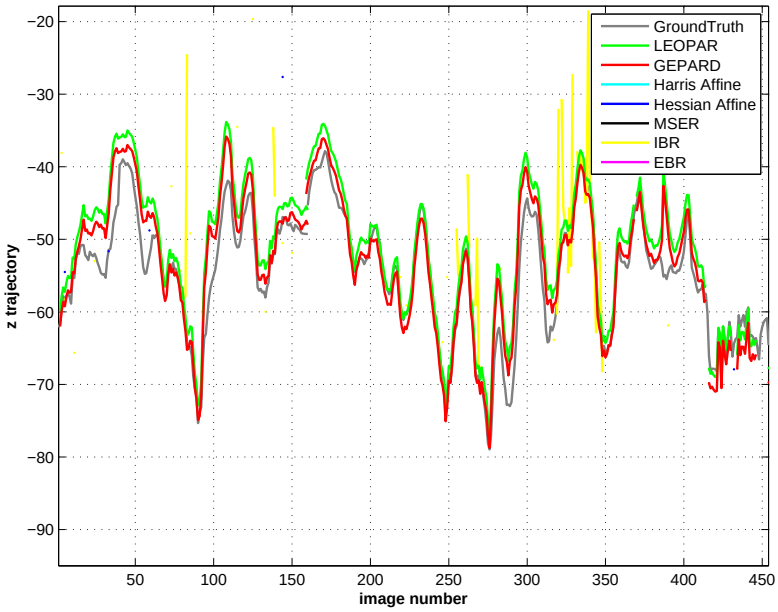


(a)

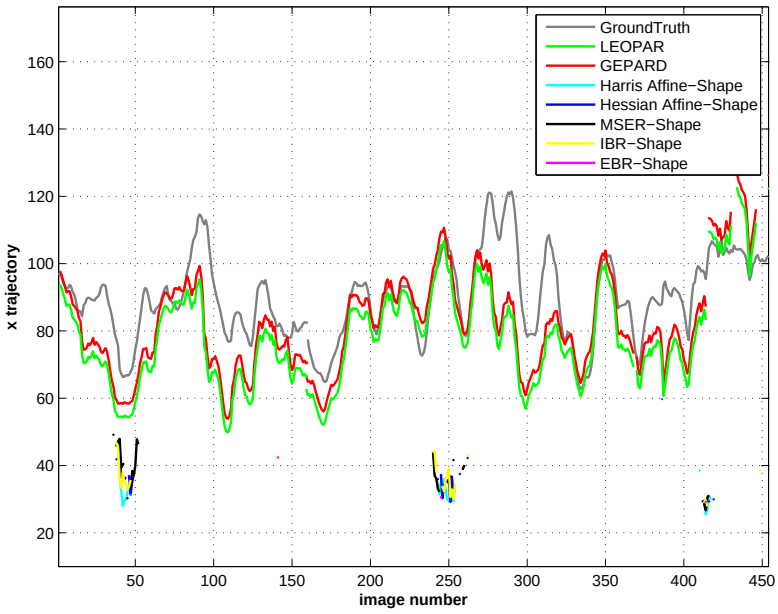


(b)

Figure 2.15: Camera trajectories retrieved by different methods for a video sequence of a footprint in snow. For clarity we do not display results if the error on the camera center is larger than 50 units. **(a),(b)**: X and Y coordinates of the camera center over the sequence in a coordinates system centered on the footprint in snow. For the affine region detectors, the camera pose was retrieved using Method A as explained in Section 2.5.2.



(c)

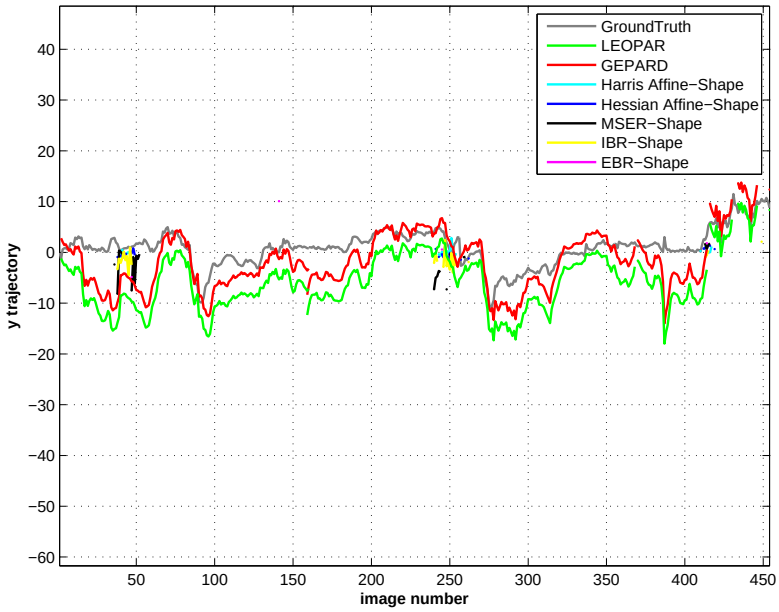


(d)

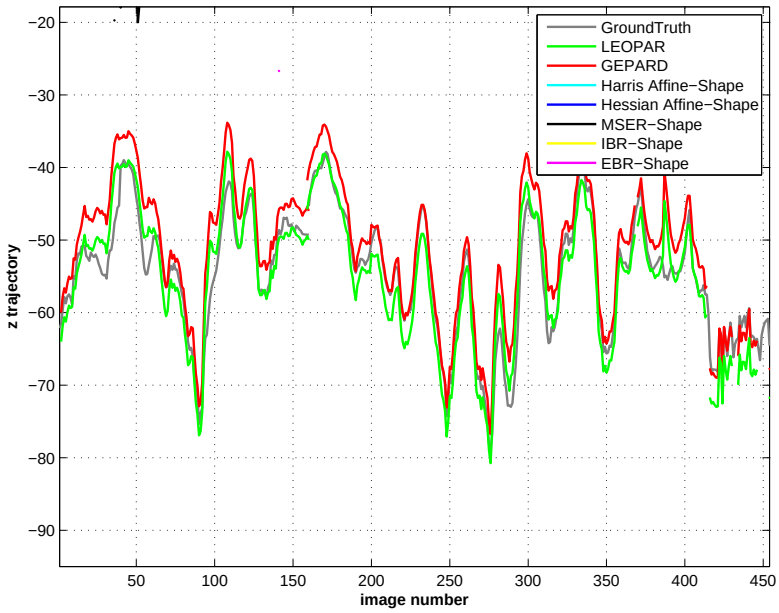
Figure 2.15 (cont.): Camera trajectories retrieved by different methods for a video sequence of a footprint in snow. For clarity we do not display results if the error on the camera center is larger than 50 units. **(c)**: Z coordinates of the camera center over the sequence in a coordinates system centered on the footprint in snow. For the affine region detectors, the camera pose was retrieved using Method A as explained in Section 2.5.2. **(d)**: Same as **(a)** but using Method B.

2.5.4 Memory

Typically, Leopar needs about 8 MB per keypoint, while Gepard needs only 350 KB. The actual amount depends on several parameters, but these values are representative. In



(e)



(f)

Figure 2.15 (cont.): Camera trajectories retrieved by different methods for a video sequence of a footprint in snow. For clarity we do not display results if the error on the camera center is larger than 50 units. **(e),(f)**: Same as **(b)**and **(c)** but using Method B for the affine region detectors as explained in Section 2.5.2.

particular the ratio between the two methods is typical: Leopar trades a large amount of memory for runtime speed.

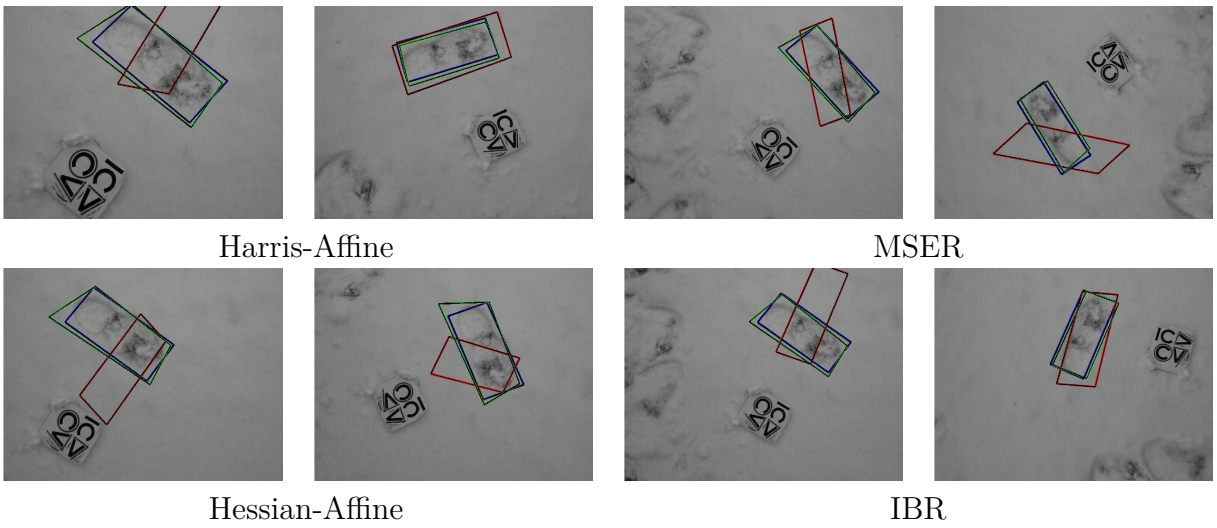


Figure 2.15 (cont.): Typical results for different methods shown on the example of a footprint in snow. The blue quadrangle is the ground truth, the green one was retrieved using Gepard, the red one using one of the affine region detectors.

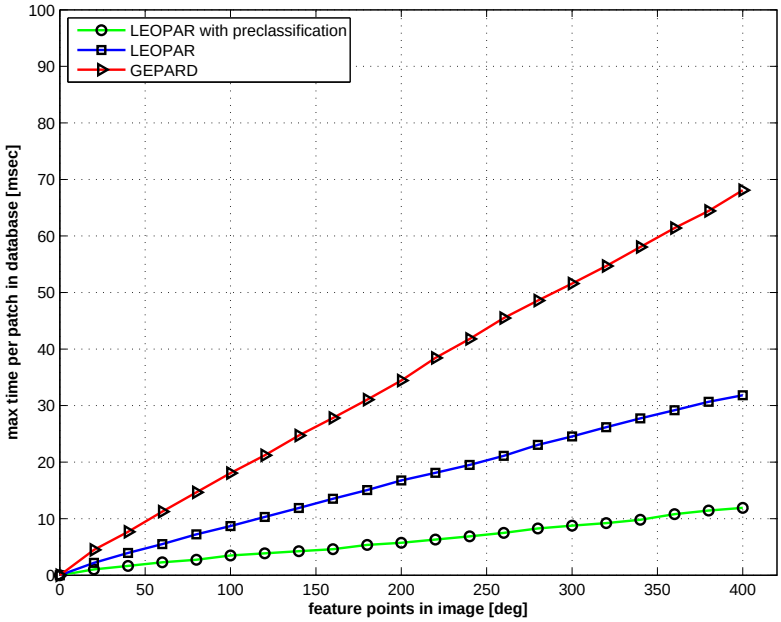


Figure 2.16: We compare the maximal runtime per keypoint of both of our methods with respect to the number of keypoints extracted in the image. For Leopar we give two different runtimes: The first one uses the same matching scheme as Gepard which is more robust but slower. If we use the patch preclassification described in Eq. 2.1 the runtime is decreased even more. Few hundreds of patches can be handled in real-time if the preclassification is switched on.

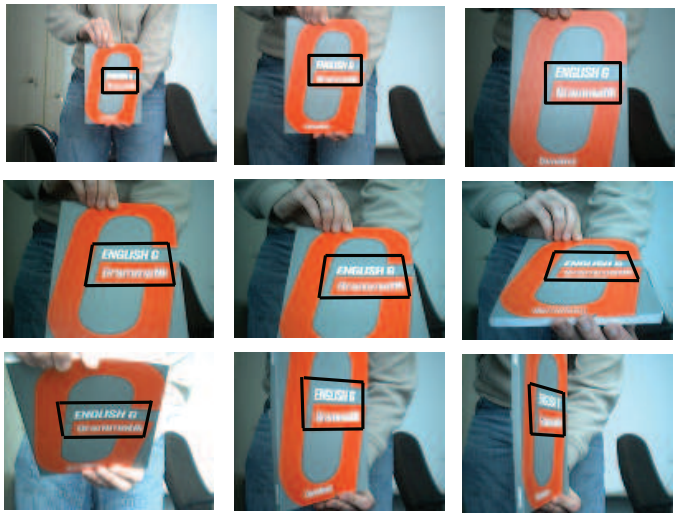


Figure 2.17: Training framework. We incrementally train the classifiers and the linear predictors over the frames of a training sequence as shown in this figure. To this end, the object is automatically registered in each incoming frame using the current state of these classifiers and linear predictors. The outcome of the single registration steps is shown by the quadrangles.

2.6 Applications

2.6.1 Training Framework

Our methods can be trained using either a small set of training images or a video sequence. In the first case, we synthesize images by warping the original patches with random homographies and adding noise to train the classifiers and the linear predictors. A video sequence and a 3D model could also be used if available. In this case we proceed as proposed in [82]: The first image is registered manually and approximately. It is used to partially train the classifiers and the linear predictors. Assuming a small interframe displacement in the training sequence, this is enough to recognize feature points in the next image, and to register it. The procedure is iterated to process the whole sequence as shown in Fig. 2.17.

2.6.2 Examples

In Figs. 2.18, 2.19, 2.20, and 2.21, we apply Leopar to object detection and pose estimation application using a low-quality camera. Leopar is robust and accurate even in presence of drastic perspective changes, light changes, blur, occlusion, and deformations. For each of these objects we learned the patches from an initial frontal view. In Figs. 2.20 and 2.21 we additionally used the template matching-based ESM algorithm [8] to refine the pose obtained from a single patch. As one can see, one extracted patch is already good enough to obtain the pose of the object reliably.

Several applications using Gepard are shown in Figs. 2.23, 2.24, 2.25, and 2.22,

showing SLAM relocalization using a single keypoint, SLAM relocalization in a room, poorly textured object detection, and deformable object detection, respectively.

For the experiment shown in Fig. 2.23, we considered an image sequence that is typically very challenging for existing SLAM systems as very few feature points can be detected, and in which the camera moves very quickly. We used a frontal view of the power outlet to train Gepard, which was then able to detect it and provide the camera pose throughout the 372 images of the sequence.

Fig. 2.22 shows another example on a larger scene. We walked around in an office space and learned a few key landmarks which were then reliably redetected in the next frames when visible, even under new viewpoints. While scalability is an issue in our current application, as we cannot handle too many patches simultaneously, this is balanced by the fact that each patch provides all six degrees-of-freedom of the camera. We currently ignore the spatial relations between patches, and the camera pose is only computed in the local frame of each patch. It should however be possible to build a SLAM system that estimates the geometric transformations between the local frames of the patches and computes the camera pose in a global coordinates system.

In Fig. 2.24 we applied Gepard to detect and estimate the 3–D pose of three poorly textured objects, under different scales and poses. This shows the potential of our approach for object recognition.

For Fig. 2.25 we tried our approach on a deformable surface. We learned five patches on the book cover from an initial frontal view. Although the book is then strongly deformed and we do not model the deformation within our recognition pipeline, most of the learned patches are reliably recognized. The local frames also fit well to the local deformations, at least visually. This provides very strong constraints on the shape of the surface that could be exploited to retrieve the deformations, for example using a global deformation model such as the one developed in [91].

A clear limitation of our approach is that it relies on feature point detection. If the feature point corresponding to the patch center is not detected in the first place, because of image noise or some imperfection of the feature point detector, our approach fails. Skipping the feature point detection step, for example, by parsing the complete image and looking for the patches, is part of our future work.

Another part of future work is the integration of Leopar and Gepard into the *Computer Vision Cad Model* (CV-CAD) [73]. The CV-CAD incorporates computer vision methods and 3D data and is highly scalable with respect to the number of object parts. Its goal is the easy composition of object parts — each described by a single CV-CAD model — to a new object without the need of relearning and retraining computer vision methods. Instead, the computer vision functionality of the new object is built by fusing the vision functionality of the single CV-CAD entities.

This is made possible by using *Natural 3D Markers* (N3M's) [40]. N3M's are local entities that are able to autonomously estimate and self-verify the pose of the object they belong to only based on spatially limited regions and independently of all remaining features. Since Leopar and Gepard show these properties, they can be clearly classified as N3M's.

For the new composition of an object, one then has to apply only the following steps:

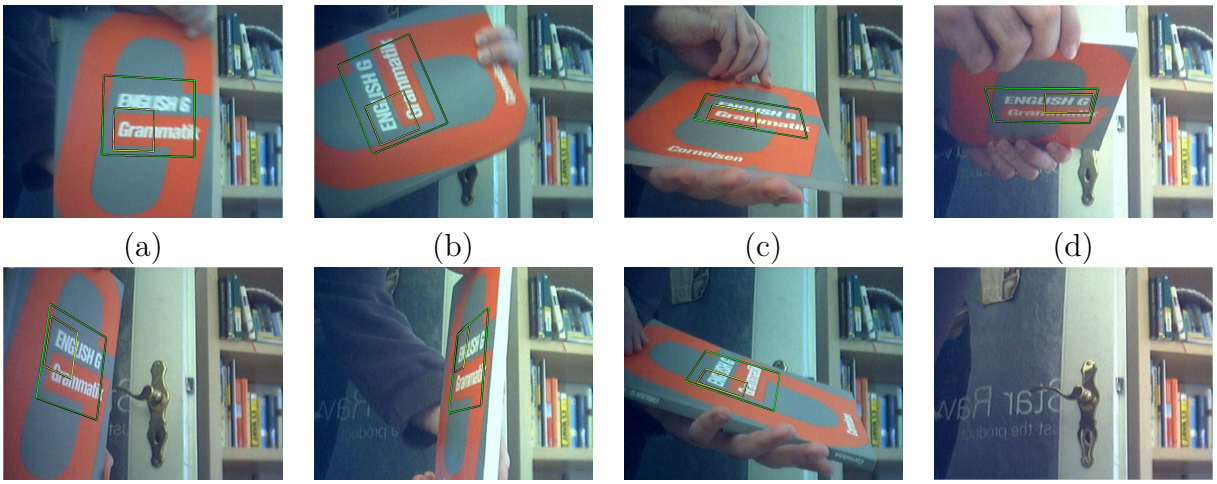
determining which of the N3M's are still visible, deactivating invisible N3M's, computing the new relative pose of the single N3M's and updating the global geometry of the new object. This can be done fully automatically by simply taking some images of the new object and computing the relative pose of the attached N3M's. It then immediately allows for the efficient computation of all remaining steps. As a result, the new composition of single object parts becomes highly efficient and uncomplicated. Even non-expert users are expected to be able to create new objects which can be handled with computer vision techniques, if the corresponding CV-CAD models for the single parts exist.

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

Figure 2.18: Robustness of Leopar to deformation and occlusion. **(a)** Patches detected on the book in a frontal view. **(b)** Most of these patches are detected even under a strong deformation. **(c)** The book is half occluded but some patches can still be extracted. **(d)** The book is almost completely hidden but one patch is still correctly extracted. No outliers were produced.

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

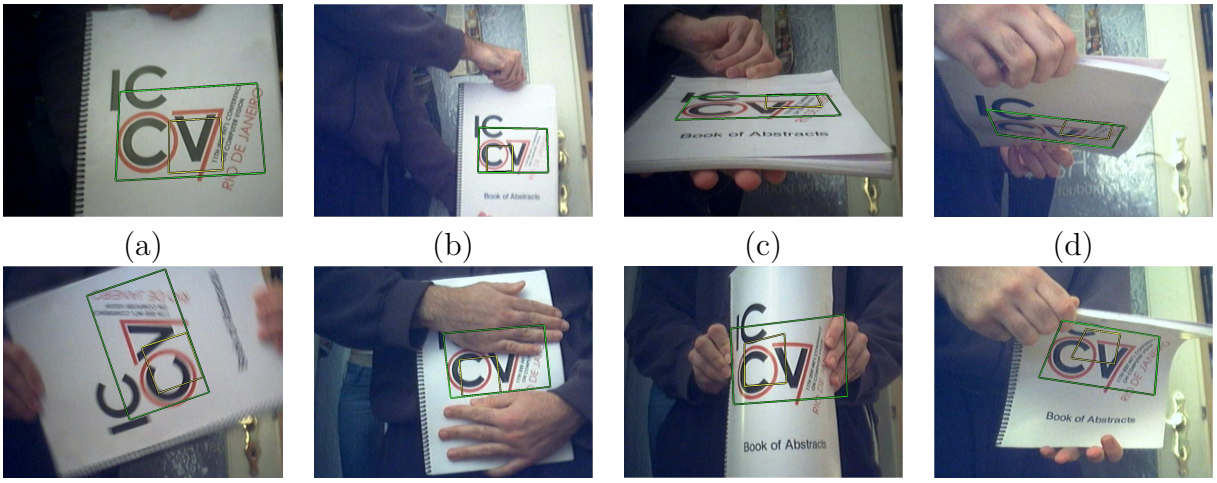
Figure 2.19: Accuracy of Leopar of the retrieved transformation. For each of these images, we draw the borders of the book estimated from a single patch. This is made possible by the fact we estimate a full perspective transform instead of only an affine one.



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

(e) (f) (g) (h)

Figure 2.20: Some frames of a Tracking-by-Detection with Leopar sequence shot with a low-quality camera. **(a)-(g)** The book pose is retrieved in each frame independently at 10fps. The yellow quadrangle is the best patch obtained by Leopar. The green quadrangle is the result of the ESM algorithm [8] initialized with the pose obtained from this patch. The retrieved pose is very accurate despite drastic perspective and intensities changes and blur. **(h)** When the book is not visible, our method does not produce a false positive.



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

(e) (f) (g) (h)

Figure 2.21: Another example of a Tracking-by-Detection sequence with Leopar. The book pose is retrieved under **(b)** scale changes, **(c-d)** drastic perspective changes, **(e)** blur, **(f)** occlusion, and **(g-h)** deformations.

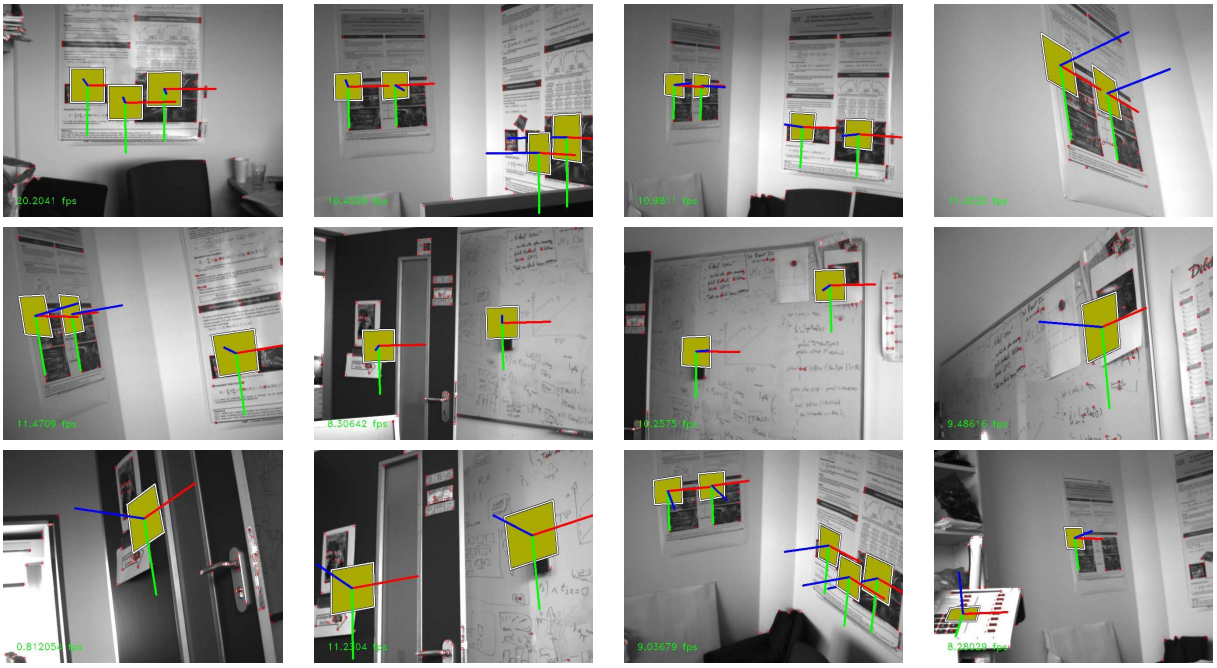


Figure 2.22: An example of SLAM relocalization with Gepard, using 8 different patches.

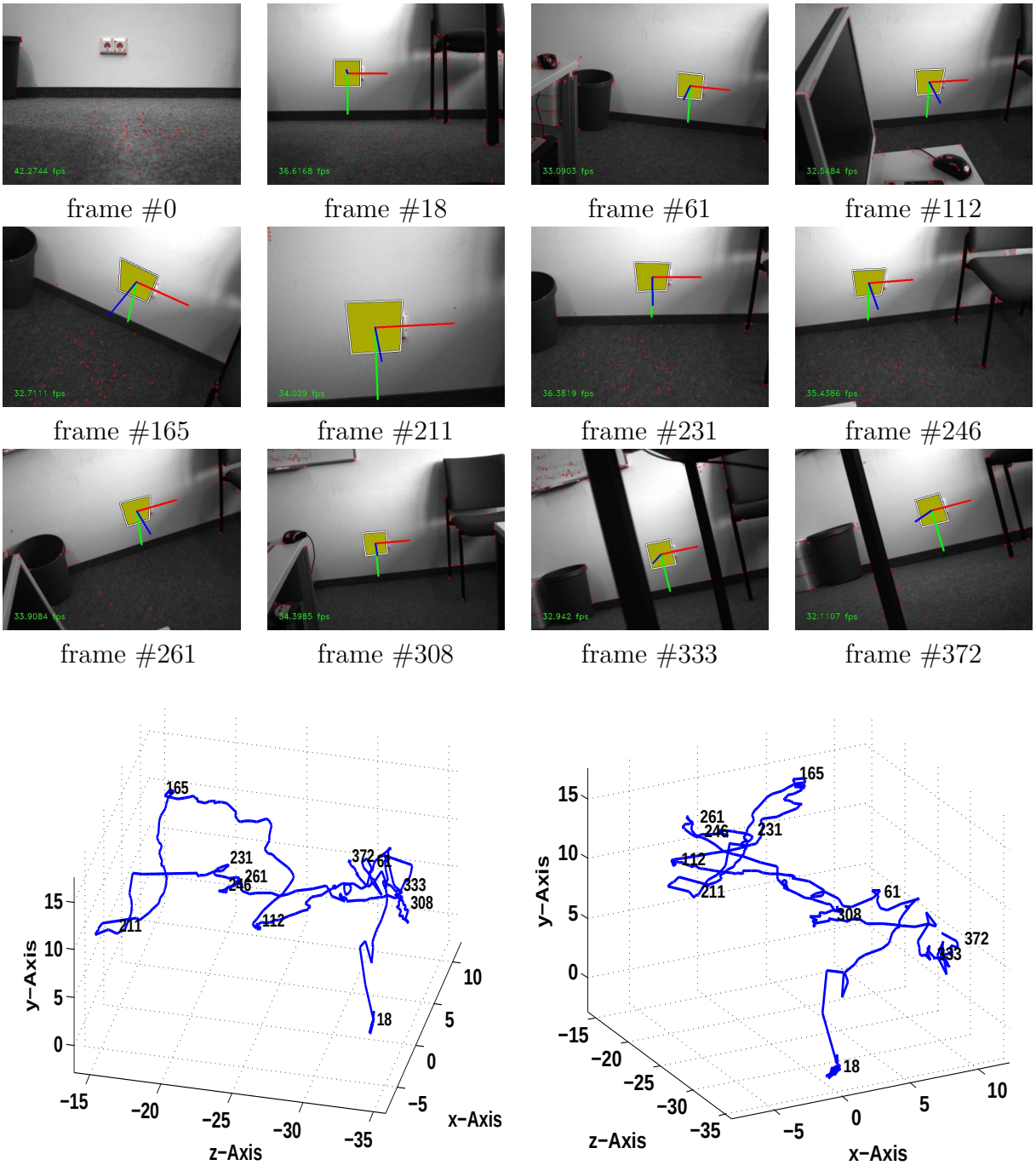


Figure 2.23: Tracking a power outlet with Gepard. We can retrieve the camera trajectory through the scene despite very limited texture and large viewpoint changes. Since the patch is detected and its poses estimated in every frame independently, the method is very robust to fast motion and occlusion. The two graphs show the retrieved trajectory.

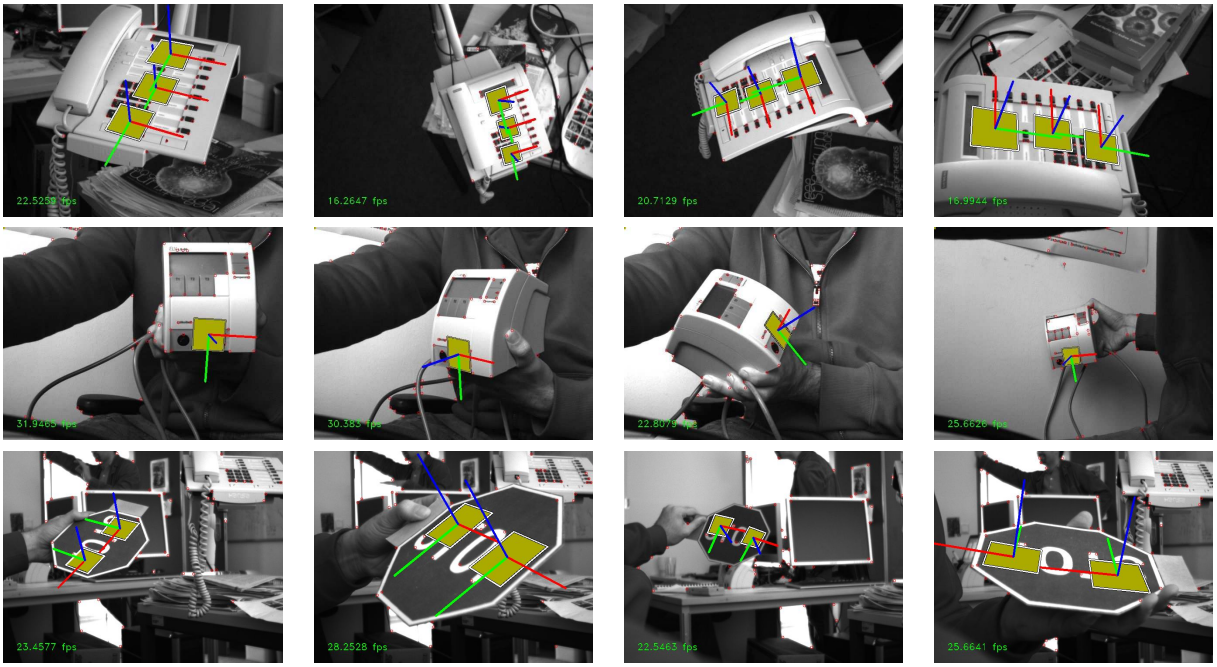


Figure 2.24: Application to tracking-by-detection of poorly textured objects under large viewing changes with Gepard.

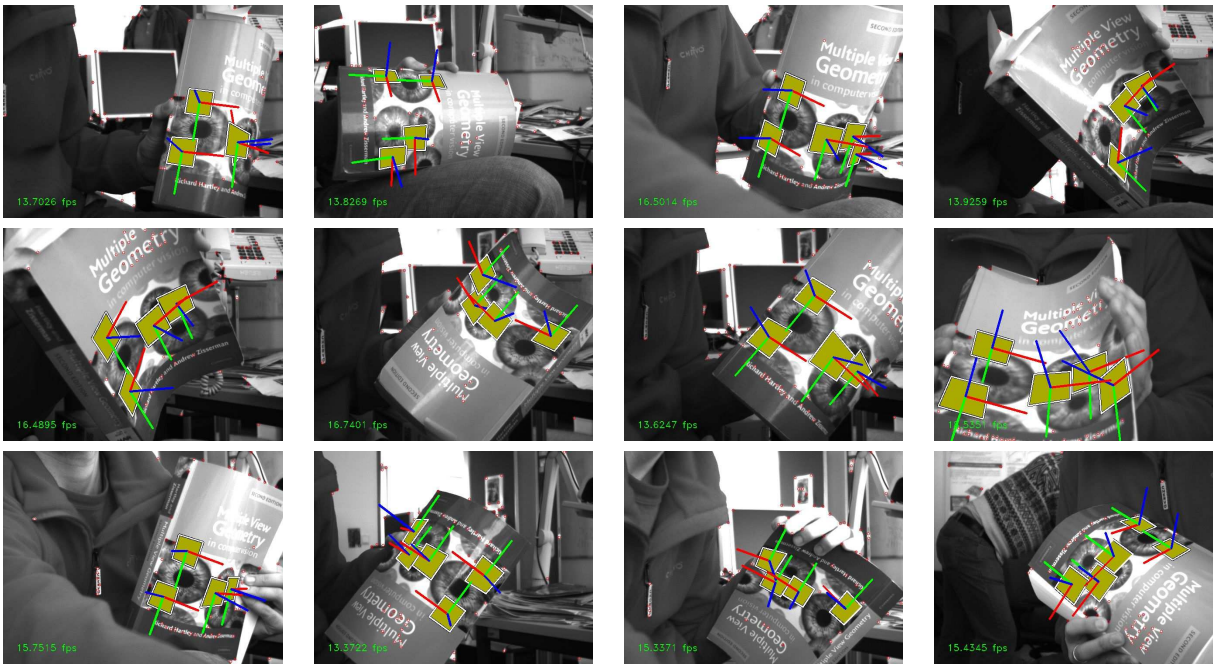


Figure 2.25: Application to a deformable object with Gepard. We can retrieve an accurate pose even under large deformations. While it is not done here, such cues would be very useful to constrain the 3D surface estimation.

CHAPTER

THREE

REAL-TIME DETECTION OF TEXTURE-LESS OBJECTS BY TEMPLATE MATCHING

At least since the seminal work of Viola and Jones [108], the dominant approach to real-time object recognition is to train a classifier offline to achieve fast online performance [21, 26, 13, 27]. This is remarkably effective when the target objects are known *a priori* but inappropriate when new objects have to be learned online. In robotic applications for instance, autonomous systems continuously have to adapt to a changing and unknown environment which makes it necessary to learn and recognize new objects in runtime. Therefore, statistical-learning techniques are often not adequate, since they tend to require many samples and to be too computationally intensive for real-time performance.

When the object is textured enough for keypoints to be found and recognized on the basis of their appearance, this difficulty has been successfully addressed by defining patch descriptors that can be computed quickly and used to characterize the object [63]. However, this kind of approach will fail on texture-less objects such as those of Fig. 3.1, whose appearance is often dominated by their projected contours. To overcome this problem, we propose a novel approach based on real-time template recognition, where the templates can both be built and matched very quickly. We will show that this makes it very easy and virtually instantaneous to learn new incoming objects by simply adding new templates to the database while maintaining reliable real-time recognition.

However, we also wish to keep the efficiency and robustness of statistical methods, as they learn how to reject unpromising image locations very quickly and tend to be very robust, because they can generalize well from the training set. For these reasons, we describe two new approaches in this theses — the one consisting of a new template [46] and the other consisting of a new image representation [43] — which both hold local image statistics and are fast to compute. In short, they are designed to be invariant to small translations and deformations, which has been shown to be a key factor to generalization [63]. In addition, they also allow us to quickly parse the image by skipping many locations without loss of reliability.

The first presented approach — called Dot (for *Dominant Orientation Templates*) — is a gradient-based template representation that is invariant enough to make search in the images very fast and generalizes well (see Fig. 3.1). As a result, we can almost

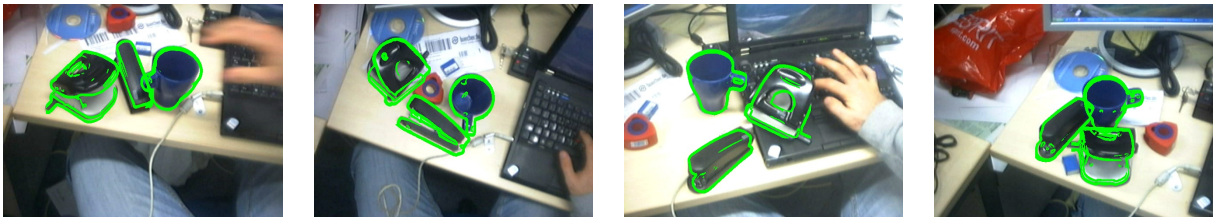


Figure 3.1: Dot Overview. Our templates can detect non-textured objects over little cluttered background in real-time without relying on feature point detection. Adding new objects is fast and easy, as it can be done online without the need for an initial training set. Only a few templates are required to cover all appearances of the objects.

Figure 3.2: Line Overview. **Upper Row:** Our method can detect texture-less 3D objects in real-time under different poses over heavily cluttered background using spread gradient orientations. **Lower Row:** it can also be generalized using multiple different modalities which increases the robustness and decreases the number of false positives.

instantaneously learn new objects and recognize them in real-time without requiring much time for training or any feature point detection at runtime.

The representation of Dot is related to the Histograms-of-Gradients (HoG) based representation [21] that has proved to generalize well. Instead of local histograms, it relies on locally dominant orientations, and is made explicitly invariant to small translations. Our experiments show that it is in practice comparable discriminant as HoG, while being much faster. Because it is explicitly made invariant to small translations, we can skip many locations while parsing the images without the risk of missing the targets. Moreover we developed a bit-coding method inspired by [101] to evaluate an image location for the presence of a template. It mostly uses simple bit-wise operations, and is therefore very fast on modern CPUs. Our similarity measure also fulfills the requirements for recent branch-and-bound exploration techniques [56], speeding-up the search even more.

While Dot works remarkably well for small objects over little cluttered background, it has its problems with strong background clutter and significantly slows down when the size of the templates increases.

Therefore, we present a second approach — called Line (for *LINE*arizing the memory) — that addresses this issue while being much faster for larger templates. While initially designed for gradient orientations only, we show in this thesis how to generalize this method to make use of multiple different visual cues or modalities simultaneously once a proper quantization of the input data is available.

Instead of making the templates invariant to small deformations and translations as in Dot, Line builds a representation of the input images which has similar invariance properties. This allows us to consider all quantized values in local image neighborhoods instead of the dominant ones only. By introducing a novel similarity measure taking them into account, we can now avoid the problems due to too strong background clutter as illustrated by Figure 3.2.

To avoid slowing down detection when using this finer method, we have to make careful considerations about how modern CPUs work. A naive implementation would result in many "memory cache misses", which slow down the computations, and we thus show how to structure our image representation in memory to prevent them and to additionally exploit heavy SSE parallelization. We consider this as an important contribution: Because of the nature of the hardware improvements, it is not guaranteed anymore that legacy code will run faster on the new versions of CPUs [11]. This is particularly true for Computer Vision, which algorithms are often computationally expensive. It is now required to take the CPU architecture into account, which is not an easy task.

In the remainder of this chapter we first discuss related work before we explain Dot and Line and show how their similarity measures can be evaluated very fast. For each method we additionally show quantitative experiments and real world applications.

3.1 Related Work

Template Matching has played an important role in tracking-by-detection applications for many years. This is due to its simplicity and its capability to handle different types of objects. It neither needs a large training set nor a time-consuming training stage, and can handle low-textured or texture-less objects, which are, for example, difficult to detect with feature points-based methods [63, 45]. Unfortunately, this increased robustness often comes at the cost of an increased computational load that makes naïve template matching inappropriate for real-time applications. So far, several works have attempted to reduce this complexity.

An early approach to Template Matching [79] and its extension [30] include the use of the Chamfer distance between the template and the input image contours as a dissimilarity measure. For instance, Gavrila and Philomin [30] introduced a coarse-to-fine approach in shape and parameter space using Chamfer Matching [12] on the Distance Transform of a binary edge image. The Chamfer Matching minimizes a generalized distance between two sets of edge points. Although being fast when using the Distance Transform (DT), the disadvantage of the Chamfer Transform is its sensitivity to outliers which often result from occlusions.

Another common measure on binary edge images is the Hausdorff distance [90]. It measures the maximum of all distances from each edge point in the image to its nearest

neighbor in the template. However, it is sensitive to occlusions and clutter. Huttenlocher *et al.* [50] tried to avoid that shortcoming by introducing a generalized Hausdorff distance which only computes the maximum of the k-th largest distances between the image and the model edges and the l-th largest distances between the model and the image edges. This makes the method robust against a certain percentage of occlusions and clutter. Unfortunately, a prior estimate of the background clutter in the image is required but not always available. Additionally, computing the Hausdorff distance is computationally expensive and prevents its real-time application when many templates are used.

Both Chamfer Matching and the Hausdorff distance can easily be modified to take the orientation of edge points into account. This drastically reduces the number of false positives as shown in [79], but unfortunately also increases the computational load.

A more recent approach that speeds up the oriented Chamfer Matching uses binary depth-edges generated by a multi-flash camera [62]. This method computes the distance transform on the different orientation channels of the binary depth-edge map and uses a line-based integral image to enable the quick computation of the similarity measure. While this approach seems to be quite robust in cluttered environments, it needs special hardware to be robust.

[48] is also based on the Distance Transform, however, it is invariant to scale changes and robust enough against planar perspective distortions to do real-time matching. Unfortunately, it is restricted to objects with closed contours, which are not always available.

Another class of algorithms for object recognition is based on the generalized Hough transform [4]. Its advantage is its robustness to occlusion and clutter. Unfortunately, the generalized Hough Transform in its conventional form requires large amounts of memory and long computation times to recognize an object. Additionally, it is very sensitive to the quantization of the accumulator space and to correct edge detection which makes it inappropriate for real-time detection.

All these methods use binary edge images obtained with a contour extraction algorithm, using the Canny method [16] for example, and they are very sensitive to illumination changes, noise and blur. For instance, if the image contrast is lowered, the number of extracted edge pixels progressively decreases which has the same effect as increasing the amount of occlusion.

The method proposed in [97] tries to overcome these limitations by considering the image gradients in contrast to the image contours. It relies on the dot product as a similarity measure between the template gradients and those in the image. Unfortunately, this measure rapidly declines with the distance to the object location, or when the object appearance is even slightly distorted. As a result, the similarity measure must be evaluated densely, and with many templates to handle appearance variations, making the method computationally costly. Using image pyramids provides some speed improvements, however, fine but important structures tend to be lost if one does not carefully sample the scale space.

Amit [1] also proposed a coarse to fine approach, however, in contrast to [97], by making use of spreading gradient orientations in local neighborhoods. The amount of spreading is learned for each object part in an initial stage. While this approach — used for license plate reading — achieves high recognition rates, it is not real-time capable.

If the image has to be parsed only with a small number of templates, another method which is fast but simple is commonly used: the Fast Fourier Transform [20] provides efficient means to match a single template in real-time with an image, given that the Fourier transforms of both are given. In order to make the FFT applicable the underlying similarity measure has to be formulated in a correlation based scheme [69]. However, as soon as some tens of templates have to be shifted over the image, this approach becomes quickly intractable.

Histogram of Gradients (HoG) [21] is another related and very popular method. It statistically describes the distribution of intensity gradients in localized portions of the image. The approach is computed on a dense grid with uniform intervals and uses overlapping local histogram normalization for better performance. It has proven to give reliable results but tends to be slow due to the computational complexity.

Ferrari *et al.* [27] provided a learning based method that recognizes objects via a Hough-style voting scheme with a non-rigid shape matcher on object boundaries of a binary edge image. The approach applies statistical methods to learn the model from few images that are only constrained within a bounding box around the object. While giving very good classification results, the approach is neither appropriate for object tracking in real-time due to its expensive computation nor is it precise enough to return the accurate pose of the object. Additionally, it is sensitive to the results of the binary edge detector, an issue that we discussed before.

Grabner and Bischof [35, 36] developed another learning based approach that put more focus on online learning. In [35, 36] it is shown how a classifier can be trained online in real-time, with a training set generated automatically. However, [35] was demonstrated on textured objects, and [36] cannot provide the object pose.

Opposite to the above mentioned learning based methods, there are also approaches that are specifically trained on different viewpoints. As with our template-based approach, they can detect objects under different poses but typically require a large amount of training data and a long offline training phase. For example, in [109, 49, 83], one or several classifiers are trained to detect faces or cars under various views.

More recent approaches for 3D object detection are related to object class recognition. Stark *et al.* [96] rely on 3D CAD models and generate a training set by rendering them from different viewpoints. Liebelt and Schmid [60] combine a geometric shape and pose prior with natural images. While these approaches are able to generalize to the object class they are not real-time capable and require expensive training.

From the related works which take into account multi-modal data information there are mainly approaches related to pedestrian detection [24, 25, 31, 112]. They use three kinds of clues: image intensity, depth and motion (optical flow). The most recent approach of Enzweiler *et. al* [24] builds part based models of pedestrians in order to handle occlusions caused by other objects and not specifically self occlusions which are modeled in other approaches [25, 112]. Besides pedestrian detection, there has been an approach to object classification, pose estimation and reconstruction introduced by [99]. Similar to us, the training data set is composed of depth and image intensities while the object classes are detected using the modified Hough transform. While being quite effective in real applications these approaches still require exhaustive training using large training data

sets. This is usually prohibited in robotic applications where the robot has to explore an unknown environment and learn new objects online.

The first method presented in this chapter — called Dot [46] — has the strength of the similarity measure of [97], the robustness of [21] and the online learning capability of [35, 36]. In addition, by binarizing the template representation and using a recent branch-and-bound method of [56] this method becomes very fast, making possible the detection of untextured 3D objects in real-time.

However, we noticed that Dot degrades significantly when the gradient orientations are disturbed by strong gradients of different orientations coming from background clutter in the input images. In practice, this often happens in the neighborhood of the silhouette of an 3D object, which is unfortunate as the silhouette is a very important cue especially for texture-less objects.

The second method we present in this chapter — called Line [43] — does not suffer from this problem while running at the same speed. This is made possible by aligning for each feature the local neighborhood exactly to the associated location whereas in Dot [46], BiGG [72], HoG [21] or SIFT [63], the features are adjusted only to some regular grid. This can be done efficiently by spreading image features to their local neighborhood and by precomputing for each feature so called *response maps* which are shared between the templates. As such, the similarity measure for each image location is evaluated only once, and by making use of the architecture of modern computers we obtain a great speed-up at runtime.

Furthermore, we show how Line can make use of multiple different modalities which increases robustness even further and significantly decreases the number of false positives. We also show how input data has to be processed on the example of color image and dense depth data.

3.2 Dot: Dominant Orientation Templates

In this section, we describe our Dominant Orientation Templates, and how they can be built and used to parse images to quickly find objects. We will start by deriving our similarity measure, emphasizing the contributions of each aspect of it. We then show how to use a binary representation to compute the similarity using efficient bit-wise operations. We finally demonstrate how to use it within a branch-and-bound exploration of the image.

3.2.1 Initial Similarity Measure

Our starting idea is to measure the similarity between an input image $\mathcal{I}$, and a reference image $\mathcal{O}$ of an object centered on a location c in the image $\mathcal{I}$ by comparing the orientations of their gradients.

We chose to consider image gradients because they proved to be more discriminant than other forms of representations [63, 97] and are robust to illumination change and noise. For even more robustness to such changes, we use their magnitudes only to retain the orientations of the strongest gradients, without using their actual values for matching. Also, to correctly handle object occluding boundaries, we consider only the orientations

of the gradients, by contrast with their directions (thus, two vectors with a 180deg angle between them have the same orientation). In this way, the measure will not be affected if the object is over a dark background, or a bright background. Moreover, as in SIFT or HoG [21], we discretize the orientations to a small number n_o of integer values.

Our initial energy function $\mathcal{E}_1$ counts how many orientations are similar between the image and the template centered on location c, and can be formalized as:

$$\mathcal{E}_1(\mathcal{I}, \mathcal{O}, c) = \sum_r \delta\Big(\text{ori}(\mathcal{I}, c + r) = \text{ori}(\mathcal{O}, r)\Big)\,, \tag{3.1}$$

where

- $\delta(P)$ is a binary function that returns 1 if P is true, 0 otherwise;
- $\text{ori}(\mathcal{O}, r)$ is the discretized gradient orientation in the reference image $\mathcal{O}$ at location r which parses the template. Similarly, $\text{ori}(\mathcal{I}, c + r)$ is the discretized gradient orientation at c shifted by r in the input image $\mathcal{I}$.

3.2.2 Robustness to Small Deformations

To make our measure tolerant to small deformations, and also to make it faster to compute, we will not consider all possible locations, and will decompose the two images into small squared regions $\mathcal{R}$ over a regular grid. For each region, we will consider only the dominant orientations. Such an approach is similar to the HMAX pooling mechanism [92]. Our similarity measure can now be modified as:

$$\mathcal{E}_2(\mathcal{I}, \mathcal{O}, c) = \sum_{\mathcal{R} \text{ in } \mathcal{O}} \delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \text{DO}(\mathcal{O}, \mathcal{R})\Big)\,, \tag{3.2}$$

where $\text{DO}(\mathcal{O}, \mathcal{R})$ returns the set of orientations of the strongest gradients in region $\mathcal{R}$ of the object reference image. In contrast, $\text{do}(\mathcal{I}, c + \mathcal{R})$ returns only one orientation, the orientation of the strongest gradient in the region $\mathcal{R}$ shifted by c in the input image.

The reason why we chose each region in $\mathcal{O}$ to be represented by the strongest gradients is that the strongest gradients are easy and fast to identify and very robust to noise and illumination change. Moreover, to describe uniform regions, we introduce the symbol $\perp$ to indicate that no reliable gradient information is available for the region. The DO(.) function therefore returns either a set of discretized gradient orientations of the k strongest gradients in the range of $[0, n_o - 1]$ or $\{\perp\}$, and can be formally written as:

$$\text{DO}(\mathcal{O}, \mathcal{R}) = \begin{cases} S(\mathcal{O}, \mathcal{R}) & \text{if } S(\mathcal{O}, \mathcal{R}) \neq \emptyset, \\ \{\perp\} & \text{otherwise} \end{cases} \tag{3.3}$$

with

$$S(\mathcal{O}, \mathcal{R}) = \{\text{ori}\,(\mathcal{O}, l) : l \in \text{maxmag}_k(\mathcal{R}) \wedge \text{mag}(\mathcal{O}, l) > \tau\} \tag{3.4}$$

where

- l is a pixel location in $\mathcal{R}$,

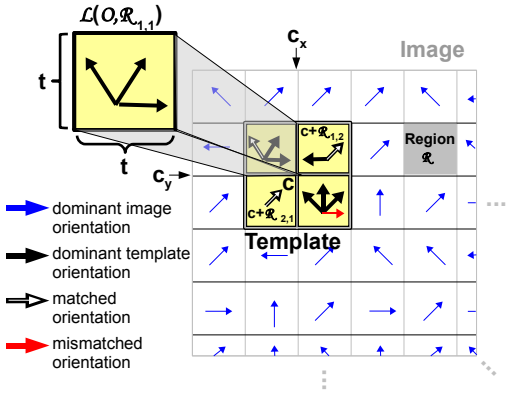


Figure 3.3: Similarity measure $\mathcal{E}_4$. Our final energy measure $\mathcal{E}_4$ counts how many times a local dominant orientation for a region $\mathcal{R}$ in the image belongs to the corresponding precomputed list of orientations $\mathcal{L}(\mathcal{O},\mathcal{R})$ for the corresponding template region. Each list is made of the local dominant orientations that are in the region $\mathcal{R}$ when the object template is slightly translated.

- $\text{ori}(\mathcal{O},l)$ is the gradient orientation at l in image $\mathcal{O}$, and $\text{mag}(\mathcal{O},l)$ its magnitude,
- $\text{maxmag}_k(\mathcal{R})$ is the set of locations for the k strongest gradients in $\mathcal{R}$. In practice we take $k=7$ but the choice of k does not seem critical.
- τ is a threshold on the gradient magnitudes to decide if the region is uniform or not.

The function $\text{do}(\mathcal{I},c+\mathcal{R})$ is computed similarly in the input image $\mathcal{I}$. However, to be faster at runtime, in $\text{do}(\mathcal{I},c+\mathcal{R})$, k is restricted to 1, and therefore $\text{do}(\mathcal{I},c+\mathcal{R})$ returns only one single element.

3.2.3 Invariance to Small Translation

We will now explicitly make our similarity measure invariant to small motions. In this way, we will be able to consider only a limited number of locations c when parsing an image and save a significant amount of time without increasing the chance of missing the target object. To do so, we consider a measure that returns the maximal value of $\mathcal{E}_2$ when the object is slightly moved, which can be written as:

$$\begin{aligned}\mathcal{E}_3(\mathcal{I},\mathcal{O},c) &= \max_{M\in\mathcal{M}} \mathcal{E}_2(\mathcal{I},\mathbf{w}(\mathcal{O},M),c)\\ &= \max_{M\in\mathcal{M}} \sum_{\mathcal{R}\text{ in }\mathcal{O}} \delta\Big(\text{do}(\mathcal{I},c+\mathcal{R}) \in \text{DO}(\mathbf{w}(\mathcal{O},M),\mathcal{R})\Big)\,,\end{aligned} \tag{3.5}$$

where $\mathbf{w}(\mathcal{O},M)$ is the image $\mathcal{O}$ of the object warped using a transformation M. In practice, we consider for M only 2D translations as it appears sufficient to handle other small deformations, and $\mathcal{M}$ is the set of all (small) translations in the range $[-\frac{t}{2};+\frac{t}{2}]^2$.

There is of course a limit for the range t. A large t will result in high speed-up but also in a loss of discriminative power of the function. In practice, we found that $t=7$ for 640×480 images is a good trade-off.

3.2.4 Ignoring the Dependence between Regions

Our last step is to ignore the dependence between the different regions $\mathcal{R}$. This will simplify and significantly speed-up the computation of the similarity. We therefore approximate $\mathcal{E}_3$ as given in Eq.(3.5) by:

$$\begin{aligned} & \mathcal{E}_4(\mathcal{I}, \mathcal{O}, c) \\ = & \sum_{\mathcal{R} \text{ in } \mathcal{O}} \max_{M \in \mathcal{M}} \delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \text{DO}(\mathbf{w}(\mathcal{O}, M), \mathcal{R})\Big) \,. \end{aligned} \tag{3.6}$$

The speed-up comes from the fact that, for each region $\mathcal{R}$, we can precompute a list $\mathcal{L}(\mathcal{O}, \mathcal{R})$ of the dominant orientations in $\mathcal{R}$ when $\mathcal{O}$ is translated over $\mathcal{M}$. As illustrated by Fig. 3.3, the measure $\mathcal{E}_4$ can thus be written as:

$$\mathcal{E}_4(\mathcal{I}, \mathcal{O}, c) = \sum_{\mathcal{R} \text{ in } \mathcal{O}} \delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \mathcal{L}(\mathcal{O}, \mathcal{R})\Big) \,, \tag{3.7}$$

and $\mathcal{L}(\mathcal{O}, \mathcal{R})$ can formally be written as:

$$\begin{aligned} & \mathcal{L}(\mathcal{O}, \mathcal{R}) \\ = & \{o : \exists M \in \mathcal{M} \text{ such that } o \in \text{DO}(\mathbf{w}(\mathcal{O}, M), \mathcal{R})\} \,. \end{aligned} \tag{3.8}$$

The collection of lists over all regions $\mathcal{R}$ in $\mathcal{O}$ forms the final object template.

3.2.5 Using Bitwise Operations

Inspired by [101], and as shown in Fig. 3.4, we efficiently compute the energy function $\mathcal{E}_4$ using a binary representation of the lists $\mathcal{L}(\mathcal{O}, \mathcal{R})$ and of the dominant orientations $\text{do}(\mathcal{I}, c + \mathcal{R})$. This allows us to compute $\mathcal{E}_4$ with only a few bitwise operations.

By setting n_o, the number of discretized orientations, to 7 we can represent a list $\mathcal{L}(\mathcal{O}, \mathcal{R})$ or a dominant orientation $\text{do}(\mathcal{I}, c + \mathcal{R})$ with one byte i.e. a 8-bit integer. Each of the 7 first bits corresponds to an orientation while the last bit stands for $\perp$.

More exactly, to each list $\mathcal{L}(\mathcal{O}, \mathcal{R})$ corresponds a byte L whose i^{th} bit with $0 \leq i \leq 6$ is set to 1 iff $i \in \mathcal{L}(\mathcal{O}, \mathcal{R})$, and whose 7^{th} bit is set to 1 iff $\perp \in \mathcal{L}(\mathcal{O}, \mathcal{R})$. A byte D can be constructed similarly to represent a dominant orientation $\text{do}(\mathcal{I}, c + \mathcal{R})$. Note that only one bit of D is set to 1. Now the term $\delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \mathcal{L}(\mathcal{O}, \mathcal{R})\Big)$ in Eq.(3.7) can be evaluated very quickly. We have:

$$\delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \mathcal{L}(\mathcal{O}, \mathcal{R})\Big) = 1 \text{ iff } L \otimes D \neq 0 \,, \tag{3.9}$$

where $\otimes$ is the bitwise AND operation.

3.2.6 Using SSE Instructions

The computation of $\mathcal{E}_4$ as formulated in Section 3.2.5 can be further speeded up using SSE operations. In addition to bitwise operations, which are already very fast, SSE technology

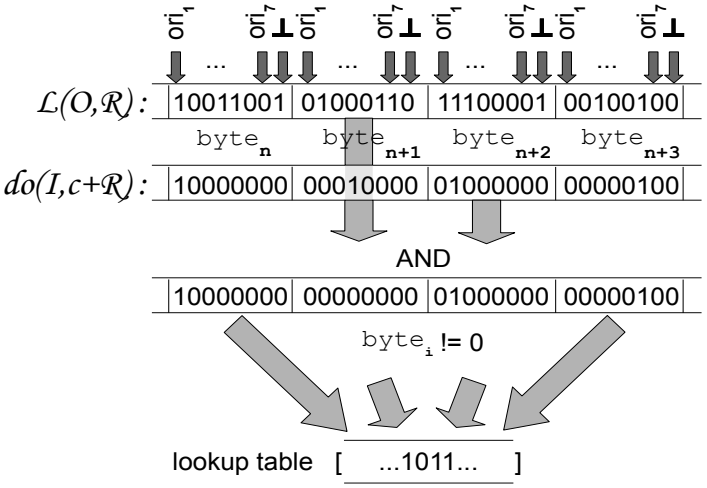


Figure 3.4: Computing the similarity $\mathcal{E}_4$ using bitwise operations and a lookup table that counts how many terms $\delta()$ as in Eq.(3.9) are equal to 1.

```
int energy_function4( __m128i lhs, __m128i rhs )
{
    __m128i a = _mm_and_si128(lhs,rhs);
    __m128i b = _mm_cmpeq_epi8(a,_mm_setzero_si128());

    return lookuptable[_mm_movemask_epi8(b)];
}
```

Listing 3.1: C++ Energy function for 16 regions with 3 SSE instructions and one look-up in a 16-bit-table. Since in SSE there is no comparison on non-equality for unsigned 8-bit integers we have—in contrast to Fig. 3.4—to compare the AND'ed result to zero and count the "0" instead.

allows to perform the same operation on 16 bytes in parallel. Thus, by using the function given in Listing 3.1, the similarity score for 16 regions can be computed with only 3 SSE operations and one lookup-table with 16-bits entries.

Thus, if n denotes the number of regions $\mathcal{R}$, we only have to use $3\left\lceil\frac{n}{16}\right\rceil$ SEE instructions, $\left\lceil\frac{n}{16}\right\rceil$ uses of a lookup table with 16-bits entries and additional $\left\lceil\frac{n}{16}\right\rceil - 1$ "+" operations if the number of regions n is larger than 16. Assuming that each operation has the same computational cost we need $5\left\lceil\frac{n}{16}\right\rceil - 1$ operations for n regions which results in only ≈ 0.3 operations per region.

This method is extremely cache friendly because only successive chunks of 128 bits are processed at a time which holds the number of cache misses low. This is very important because SSE technology is very sensitive to optimal cache alignment. This is probably why, although our energy function is slightly more computationally expensive in theory than [101], we found that our formulation performed 1.5 times faster in practice.

Another advantage of our algorithm compared to [101], however, is that it is very flexible with respect to varying template sizes without loosing the capability of using the

computational capacities very efficiently. In our method, the optimal processor load is reached by multiples of 16 in contrast to [101] that needs multiples of 128 in a possible dynamic SSE implementation. The probability of wasting computational power is therefore much lower using Dot.

3.2.7 Clustering for Efficient Branch and Bound

We can further improve the scalability of our method by exploiting the similarity between different templates representing different objects under different views. The general idea is to build clusters of similar templates—each of them being represented by what we will refer to as a *cluster template*. A cluster template is computed as a bitwise OR operation applied to all the templates belonging to the same cluster. It provides tight upper bounds and can be used in a branch and bound constrained search as described in [56]. By first computing the similarity measure $\mathcal{E}_4$ between the image and the cluster templates at runtime, we can reject all the templates that belong to a cluster template not similar enough to the current image.

We use a bottom-up clustering method: To build a cluster, we start from a template picked randomly among the templates that do not yet belong to a cluster and iteratively search for the templates T that fulfill:

$$\underset{T \notin Cluster_i}{\operatorname{argmin}} \ \max(d_h(C \text{ or } T, T), d_h(C \text{ or } T, C)), \tag{3.10}$$

where d_h is the hamming distance, "or" the bitwise OR operation and C the cluster template before OR'ing. We proceed this way until the cluster has a given number of templates assigned or no templates are left. In the first case, we continue building clusters until every template is assigned to a cluster.

For our approach, this clustering scheme allows faster runtime than the binary tree clustering suggested in [101], as will be shown in Section 3.2.8.3.

3.2.8 Experimental Validation

In the experiments, we compared our approach called Dot (for *Dominant Orientation Templates*) to Affine Region Detectors [68] (Harris-Affine, Hessian-Affine, MSER, IBR, EBR), to patch rectification methods [44, 41, 39] (Leopar, Panter, Gepard) and to the Histograms-of-Gradients (HoG) template matching approach [21].

For HoG, we used our own SSE optimized implementation. In order to detect the correct template from a large template database we replaced the Support Vector Machine mentioned in the original work of HoG by a nearest neighbor search since we want to avoid a training phase and to look for a robust representation instead.

For Dot and HoG we also applied an additional refinement method [8] after detection, as it was done for [44, 41, 39].

We did the performance evaluation on the Oxford Graffiti and on the Oxford Wall image set [68]. Since no video sequence is available, we synthesized a training set by scaling and rotating the first image of the dataset for changes in the viewpoint angle up to 75 degrees and by adding random noise and affine illumination change.

3.2.8.1 Robustness

The matching scores of the different methods are shown in Fig. 3.5(a) for the Graffiti dataset, and in Fig. 3.5(b) for the Wall dataset. As defined in [68], these scores are the ratios between the numbers of correct matches and the smaller numbers of regions detected in one of the two images.

For the affine regions, we first extract the regions using different region detectors and match them using SIFT. Two of them are said to be correctly matched if the overlap error of the normalized regions is smaller than 40%. In our case, the regions are defined as the patches warped by the retrieved transformation. For a fair comparison, we used the same numbers and appearances of templates for the Dot and HoG approaches. We also turned off the final check on the correlation for all patch rectification approaches (Leopar, Panter, Gepard) since there is no equivalent for the affine regions.

Dot and HoG clearly outperform the other approaches by delivering optimal matching results of 100% on the Graffiti image set. For the Wall image set, Dot performs optimal again with a matching rate of 100% while HoG performs worse for larger viewpoint changes.

These very good performances can be explained by the fact that Dot and HoG scan the whole image while the affine regions approach is dependent on the quality of the region extraction. As it will be shown in Section 3.2.8.3, even if it parses the whole image, Dot is fast enough to compete with affine region and patch rectification approaches in terms of computation times.

3.2.8.2 Detection Accuracy

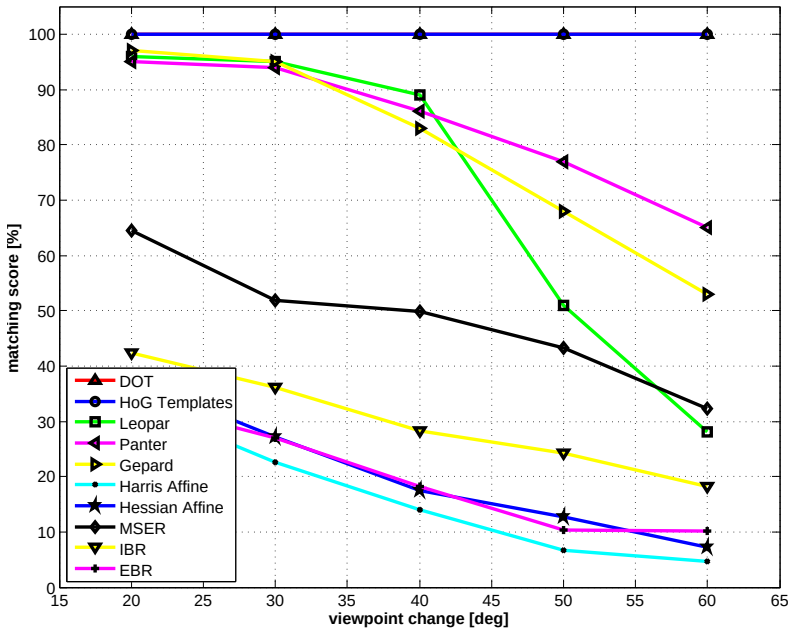
As it was done in [41], in Fig. 3.5(c), we compare the average overlap between the ground truth quadrangles and their corresponding warped versions obtained with Dot, HoG, the patch rectification methods and with the affine regions detectors. We did the experiments for overlap and accuracy on both image sets but due to the similarity of the results and the lack of space we only show the results on the Graffiti image set. Since the Affine Region Detectors deliver elliptic regions we fit quadrangles around these ellipses by aligning them to the main gradient orientation as it was done in [41].

The average overlap is very close to 100% for Dot and HoG, about 10% better than MSER and about 20% better than the other affine region detectors.

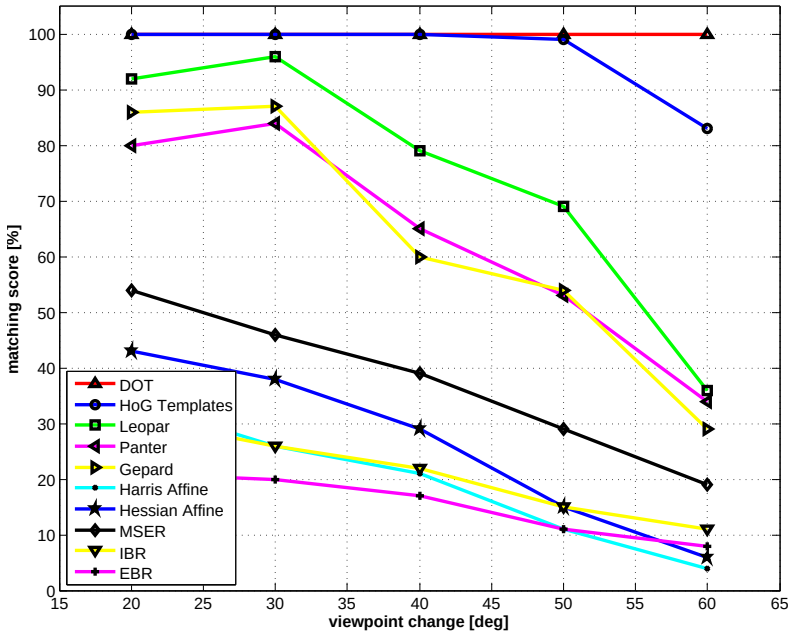
In Fig. 3.5(d), we compare the average error between the quadrangle corners. Once again, Dot, HoG and the patch rectification methods perform similar and thus much better than the affine region detectors. The error of the patch corner is less than three pixels in average.

3.2.8.3 Speed

Although performing similar in terms of robustness and accuracy, Dot clearly outperforms HoG in terms of speed by several magnitudes. In order to compare both approaches, we trained them on the same locations and appearances on a 640×480 image with $|\mathcal{R}| = 121$. The experiment was done on a standard notebook with an Intel Cen-



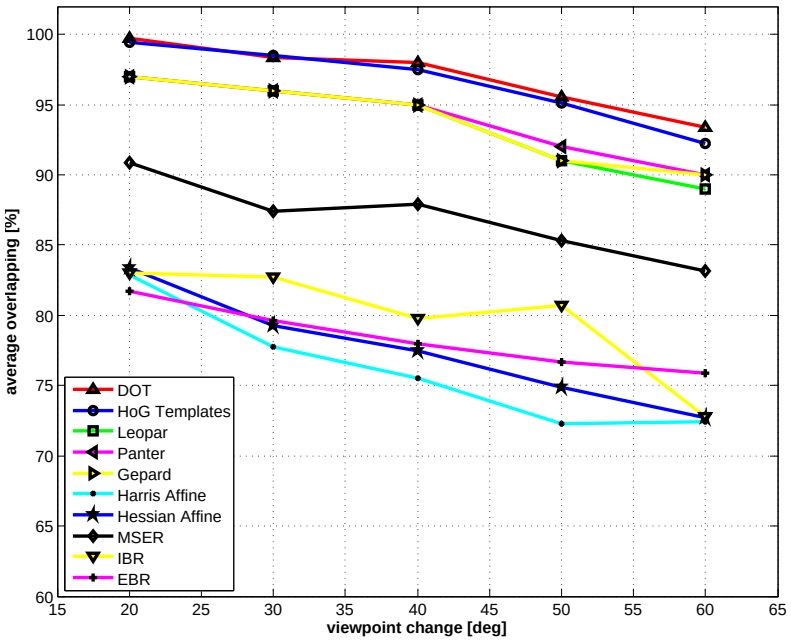
(a)



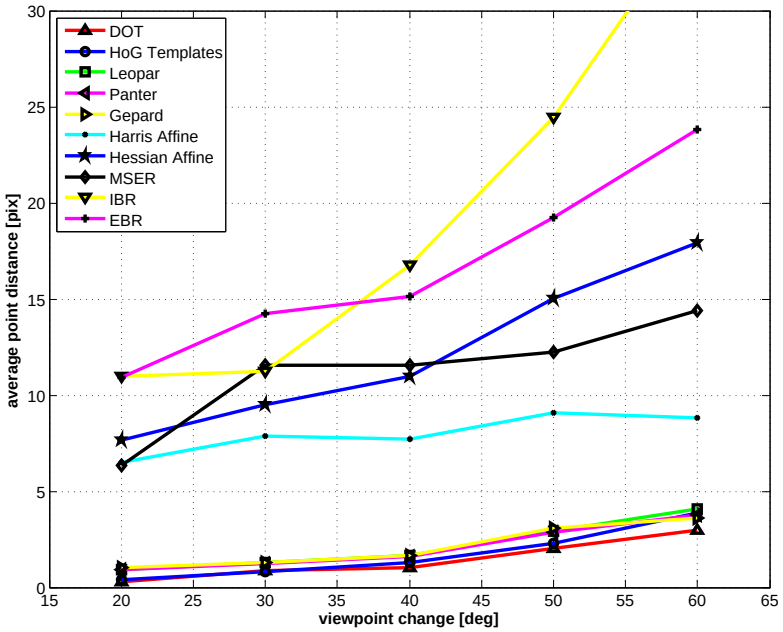
(b)

Figure 3.5: Methods comparisons on the Graffiti and Wall Oxford datasets. **(a)**,**(b)**: Matching scores for Graffiti and Wall sets when increasing the viewpoint angle. Our method is referred as "Dot", and reaches a 100% score on both sets for every angle. These results are discussed in Section 3.2.8.1.

trino Processor Core2Duo with 2.4GHz and 3GB RAM where unoptimized training of one template took 1.8ms and the clustering of about 1600 templates 0.76s. As one can see in Fig. 3.6, when using about 1600 templates Dot is about 310 times faster at runtime



(c)



(d)

Figure 3.5 (cont.): Methods comparisons on the Graffiti and Wall Oxford datasets. **(c)** shows the overlaps between the retrieved and expected regions as an accuracy measure for Graffiti. **(d)** shows the localization error in terms of distance between regions corners. These results are discussed in Section 3.2.8.2.

than our SSE optimized HoG implementation. The reason for this is both the robustness to small deformations that allows Dot to skip most of the pixel locations and the binary

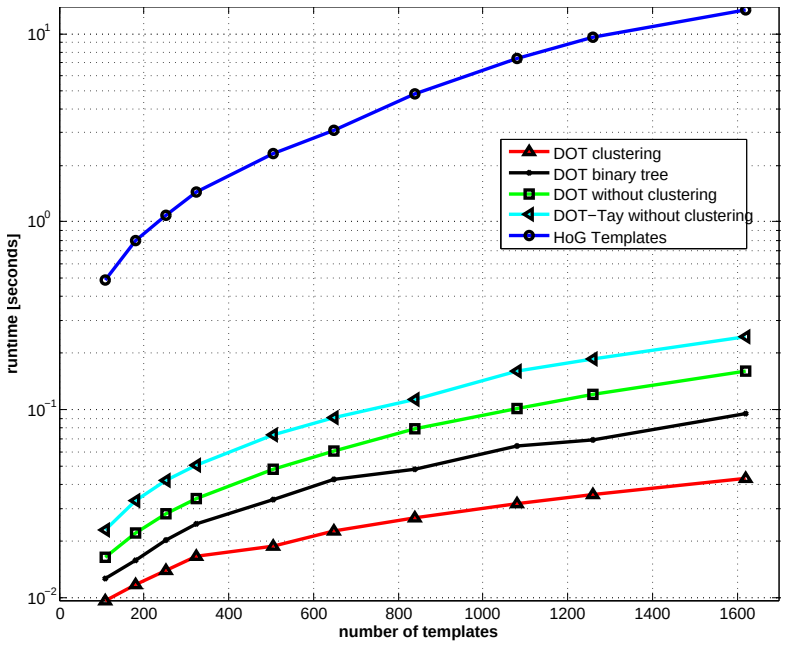


Figure 3.6: Comparison of different methods and cluster schemes with respect to speed. Our method with our cluster scheme performs superior over all other methods and cluster schemes as discussed in Section 3.2.8.3.

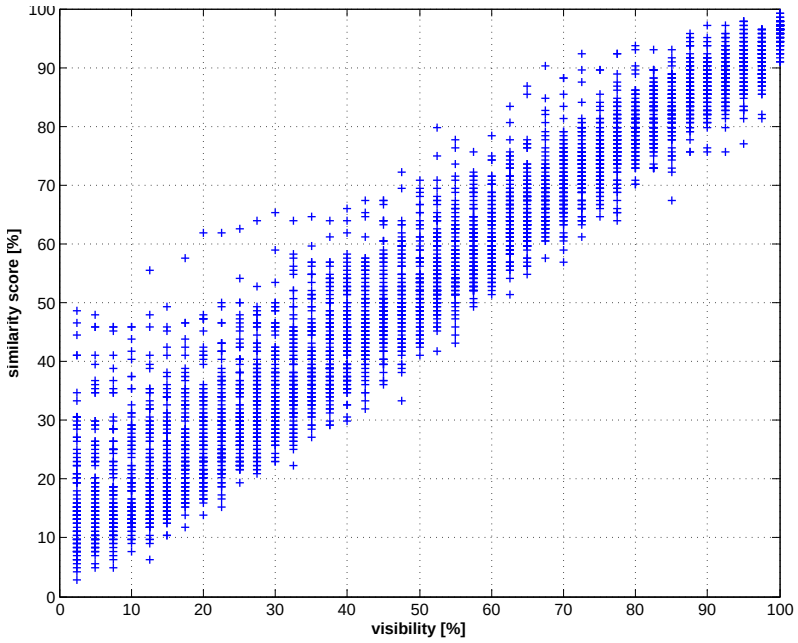


Figure 3.7: In Section 3.2.8.4 we discuss the linear behavior of our method with respect to occlusion.

representation of our templates that enables a fast similarity evaluation.

We also compared our similarity measure to a SSE optimized version of Taylor's ver-

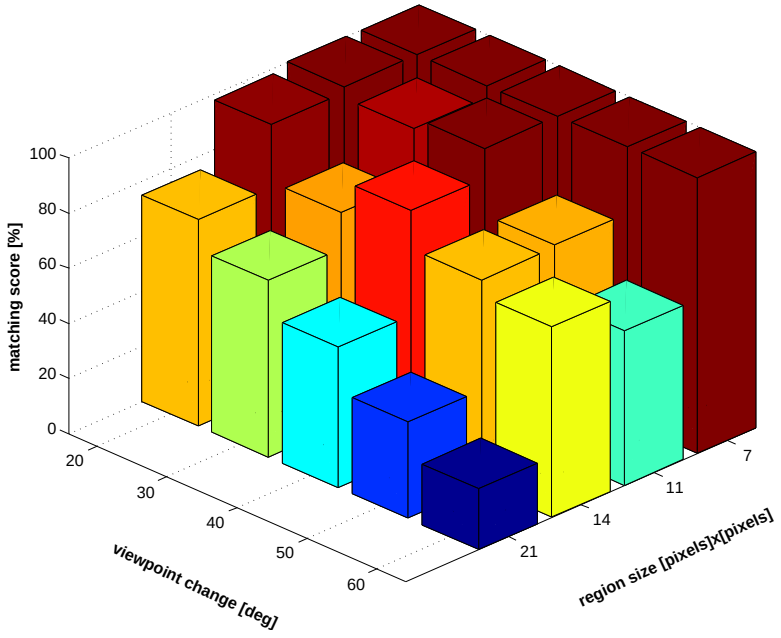


Figure 3.8: $t = 7$ is a good trade-off between speed and robustness (Section 3.2.8.5).

sion [101]. Our approach is constantly about 1.5 times faster than Taylor's. We believe it is due to the cache friendly formulation of $\mathcal{E}_4$ where we successively use sequential chunks of 128 bits at a time while [101] has to jump back and forth within 1024 bits (in case $|\mathcal{R}| = 121$) for successively OR'ing pairs of 128 bit vectors and accumulating the result (for a closer explanation of Taylor's similarity measure please refer to [101]) in a SSE register.

We also did experiments with respect to the different clustering schemes. We compared the approach where no clustering is used to the binary tree of [101] and our clustering described in Section 3.2.7. Surprisingly, our clustering is twice as fast as the binary tree clustering at runtime. Although the matching should behave in $O(log(N))$ time, our implementation of the binary tree clustering behaves linearly up to about 1600 templates as it was also observed by [101]. As the authors of [101] claim, the reason for this might be that there are not enough overlapping templates to fully exploit the potential of their tree structure.

3.2.8.4 Occlusion

Occlusion is a very important aspect in template matching. To test Dot towards occlusion we selected 100 templates on the first image of the Oxford Graffiti image set, added small image deformation, noise and illumination changes and incrementally occluded the template in 2.5% steps from 0% to 100%. The results are displayed in Fig. 3.7. As expected the similarity of our method behaves linearly to the percentage of occlusion. This is a desirable property since it allows to detect partly occluded templates by setting the detection threshold with respect to the tolerated percentage of occlusion.

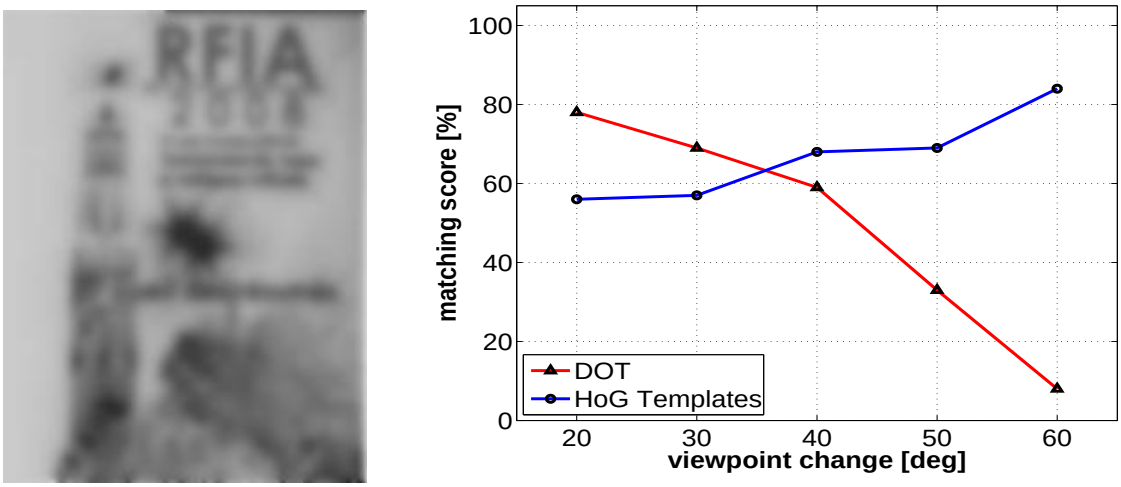


Figure 3.9: Failure Case. When the object does not exhibit strong gradients, like the blurry image on the left, our method performs worse than HoG.

3.2.8.5 Region Size

The size of the region $\mathcal{R}$ is another important parameter. The larger the region $\mathcal{R}$ gets the faster the approach becomes at runtime. However, at the same time as the size of the region increases the discriminative power of the approach decreases since the number of gradients to be considered rises. Therefore, it is necessary to choose the size of the region $\mathcal{R}$ carefully to find a compromise between speed and robustness. In the following experiment on the Graffiti image set we tested the behavior of Dot with respect to the matching score and the size of the region $\mathcal{R}$. The result is shown in Fig. 3.8. As the matching score is still 100% for regions of 7×7 pixels, one can see that the robustness decreases with increasing region size. Although dependent on the texture and on the density of strong gradients within one region $\mathcal{R}$, we empirically found on many different objects that a region size of 7×7 gives very good results.

3.2.8.6 Failure Cases

Fig. 3.9 shows the limitation of our method: To obtain such optimal results as in Fig. 3.5, the templates have to exhibit strong gradients. In case of too smooth or blurry template images, HoG tends to perform better.

3.2.8.7 Applications

Due to the robustness and the real-time capability of our approach, Dot is suited for many different applications including untextured object detection as shown in Fig. 3.11, and planar patches detection as shown in Fig. 3.12. Although neither a final refinement nor any final verification, by contrast with [41] for example, was applied to the found 3D objects, the results are very accurate, robust and stable. Creating the templates for new objects is easy and illustrated by Fig. 3.10.

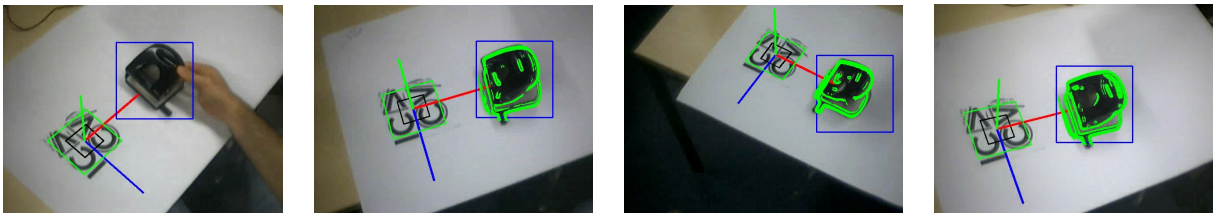


Figure 3.10: Templates creation. To easily define the templates for a new object, we use Dot to detect a known object—the ICCV logo in this case—next to the object to learn in order to estimate the camera pose and to define an area in which the object to learn is located. A template for the new object is created from the first image, and we start detecting the object while moving the camera. When the detection score becomes too low, a new template is created in order to cover the different object appearances when the viewpoint changes.

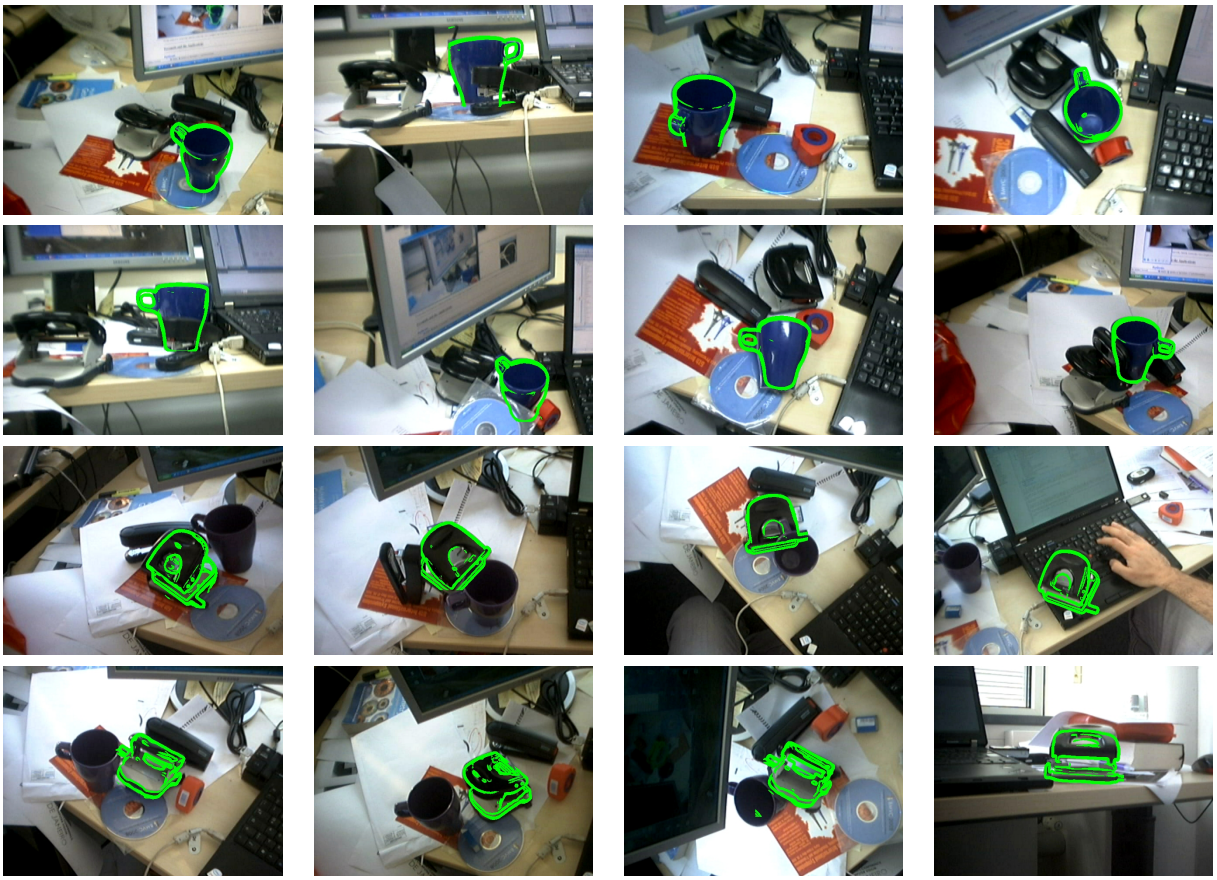


Figure 3.11: Detection of different objects at about 12 fps over a moderately cluttered background. The detections are shown by superimposing the thresholded gradient magnitudes from the object image over the input images.

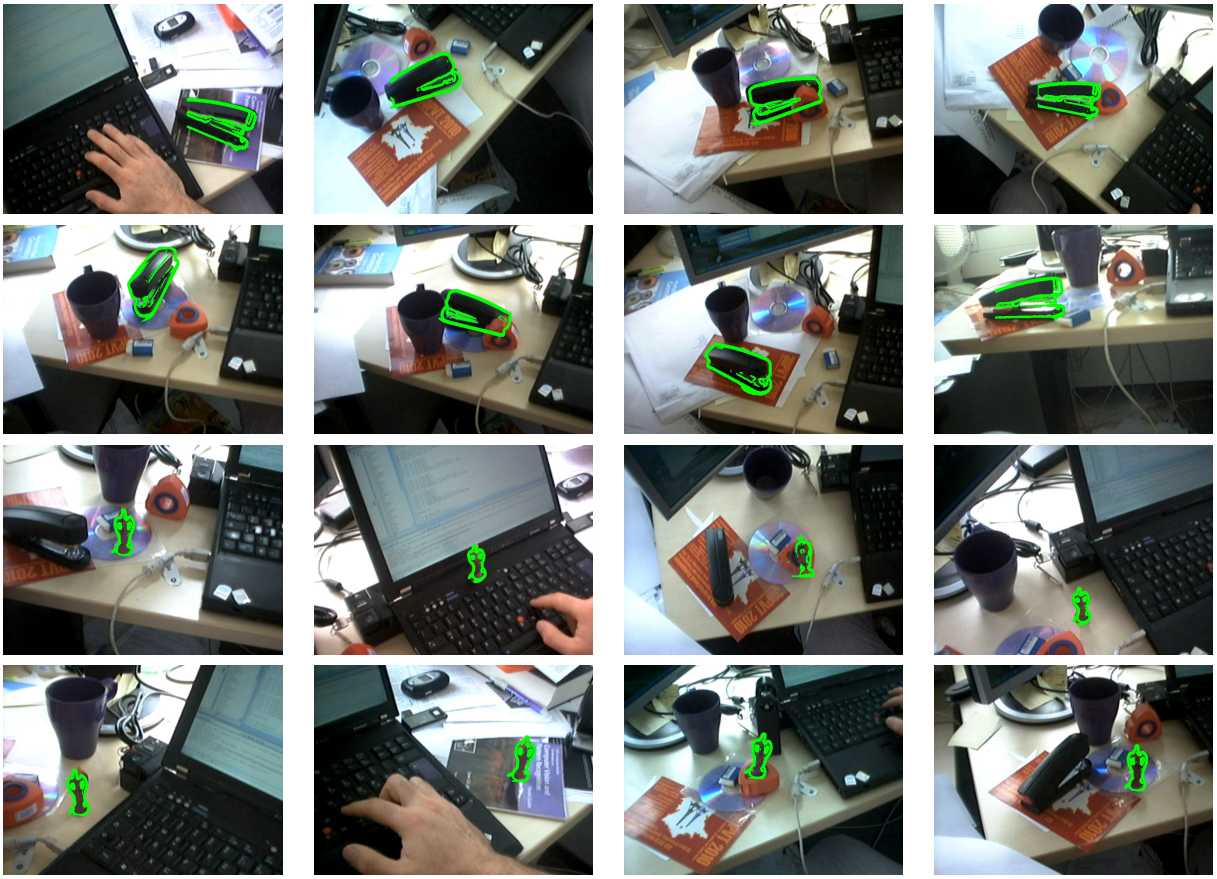


Figure 3.11 (cont.): Detection of different objects at about 12 fps over a moderately cluttered background. The detections are shown by superimposing the thresholded gradient magnitudes from the object image over the input images.

3.3 Line: Response Maps for Real-Time Detection of Texture-Less Objects

While Dot works very well for normal scenes, it suffers — similar to previous template matching approaches [12, 50, 30, 79] — from severe degradation of performance in presence of strong background clutter.Therefore, we describe in this section an alternative template representation that is much more robust with respect to strong background clutter than Dot and handles templates of different size much more efficiently.

In this context, we will show how a new representation of the input image can be built and used to parse the image to quickly and robustly find objects. We will also demonstrate how several different visual cues and modalities can be incorporated into this schema to increase the robustness and to significantly decrease the number of false positives. Additionally, we will show how we implement Line to use modern processor architectures and how to preprocess the different modalities efficiently — i.e. a color image and its registered depth map — to enable real-time performance.

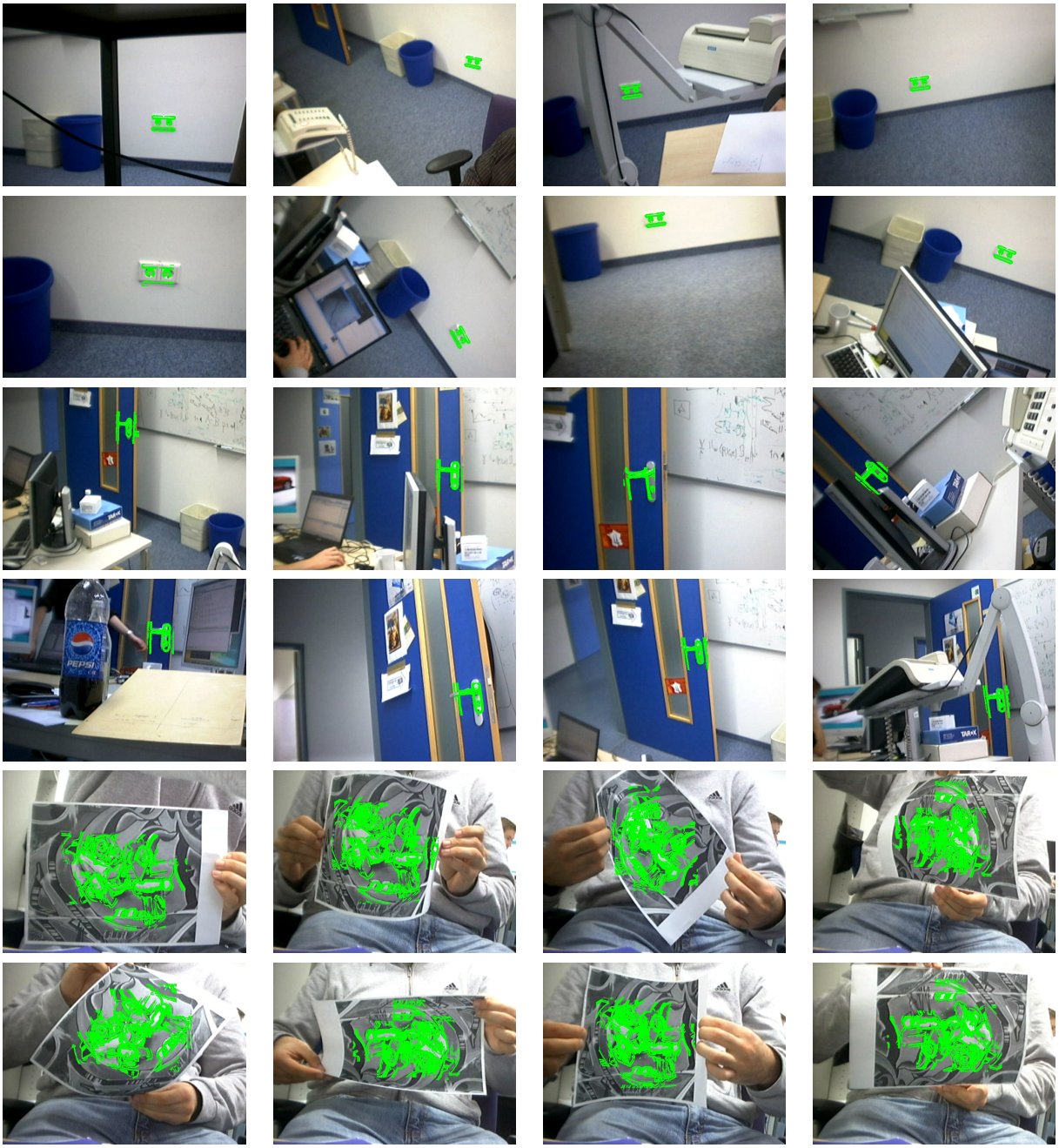


Figure 3.12: Patch 3D orientation estimation. Like Gepard [44], Dot can detect (low-textured) planar patches and provide an estimate of their orientations. Dot is however much more reliable as it does not rely on feature point detection, but parses the image instead. Even in case of deformations it works reliably.

3.3.1 Similarity Measure

Our unoptimized similarity measure can be seen as the measure defined by Steger in [97] modified to be robust to small translations and deformations. Steger suggests to use:

$$\mathcal{E}_{\text{Steger}}(\mathcal{I}, \mathcal{T}, c) = \sum_{r \in \mathcal{P}} \left| \cos(\text{ori}(\mathcal{O}, r) - \text{ori}(\mathcal{I}, c + r)) \right| , \tag{3.11}$$

where $\text{ori}(\mathcal{O}, r)$ is the gradient orientation in radians at location r in a reference image $\mathcal{O}$ of an object to detect. Similarly, $\text{ori}(\mathcal{I}, c + r)$ is the gradient orientation at c shifted by r in the input image $\mathcal{I}$. We use a list, denoted by $\mathcal{P}$, to define the locations r to be considered in $\mathcal{O}$. This way we can deal with arbitrarily shaped objects efficiently. A template $\mathcal{T}$ is therefore defined as a pair $\mathcal{T} = (\mathcal{O}, \mathcal{P})$.

Considering only the gradient orientations and not their norms makes the measure robust to contrast changes, and taking the absolute value of the cosine allows it to correctly handle object occluding boundaries: It will not be affected if the object is over a dark background, or a bright background.

The similarity measure of Eq. (3.11) is very robust to background clutter, but not to small shifts and deformations. A common solution is to first quantize the orientations and to use local histograms like in SIFT [63] or HoG [21]. However this can be unstable when strong gradients are close to the boundaries of the bins. In Dot [46], we kept the dominant orientations of a region. This was faster than building histograms but suffers from the same instability. Another option is to apply Gaussian convolution to the orientations like in DAISY [103], but this would be too slow for our purpose.

We therefore propose a more efficient solution. While we initially designed the new similarity measure to operate on the gradient image [43] only, we recently proposed a generalized framework [42] where we showed how other modalities can be easily incorporated. Therefore, we will directly propose the generalized framework instead of the specialized one.

Given a set of aligned reference images $\{\mathcal{O}_m\}_{m \in \mathcal{M}}$ of the object from a set $\mathcal{M}$ of modalities we redefine a template as $\mathcal{T} = (\{\mathcal{O}_m\}_{m \in \mathcal{M}}, \mathcal{P})$. $\mathcal{P}$ is a list of pairs (r, m) made of the locations r of a discriminant feature in modality m. "Discriminant feature" in this context means that the feature can be well recognized even if it is seen under some viewpoint changes. E.g. in terms of image gradients the larger the norm of the gradient the more discriminant this feature is. Each template is created by extracting for each modality a small set of its most discriminant features from the corresponding reference image and by storing their locations. As shown in Fig. 3.13, the modalities we use in our experiments come from a standard camera and a depth sensor aligned with the camera.

Our similarity measure is a generalization of the measure defined in [43] which is robust to small translations and deformations. It can be formalized as:

$$\mathcal{E}(\{\mathcal{I}_m\}_{m \in \mathcal{M}}, \mathcal{T}, c) = \sum_{(r,m) \in \mathcal{P}} \left(\max_{t \in \mathcal{R}(c+r)} f_m(\mathcal{O}_m(r), \mathcal{I}_m(t)) \right) , \tag{3.12}$$

where $\mathcal{R}(c+r) = \left[c + r - \frac{T}{2}, c + r + \frac{T}{2}\right] \times \left[c + r - \frac{T}{2}, c + r + \frac{T}{2}\right]$ defines the neighborhood of size T centered on location $c+r$ in the input image $\mathcal{I}_m$ and the function $f_m(\mathcal{O}_m(r), \mathcal{I}_m(t))$

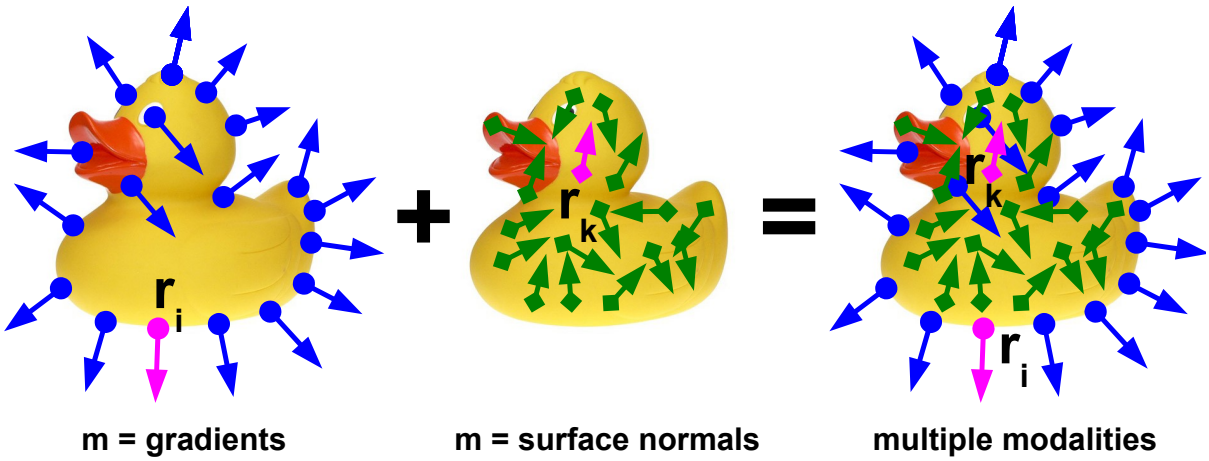


Figure 3.13: A toy duck with different modalities. **Left:** Image gradients are mainly found on the contour. The gradient location r_i is displayed in pink. **Middle:** Surface normals are found on the body of the duck. The normal location r_k is displayed in pink. **Right:** LINE can combine multiple cues which are complementary: gradients are usually found on the object contour while surface normals are found on the object interior

computes the similarity score for modality m between the reference image at location r and the input image at location t. Thus, for each feature we align the local neighborhood exactly to the associated location whereas in DOT [46], BiGG [72], HoG [21] or SIFT [63], the features are adjusted only to some regular grid. As a result, we tremendously gain robustness when using the silhouette of the object. We show below how to compute this measure efficiently.

3.3.2 Spreading the Features

In order to avoid evaluating the max operator in Eq. (3.12) every time a new template must be evaluated against an image location, we first introduce a new binary representation — denoted by $\mathcal{J}_m$ — of the features of modality m around each image location. We will then use this representation together with lookup tables to efficiently precompute these maximal values.

The computation of $\mathcal{J}_m$ is depicted in Fig. 3.14. We first quantize the input data for each modality into a small number of n_o values as done in previous approaches [63, 21, 46]. This allows us to "spread" the data of the input image $\mathcal{I}_m$ around their locations to obtain a new representation of the original image.

For efficiency, we encode the possible combinations of quantized input data spread to a given image location l using a binary string: Each individual bit of this string corresponds to one quantized value, and is set to 1 if this value is present in the neighborhood of l. The strings for all the image locations form the image $\mathcal{J}_m$ on the right part of Fig. 3.14. These strings will be used as indices of lookup tables for fast precomputation of the similarity measure, as it is described in the next subsection.

$\mathcal{J}_m$ can be computed very efficiently: We first compute a map for each quantized

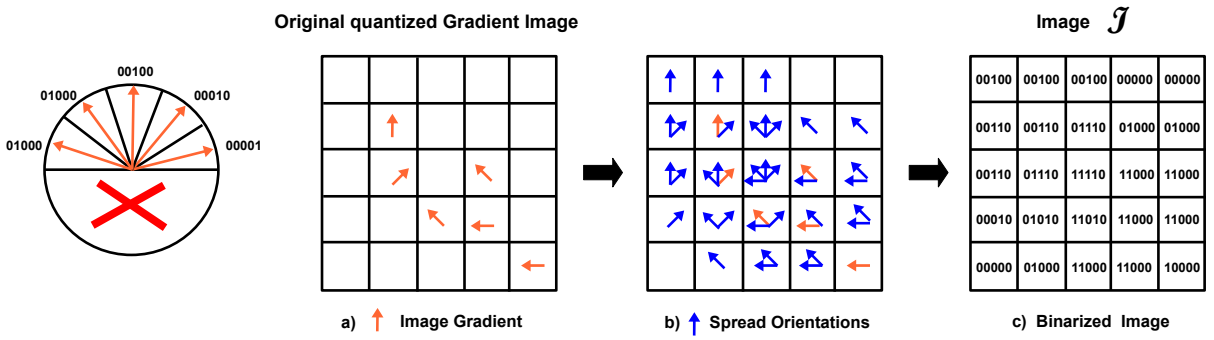


Figure 3.14: Spreading demonstrated on the example of gradient orientations. For simplicity we omit modality m. **Left:** The gradient orientations and their binary code. We do not consider the direction of the gradients. **(a)** The gradient orientations in the input image, shown in orange, are first extracted and quantized. **(b)** Then, the locations around each orientation are also labeled with this orientation, as shown by the blue arrows. This allows our similarity measure to be robust to small translations and deformations. **(c)** $\mathcal{J}$ is an efficient representation of the orientations after this operation, and can be computed very quickly. For this figure, $T = 3$ and $n_o = 5$. In practice, we use $T = 8$ and $n_o = 8$.

feature value, whose values are set to 1 if the corresponding pixel location in the input image has this feature value, and 0 if it does not. $\mathcal{J}_m$ is then obtained by shifting these maps over the range of $\left[-\frac{T}{2}, +\frac{T}{2}\right] \times \left[-\frac{T}{2}, +\frac{T}{2}\right]$ and merging all shifted versions with an OR operation.

3.3.3 Precomputing Response Maps

As shown in Fig. 3.15, $\mathcal{J}_m$ is used together with modality dependent lookup tables to precompute the value of the max operation in Eq. (3.12) for each location and each possible quantized value i in the template. We store the results into 2D maps $\mathcal{S}_{i,m}$ where m is the specific modality. Then, to evaluate the similarity function, we will just have to sum values read from these $\mathcal{S}_{i,m}$s.

We use a lookup table $\tau_{i,m}$ for each modality and for each of the n_o quantized orientations, computed offline as:

$$\tau_{i,m}[\mathcal{L}_m] = \max_{l \in \mathcal{L}_m} f_m(i, l) \;, \tag{3.13}$$

where

- i is the index of the quantized value of modality m. To keep the notations simple, we also use i to represent the corresponding value;
- $\mathcal{L}_m$ is a list of values of a special modality m appearing in a local neighborhood of a value i as described in Section 3.3.2. In practice, we use the integer value corresponding to the binary representation of $\mathcal{L}_m$ as an index to the element in the lookup table.

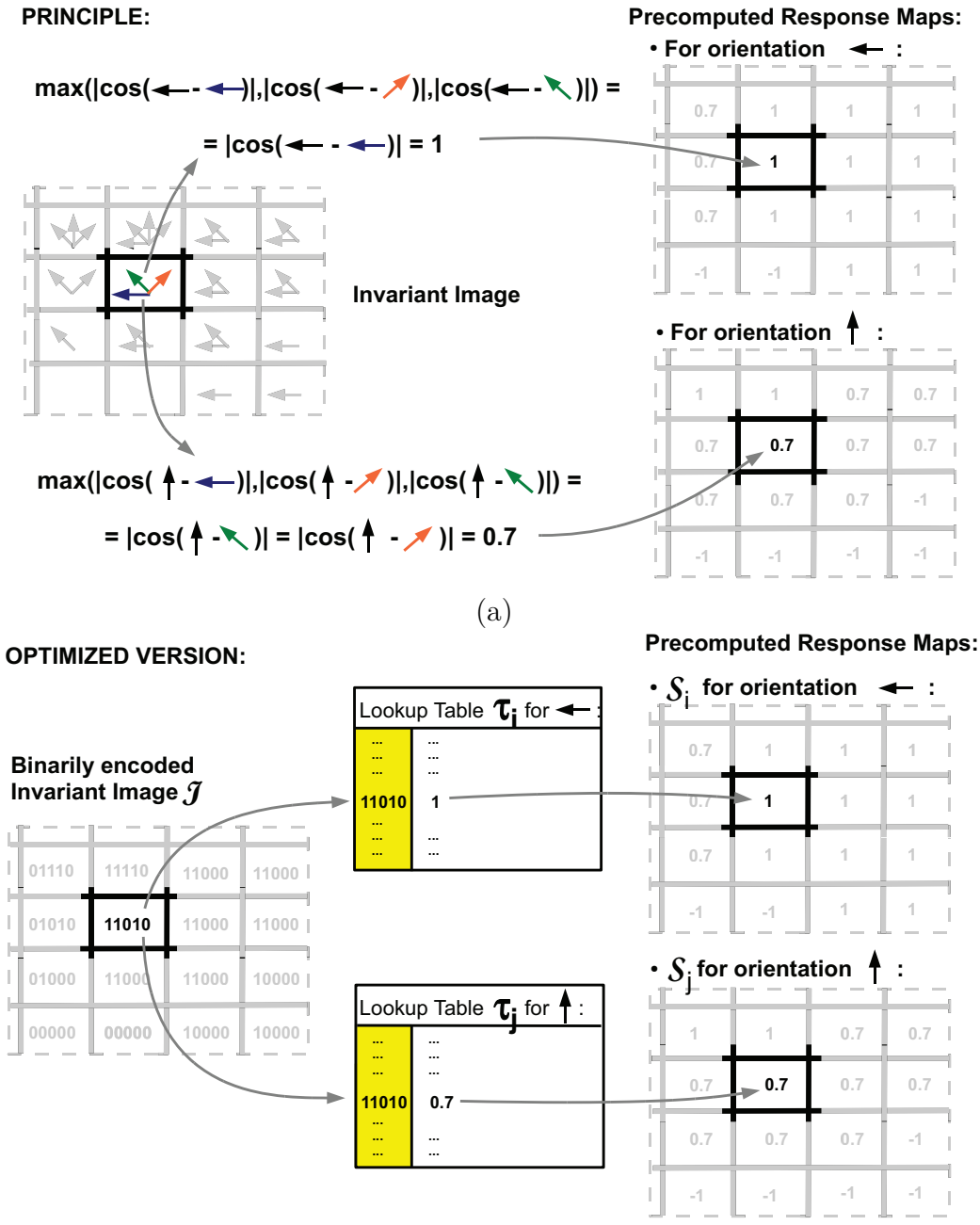


Figure 3.15: Precomputing the Response Maps $\mathcal{S}_i$ on the example of gradient orientation. For simplicity we omit modality m. **(a):** There is one response map for each quantized orientation. They store the maximal similarity between their corresponding orientation and the orientations ori_j already stored in the "Invariant Image". **(b):** This can be done very efficiently by using the binary representation of the list of orientations in $\mathcal{J}$ as an index to lookup tables of the maximal similarities.

For quantized value i we can now compute the value at each location c of the response

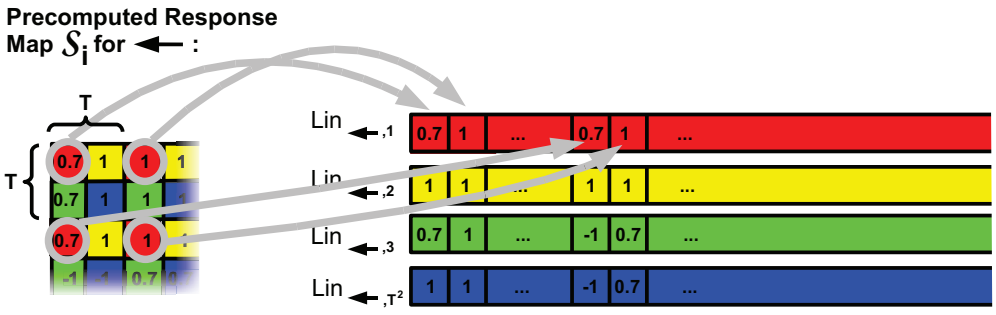


Figure 3.16: Restructuring the way the response images $\mathcal{S}_{i,m}$ are stored in memory. For simplicity we omit modality m within the figure. The values of one image row that are T pixels apart on the x axis are stored next to each other in memory. Since we have T^2 such linear memories per response map, and n_o quantized orientations, we end up with $T^2 \cdot n_o$ different linear memories.

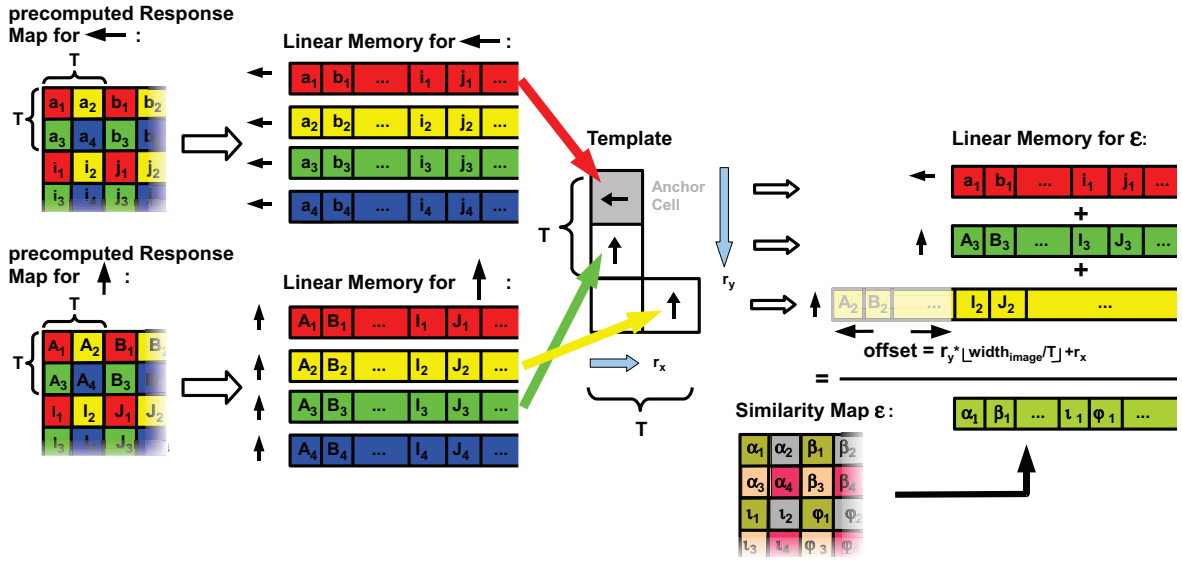


Figure 3.17: Using the linear memories on the example of gradient orientation. We can compute the similarity measure over the input image for a given template by adding up the linear memories for the different orientations of the template, shifted by an amount depending on the locations in the template. Performing these additions with parallel SSE instructions further speeds up the computation.

map $\mathcal{S}_{i,m}$ as:

$$\mathcal{S}_{i,m}(c) = \tau_{i,m}[\mathcal{J}_m(c)] \,. \tag{3.14}$$

Finally, the similarity measure of Eq. (3.12) can be evaluated as:

$$\mathcal{E}(\{\mathcal{I}_m\}_{m\in\mathcal{M}}, \mathcal{T}, c) = \sum_{(r,m)\in\mathcal{P}} \mathcal{S}_{\mathcal{O}_m(r),m}(c+r) \,. \tag{3.15}$$

Since the maps $\mathcal{S}_{i,m}$ are shared between the templates, matching several templates against the input image can be done very fast once they are computed.

3.3.4 Linearizing the Memory for Parallelization

Thanks to Eq. (3.15), we can match a template against the whole input image by only adding the values in the response maps $\mathcal{S}_{i,m}$. However, one of the advantages of spreading the quantized values as was done in Section 3.3.2 is that it is sufficient to do the evaluation only every T^{th} pixel without reducing the recognition performance. If we want to exploit this property efficiently, we have to take into account the architecture of modern computers.

Modern processors do not only read one data value at a time from the main memory but several ones simultaneously, called a *cache line*. Accessing the memory at random places results in a *cache miss* and slows down the computations. On the other hand, accessing several values from the same cache line is very cheap. As a consequence, storing data in the same order as they are read speeds up the computations significantly. In addition, this allows parallelization: For instance, if 8-bit values are used as it is the case for our $\mathcal{S}_{i,m}$ maps, SSE instructions can perform operations on 16 values in parallel.

Therefore, as shown in Fig. 3.16, we store the precomputed response maps $\mathcal{S}_{i,m}$ into memory in a cache-friendly way: We restructure each response map so that the values of one row that are T pixels apart on the x axis are now stored next to each other in memory. We continue with the row which is T pixels apart on the y axis once we finished with the current one.

Finally, as described in Fig. 3.17, computing the similarity measure for a given template at each sampled image location can be done by adding the linearized memories with an appropriate offset computed from the locations r in the templates.

3.3.5 Modality Extraction

We now turn to how we handle the different modalities and demonstrate this on image and depth data.

3.3.5.1 Image Cue

We chose to consider image gradients because they proved to be more discriminant than other forms of representations [63, 97] and are robust to illumination change and noise. Additionally, image gradients are often the only reliable image cue when it comes to texture-less objects. Considering only the orientation of the gradients and not their norms makes the measure robust to contrast changes, and taking the absolute value of cosine between them allows it to correctly handle object occluding boundaries: It will not be affected if the object is over a dark background, or a bright background.

To increase robustness, we compute the orientation of the gradients on each color channel of our input image separately and for each image location use the gradient orientation of the channel whose magnitude is largest. Given an RGB color image $\mathcal{I}$, we compute the gradient orientation map $\mathcal{I}_{\mathcal{G}}(x)$ at location x with

$$\mathcal{I}_{\mathcal{G}}(x) = \text{ori}(\hat{\mathcal{C}}(x)) \tag{3.16}$$

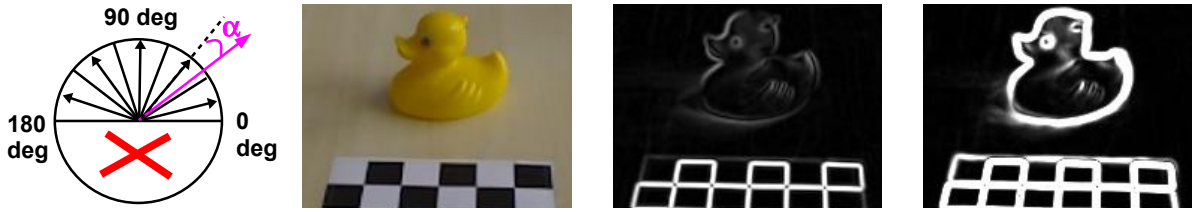


Figure 3.18: **Upper Left:** Quantizing the gradient orientations: the pink orientation is closest to the second bin. **Upper right:** A toy duck with a calibration pattern **Lower Left:** The gradient image computed on a gray value image. The object contour is hardly visible. **Lower right:** Gradients computed with our method. Details of the object contours are clearly visible.

where

$$\hat{\mathcal{C}}(x) = \underset{\mathcal{C}\in\{R,G,B\}}{\operatorname{argmax}} \left\| \frac{\partial \mathcal{C}}{\partial x} \right\| \tag{3.17}$$

and R, G, B are the RGB channels of the corresponding color image. Our similarity measure is then:

$$f_{\mathcal{G}}(\mathcal{O}_{\mathcal{G}}(r), \mathcal{I}_{\mathcal{G}}(t)) = |cos(\mathcal{O}_{\mathcal{G}}(r) - \mathcal{I}_{\mathcal{G}}(t))| \tag{3.18}$$

where $\mathcal{O}_{\mathcal{G}}(r)$ is the gradient orientation map of the reference image at location r and $\mathcal{I}_{\mathcal{G}}(t)$ the gradient orientation map of the current image at location t respectively.

In order to quantize the gradient orientation map we omit the gradient direction, consider only the gradient orientation and divide the orientation space into n_0 equal spacings as shown in Fig. 3.18. To make the quantization robust to noise, we assign to each location the gradient whose quantized orientation occurs most often in a 3×3 neighborhood. We also keep only the gradients whose norms are larger than a small threshold. The whole unoptimized process takes about 31ms on the CPU for a VGA image.

3.3.5.2 Depth Cue

Similar to the image cue, we decided to use quantized surface normals computed on a dense depth field for our template representation as shown in Fig. 3.19. They allow us to represent both close and far objects while fine structures are preserved.

In the following, we propose a method for the fast and robust estimation of surface normals in a dense range image. Around each pixel location x, we consider the first order Taylor expansion of the depth function $\mathcal{D}(x)$:

$$\mathcal{D}(x + dx) - \mathcal{D}(x) = dx^{\top} \nabla \mathcal{D} + h.o.t. \tag{3.19}$$

Within a patch defined around x, each pixel offset dx yields an equation that constrains the value of $\nabla \mathcal{D}$, allowing to estimate an optimal gradient $\hat{\nabla \mathcal{D}}$ in a least-square sense.

Figure 3.19: **Upper Left:** Quantizing the surface normals: the pink surface normal is closest to the precomputed surface normal v_4. It is therefore put into the same bin as v_4. **Upper right:** A person standing in an office room. **Lower Left:** The corresponding depth image. **Lower right:** Surface normals computed with our approach. Details are clearly visible and depth discontinuities are well handled. We removed the background for visibility reasons.

This depth gradient corresponds to a 3D plane going through three points X, X_1 and X_2:

$$X = \vec{v}(x)\mathcal{D}(x), \tag{3.20}$$

$$X_1 = \vec{v}(x + [1,0]^\top)(\mathcal{D}(x) + [1,0]\hat{\nabla\mathcal{D}}), \tag{3.21}$$

$$X_2 = \vec{v}(x + [0,1]^\top)(\mathcal{D}(x) + [0,1]\hat{\nabla\mathcal{D}}). \tag{3.22}$$

where $\vec{v}(x)$ is the vector along the line of sight that goes through pixel x and is computed from the internal parameters of the depth sensor. The normal to the surface at the 3D point that projects on x can be estimated as the normalized cross-product of $X_1 - X$ and $X_2 - X$.

However this would not be robust around occluding contours, where the first order approximation of Eq. (3.19) no longer holds. Inspired by bilateral filtering, we ignore the contributions of pixels whose depth difference with the central pixel is above a threshold. In practice, this approach effectively smooths out quantization noise on the surface, while still providing meaningful surface normal estimates around strong depth discontinuities. Our similarity measure is then defined as the dot product of the normalized surface normals:

$$f_{\mathcal{D}}(\mathcal{O}_{\mathcal{D}}(r), \mathcal{I}_{\mathcal{D}}(t)) = \mathcal{O}_{\mathcal{D}}(r)^\top \mathcal{I}_{\mathcal{D}}(t) \tag{3.23}$$

where $\mathcal{O}_{\mathcal{D}}(r)$ is the normalized surface normal map of the reference image at location r and $\mathcal{I}_{\mathcal{D}}(t)$ the normalized surface normal map of the current image at location t.

Finally, as shown in Fig. 3.19, we measure the angles between the computed normal and a set of precomputed vectors to quantize the normal directions into n_0 bins. These vectors are arranged in a right circular cone shape originating from the peak of the cone pointing towards the camera. To make the quantization robust to noise, we assign to each location the quantized value that occurs most often in a 5×5 neighborhood. The whole process is very efficient and needs only 14ms on the CPU and less than 1ms on the GPU.

3.3.6 Computation Time Study

In this section we compare the numbers of operations required by the original method from [97] and the Line method we propose. In order to do a fair comparison, we only assume one modality used in Line.

The time required by $\mathcal{E}_{\text{Steger}}$ from [97] to evaluate R templates over an $M \times N$ image is $M \cdot N \cdot R \cdot G \cdot (S + A)$, where G the average number of gradients in a template, S the time to evaluate the similarity function between two gradient orientations and, A the time to add two values.

Changing $\mathcal{E}_{\text{Steger}}$ to Eq. (3.12) and making use of $\mathcal{J}_{\mathcal{G}}$ leads to a computation time of $M \cdot N \cdot T^2 \cdot O + \frac{M \cdot N}{T^2} \cdot R \cdot G \cdot (L + A)$, where L is the time needed for accessing once the lookup tables τ_i and O is the time to OR two values together. The first term corresponds to the time needed to compute $\mathcal{J}_{\mathcal{G}}$, the second one to the time needed to actually compute Eq. (3.12).

Precomputing the response maps $\mathcal{S}_{i,\mathcal{G}}$ further changes the complexity of Line to $M \cdot N \cdot (T^2 \cdot O + n_o \cdot L) + \frac{M \cdot N}{T^2} \cdot R \cdot G \cdot A$.

Linearizing our memory allows the additional use of parallel SSE instructions. In order to run 16 operations in parallel, we approximate the response values in the lookup tables using bytes. The final complexity of our algorithm is then $M \cdot N \cdot (T^2 \cdot O + (n_o + 1) \cdot L) + \frac{M \cdot N}{16T^2} \cdot R \cdot G \cdot A$.

In practice we use $T = 8$, $M = 480$, $N = 640$, $R > 1000$, $G \approx 100$ and $n_o = 8$. If we assume for simplicity that $L \approx A \approx O \approx 1$ time unit, this leads to a speed improvement compared to the original energy formulation $\mathcal{E}_{\text{Steger}}$ of a factor $T^2 \cdot 16(1 + S)$ if we assume that the number of templates R is large. Note that we did not incorporate the cache friendliness of Line since it is very hard to model. Still, since [97] evaluates the similarity of two orientations with the normalized dot product of the two corresponding gradients, S can be set to 3 and we obtain a theoretical gain in speed of at least a factor of 4096.

3.3.7 Experimental Validation

We compared our approaches, which we call Line-MOD [42] (for "multimodal-Line"), Line-2D as introduced in [43], which uses only the image intensities and a variant that we call Line-3D, that uses only the depth map, to DOT [46] and HOG [21]. For HOG, we used our own optimized implementation and replaced the Support Vector Machine mentioned in the original work of HOG by a nearest neighbor search. In this way, we can use it as a robust representation and quickly learn new templates as with the other methods. The experiments were performed on one processor of a standard notebook with an Intel Centrino Processor Core2Duo with 2.4 GHz and 3 GB of RAM. For obtaining the image and the depth data we used the Primesense$^{(tm)}$ PSDK 5.0 device.

3.3.8 Robustness

We used six sequences made of over 2000 real images each. Each sequence presents illumination and large viewpoint changes over heavy cluttered background. Ground truth is obtained with a calibration pattern attached to each scene that enables us to know the actual location of the object. The templates were learned over homogeneous background.

We consider the object to be correctly detected if the location given back is within a fixed radius of the ground truth position. As depicted in the left columns of Fig. 3.24 and Fig. 3.25, Line-MOD always outperforms all the other approaches and shows only few

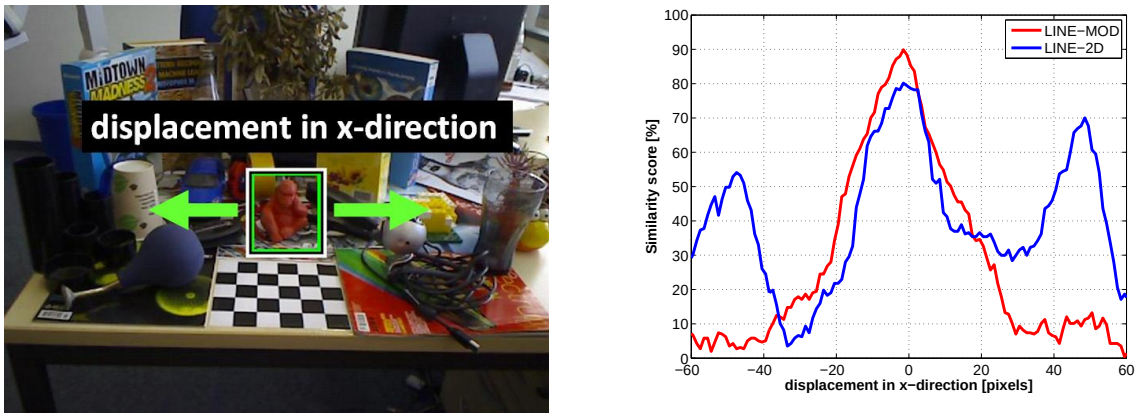


Figure 3.20: Combining many modalities results in a more discriminative response function. Here we compare Line-MOD against Line-2D on the shown image. We plot the response function of both methods with respect to the true location of the monkey. One can see that the response of Line-MOD exhibits a single and discriminative peak whereas Line-2D has several peaks which are of comparable height. This is one explanation why Line-MOD works better and produces fewer false positives.

false positives. We believe that this is due to the complementarity of the object features that compensate for the weaknesses of each other. The superiority of Line-MOD becomes even more obvious in Table 3.1: If we set the threshold for each approach to allow for 97% true positive rate and only evaluate the hypothesis with the largest response, we obtain for Line-MOD a high detection rate with a very small false positive rate. This is in contrast to Line-2D, where the true positive rate is often over 90%, but the false positive rate is not negligible, which makes expensive post-processing necessary. In Line-MOD, using only the response with the largest value might be sufficient in most cases.

One reason for this high robustness is the good separability of the multimodal approach as shown in the middle of Fig. 3.24 and Fig. 3.25: one can see that a specific threshold—about 80 in our implementation—separates almost all true positives well from almost all false positives. This has several advantages. First, we will detect almost all instances of the object by setting the threshold to this specific value. Second, we also know that almost every returned template with a similarity score above this specific value is a true positive. And third, the threshold is always around the same value which supports the conclusion that it might also work well for other objects. One hint why the novel multimodal approach has such a good separability property is given in Fig. 3.20. One can see that the response function has only one clear peak around the true location of the object while Line-2D shows other peaks with almost the same height.

3.3.9 Speed

Learning new templates only requires extracting and storing multimodal features, which is almost instantaneous. Therefore, we concentrate on runtime performance.

The runtimes given in Fig. 3.21 show that the general Line approach is real-time and can parse a VGA image with over 3000 templates with about 10 fps on the CPU. The

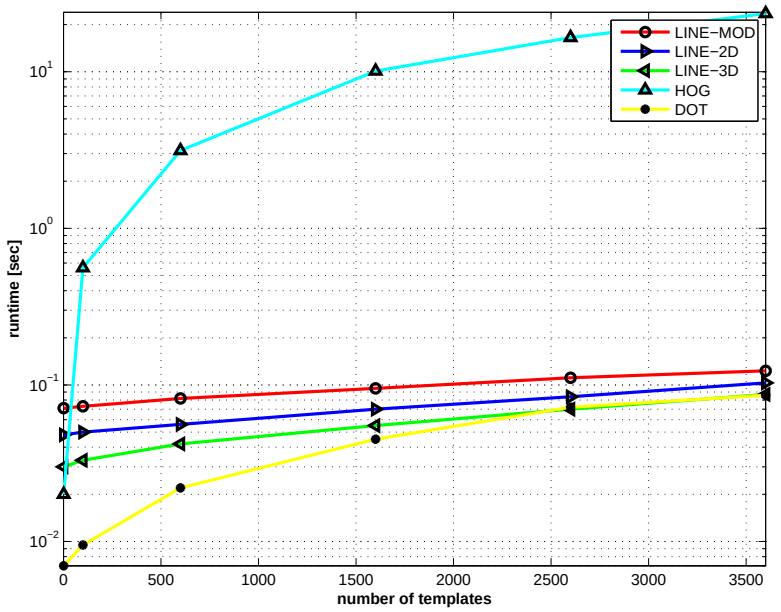


Figure 3.21: LINE runs in real-time and can parse a 640×480 image with over 3000 templates with about 10 fps.

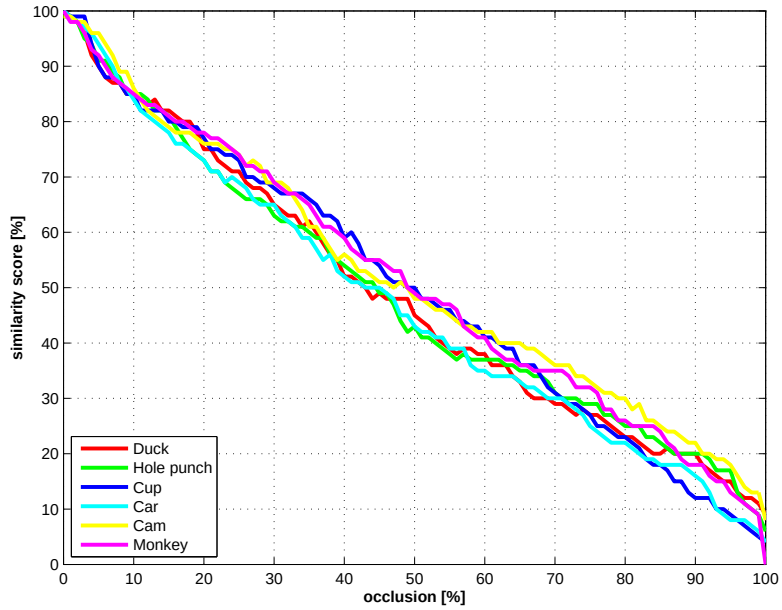


Figure 3.22: LINE-MOD is linear with respect to occlusion.

small difference of computation times between LINE-MOD and LINE-2D and LINE-3D

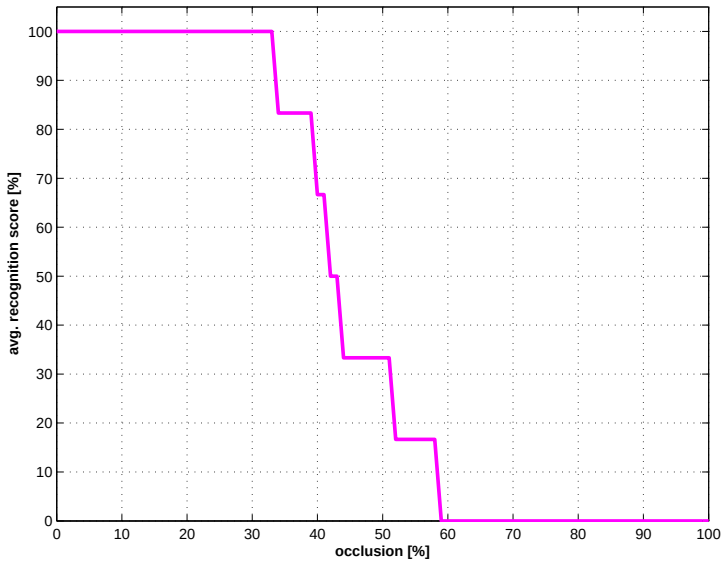


Figure 3.23: Average recognition score of Line-MOD for the six objects of Sec.3.3.8 with respect to occlusion.

Sequence (# pics)	Line-MOD	Line-2D	Line-3D	HOG	DOT
Monkey (2164)	**97.9%–0.3%**	50.8%–49.1%	86.1%–13.8%	51.8%–48.2%	8.6%–91.4%
Camera (2173)	**97.5%–0.3%**	92.8%–6.7%	61.9%–38.1%	18.2%–81.8%	1.9%–98.0%
Car (2162)	**97.7%–0.0%**	96.9%–0.4%	95.6%–2.5%	44.1%–55.9%	34.0%–66.0%
Cup (2193)	**96.8%–0.5%**	92.8%–6.0%	88.3%–10.6%	81.1%–18.8%	64.1%–35.8%
Duck (2223)	**97.9%–0.0%**	91.7%–8.0%	89.0%–10.0%	87.6%–12.4%	78.2%–21.8%
Holepunch (2184)	**97.0%–0.2%**	96.4%–0.9%	70.0%–30.0%	92.6%–7.4%	87.7%–12.0%

Table 3.1: True and false positive rates for different thresholds on the similarity measure of different methods. In some cases no hypotheses were given back so the sum of true and false positives can be lower than 100%. Line-MOD obtains very high recognition rates at the cost of almost no false positives, and outperforms all the other approaches. The corresponding best values are shown in bold print.

comes from the slightly slower preprocessing step that includes the two preprocessing steps of Line-2D and Line-3D.

DOT is initially faster than Line but becomes slower as the number of templates increases. This is because the runtime of Line is independent of the template size whereas the runtime of DOT is not. Therefore, to handle larger objects DOT has to use larger templates which makes the approach slower once the number of templates increases.

3.3.10 Occlusion

We also tested the robustness of Line-MOD with respect to occlusion. We added synthetic noise and illumination changes to the images, incrementally occluded the six different objects of Section 3.3.8 and measured the corresponding response values. As expected, the

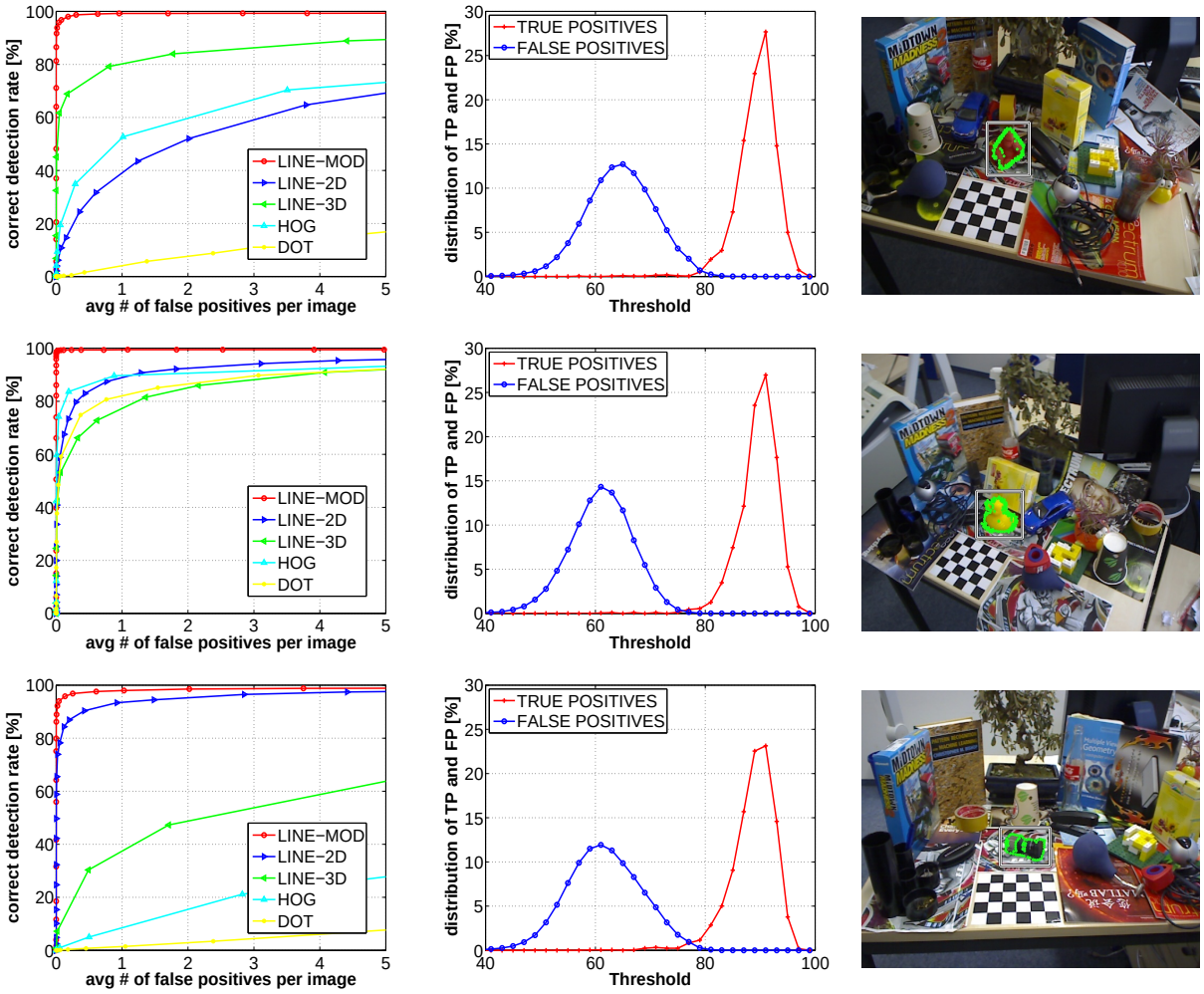


Figure 3.24: Comparison of Line-MOD with Line-2D, which is based on gradients [43], Line-3D, which is based on normals, DOT [46] and HOG [21] on real 3D objects. Each row corresponds to a different sequence (made of over 2000 images each) on heavy cluttered background: A monkey, a duck and a camera. The approaches were learned on a homogeneous background. **Left:** Percentage of true positives plotted against the average percentage of false positives. The multimodal templates provide about the same recognition rates for all objects while the other approaches have a much larger variance depending on the object type. Line-MOD outperforms the other approaches in most cases. **Middle:** The distribution of true and false positives plotted against the threshold. They are well separable from each other. **Right:** One sample image of the corresponding sequence shown with the object detected by Line-MOD.

similarity measure used by Line-MOD behaves linearly in the percentage of occlusion as reported in Fig. 3.22. This is a desirable property since it allows detection of partly occluded templates by setting the detection threshold with respect to the tolerated percentage of occlusion. We also experimented with real scenes where we first learned our six objects in front of a homogeneous background and then added heavy 2D and 3D background clutter. For recognition we incrementally occluded the objects. We define our object as correctly recognized if the template with the highest response is found within

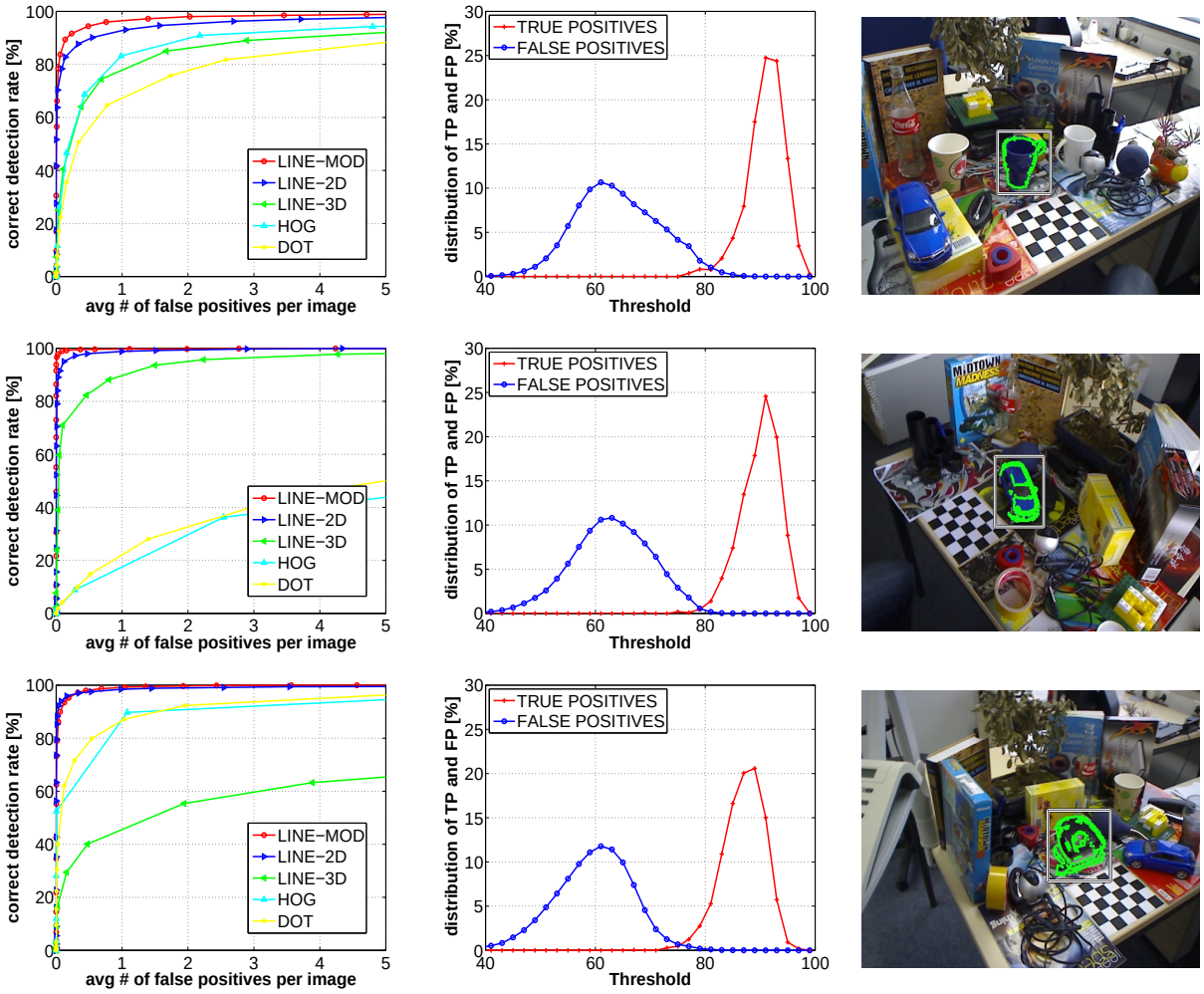


Figure 3.25: Same experiments as shown in Fig. 3.24, however for different objects: a cup, a car and a hole punch. For each object we tested the approaches on over 2000 images each.

a fixed radius of the ground truth object location. The average recognition result is displayed in Fig. 3.23: Even with over 30% occlusion Line-MOD is still able to recognize objects.

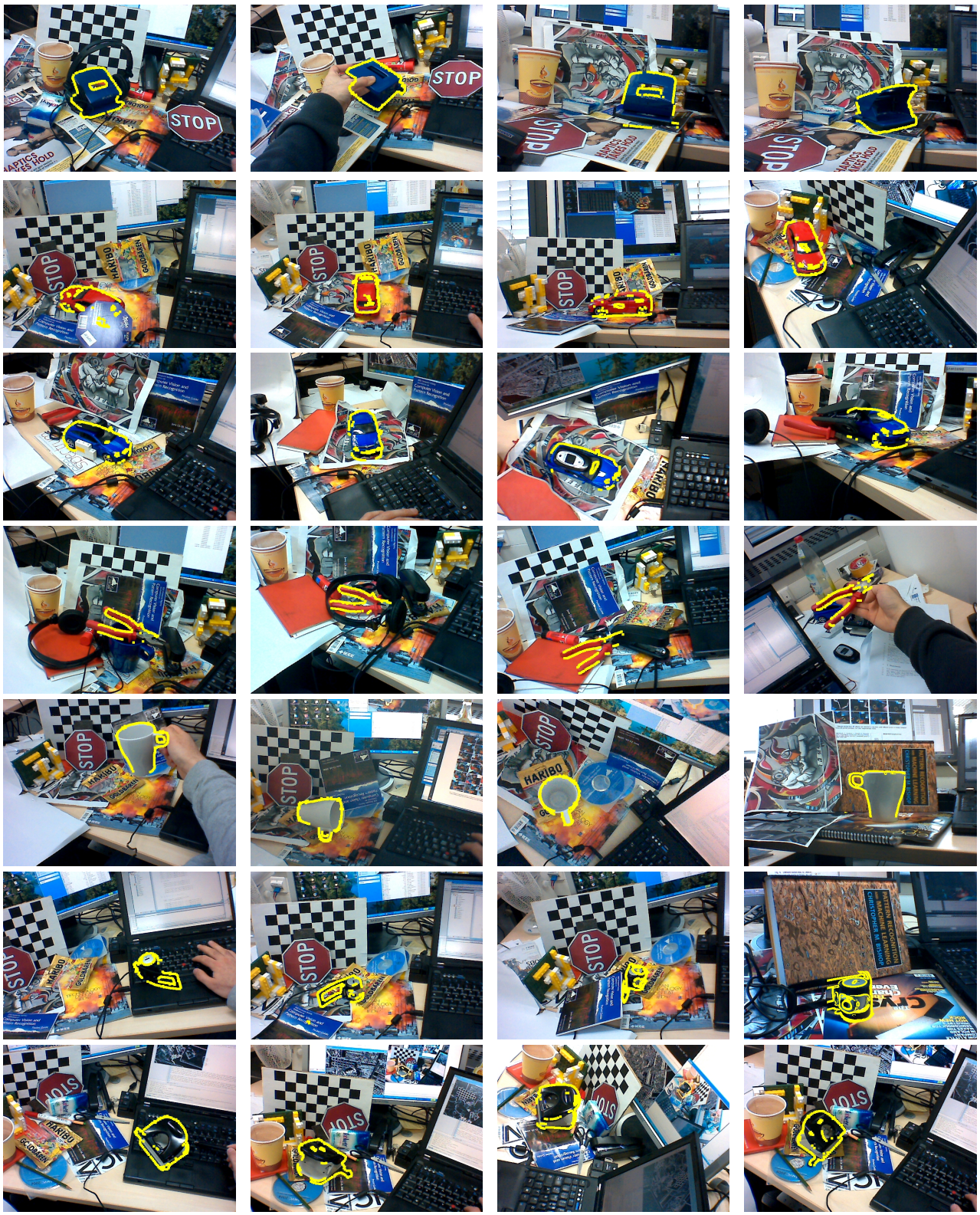


Figure 3.26: Different texture-less 3D objects are detected with Line-2D in real-time under different poses on heavily cluttered background with partial occlusion.

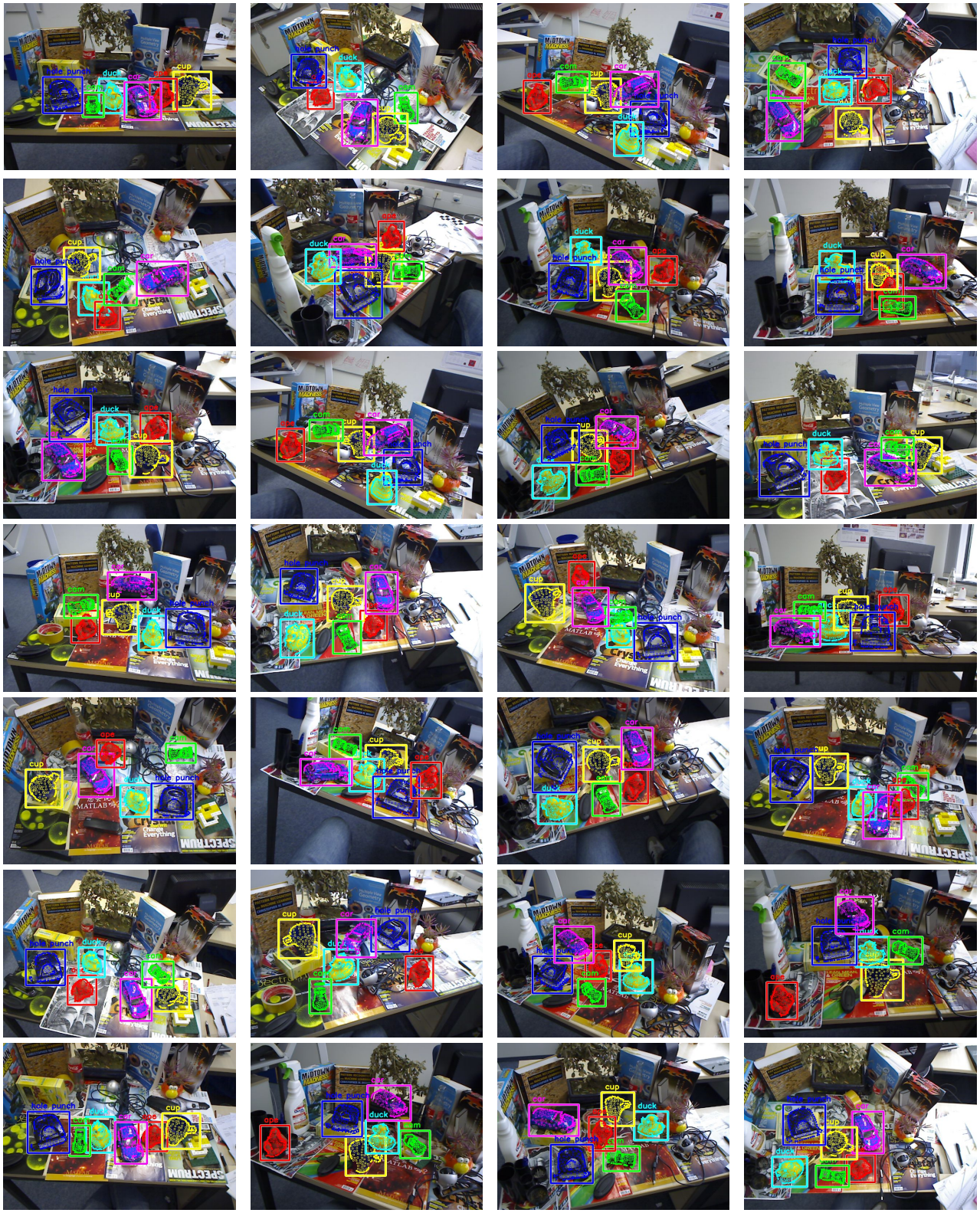


Figure 3.27: Different texture-less 3D objects detected simultaneously in real-time by our Line-MOD method under different poses on heavily cluttered background with partial occlusion.

CHAPTER

FOUR

OUTLOOK

Although the results of the methods presented in this thesis are a good start towards the robust and efficient detection of low-textured and texture-less objects, they need further improvement. In the following, we will identify open questions and define possible future research fields.

Low-Textured Objects: One major issue with LEOPAR and GEPARD is their reduced capability to handle a large amount of patches at the same time. This is especially important for SLAM (Simultaneous Localization and Matching) approaches [22, 54, 23, 76] where the system has to be reinitialized once it lost track. Since one of the major goals of SLAM is to build up large maps of the environment, initialization should work anywhere in these maps. Therefore, it is necessary to simultaneously deal with a large number of patches. Currently, LEOPAR is only able to deal with few hundreds and GEPARD with only some dozens of patches in real-time. These numbers need to be drastically increased. Concerning LEOPAR, it would be worth evaluating the use of Signatures [14], BRIEF [15] and ORB [89] since they are similar to the Ferns classifier [81] we use and able to handle a large amount of keypoints. GEPARD, on the other hand side, could be made more efficient by using the approximate best-bin-first algorithm on KD-trees [7] which is now often applied to SIFT descriptors of size 128, a size similar to the size of our mean patches.

Another direction of future research concerning patch perspective rectification approaches could be opened up by the generalization of our planar patch-based methods to arbitrary 3D patch shapes. This gives the chance to handle not only planar objects well but also non-planar ones which would drastically increase the application field of these approaches.

Texture-less Objects: Despite the robustness of DOT and LINE (and its variants), the two methods still suffer some major problems. These problems restrict the professional use of DOT and LINE in challenging scenarios like industrial assembly lines or robotic applications in households. Partial occlusion, sensitivity to missing depth data, scalability, learning from 3D models and accurate pose estimation of the detected objects are some

of the main problems and challenges. In the following, we will discuss each of these points more in detail.

Partial occlusion is generated if other objects occlude parts of the object to be found. As partial occlusion is one of the main points that hinder current detection approaches to be applied in challenging environments like industrial or daily human environments, we consider research in this direction as most important.

In addition to partial occlusion, another open research direction addresses missing depth data on local parts of the object. This effect is similar to partial occlusion but usually generated by highly reflective or black surfaces that don't allow low cost commodity hardware like the Kinect to compute valid depth data. Possible solutions to this are (a) using an additional stereo-setup that computes the depth where the Kinect fails or by (b) intelligently interpolate the depth for missing data.

Another very important issue to be solved is the scalability of the two approaches. Although, we can currently detect and estimate the pose of few objects in (almost) real-time, our system scales linearly with the number of detected objects. This avoids using our methods in settings where many different objects have to be detected simultaneously. Here, the challenge is to keep the runtime sublinear w.r.t. the number of objects while producing only few false negatives. A first trial to speed up the processing time could consist in employing the graphical unit of modern computers. However, although quite promising especially for the highly linearized and parallelized Line, this approach would not make the runtime sublinear. In this context, it would be of value to evaluate Line-MOD against an extended version of Dot where we also other modalities (e.g. depth) are added (Dot would then turn to Dot-MOD). Current research is showing promising results with Dot-MOD and this might be important since Dot uses a different matching strategy as Line: Dot is evaluating all templates at one image position first before it moves on to the next location whereas Line computes the similarity of one template for the whole image first before moving on to the next template. Here, the matching strategy of Dot could be advantageous in terms of speed since it could simply use additional constraints (e.g. using the current depth at a position compared with the depths from which the templates were generated) to drastically speed up the template search without breaking the efficient matching pipeline.

The versions of Line (and its variants) and Dot have some additional disadvantages. First, templates are learned online, which is difficult to control and results in spotty coverage of viewpoints. Second, the pose output by Line and Dot is only approximately correct, since a template covers a range of views around its viewpoint. And finally, the performance, while extremely good, still suffers from the presence of false positives.

To cover these points, our main idea is that a 3D model of the object can be exploited to remedy these deficiencies. Note that accurate 3D models can now be created very quickly [75, 84, 110, 76], and requiring a 3D model beforehand is not a disadvantage anymore. For industrial applications, a detailed 3D model often exists before the real object is even created.

Given a 3D model of an object, we could generate templates that cover a full view hemisphere by regularly sampling viewpoints of the 3D model. We could also use the 3D model to obtain a fine estimate of the object pose, starting from the one provided

Figure 4.1: We created the synthetic 3D models of these 15 texture-less objects in order to test our detection framework.

by the templates. In addition, having the 3D model would allow us to perform simple checks on depth and color to remove false positives, by checking if the object under the recovered pose aligns well with the depth map and the color is consistent with the model. This would result in a system that significantly improves the original DOT and LINE implementations in performance, while providing accurate pose for applications.

In Fig. 4.2, you can observe first results of using 3D models in the LINE-MOD framework. As you can see, 15 different objects 4.1 are correctly detected under large viewpoint and strong illumination changes. Here, we make use of an automatic template learning approach which uses only the synthetic 3D model of the object. In addition, this 3D model is also used to remove false positives by checking the depth and the color values. The preliminary results were sent to [47].

Finally, another related field of future research w.r.t. DOT and LINE is the topic of object class detection. While in this thesis we have mainly covered object instance detection, it would be interesting to apply DOT and LINE to object class recognition. This is reasonable because DOT and LINE can be seen as descriptors and thus be employed in existing object class recognition frameworks. In addition, the inherent invariance of these two methods to small deformations could help a lot to increase the robustness of such generic detection schemes.

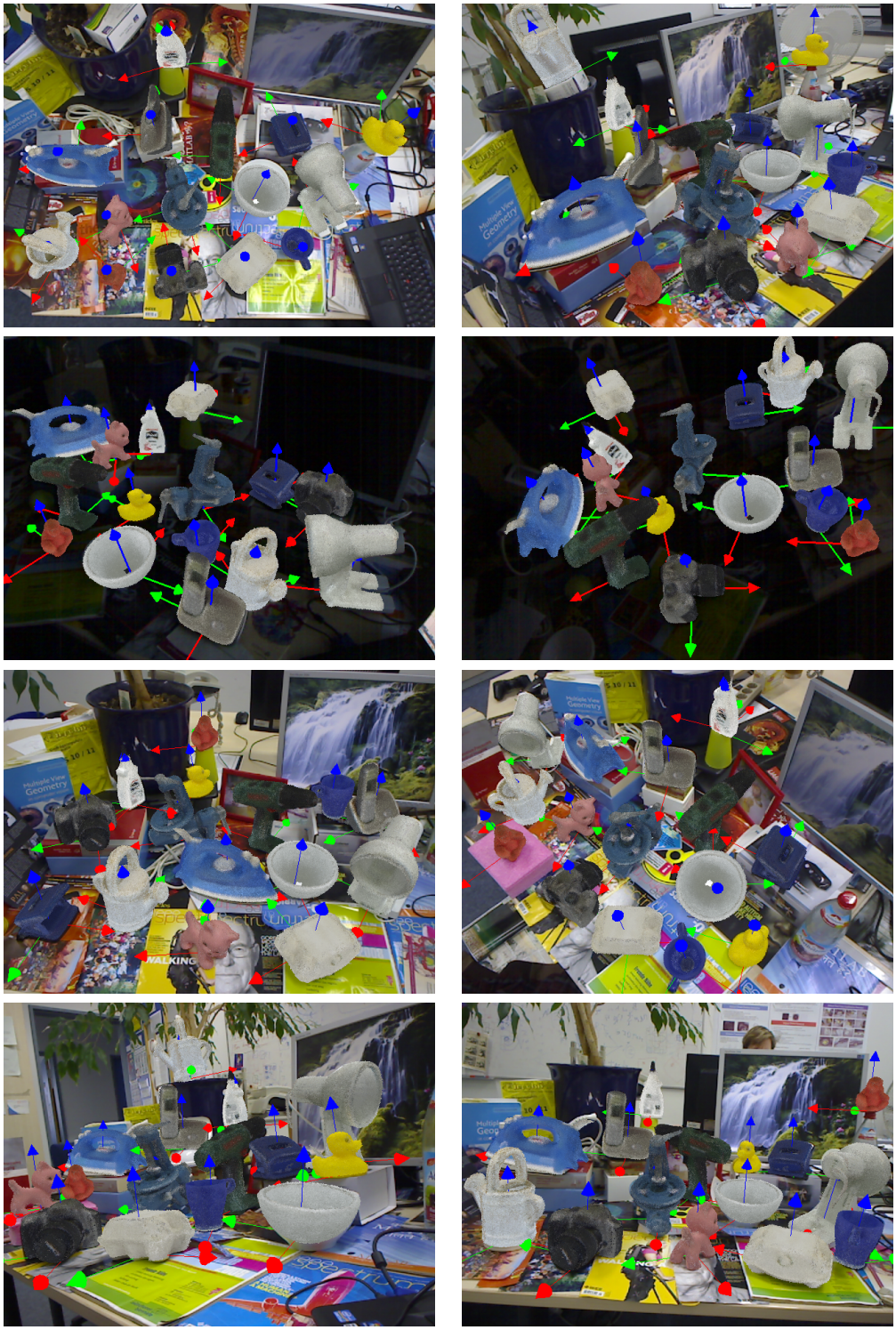


Figure 4.2: In this figure we use the 3D models of 15 different texture-less 3D objects. Once they are automatically learned, we simultaneously detected them under different poses on heavy cluttered background with partial occlusion and illumination changes. Each detected object is augmented with its 3D model. We also show the corresponding coordinate systems.

CHAPTER

FIVE

CONCLUSION

In this thesis, we have introduced four novel real-time detection and pose estimation approaches for low-textured and texture-less objects. We have also presented the characteristics and challenges of these types of objects and discussed their application in various fields. In the following, we will summarize our contributions and insights w.r.t. each corresponding object type.

Low-Textured Objects: In this line of work we showed that one available feature point and the close surrounding is already enough to detect an object and to estimate its pose. This is in contrast to state-of-the-art approaches where usually many correctly matched feature points are needed to provide robust detection.

Furthermore, we showed that including pose estimation within the recognition process considerably improves the robustness and the accuracy of the results of object detection, and this makes our approaches highly desirable. Thanks to a two-step algorithm, it is possible to get matching sets that usually contain no outliers and are only dependent on spatially limited data. Even low-textured objects can therefore be well detected and their pose well estimated.

We also showed that a Fern based classifier is able to recognize the keypoints pose in a very fast manner that allows to track several hundred patches very accurately in real-time. We also showed that the simultaneous estimation of keypoint identities and poses is more reliable but slower than the two separate steps undertaken consecutively. In addition, we showed how to build a one-way descriptor based on geometric blur in real-time that quickly, robustly and accurately estimates the pose of feature points and therefore is appropriate for applications where real-time learning is mandatory.

We demonstrated in various experiments the improved performance compared to previous state-of-the-art methods and demonstrated our approach on many applications including simple 3D tracking-by-detection, SLAM applications, low-textured object detection and deformable objects registration. However, many other applications could also benefit from it, such as object recognition, image retrieval or robot localization.

Texture-less Objects: In this line of work we showed that template matching in combination with an efficient sliding window approach considerably improves the recognition rate of texture-less objects. Hereby, we put emphasis on both gradient based and multi modality templates.

We introduced a new binary template representation — called Dot — based on locally dominant gradient orientations that is invariant to small image deformations. It can very reliably detect untextured 3D objects using relatively few templates from different viewpoints in real-time. An efficient clustering allows high-speed detection of objects even the template database is large.

We also presented another method — called Line — and demonstrated how it is able to exploit different modalities for robust real-time object detection. It is able to correctly detect 3D texture-less objects in real-time under heavy background clutter, illumination changes and noise with almost no false positives. We showed how to efficiently preprocess image and depth data to robustly integrate both cues. Additionally, we showed how we take advantage of the architecture of modern computers to build a fast but very discriminant representation of the input images that can be used to consider a few thousands of arbitrarily sized and arbitrarily shaped templates in real-time. This leads us to another important contribution of this thesis i.e. the insight that nowadays we can not neglect the power of modern hardware when we want to find new solutions to existing real-time computer vision problems. For that, one has to extensively use new concepts of modern computer architecture like caching, parallelization or vector programming.

We have shown that our two new approaches perform superior to state-of-the-art methods with respect to the combination of recognition rate and speed. Moreover, the template creation is fast and easy in both cases, does not require a training set, only a few exemplars, and can be done interactively. While Dot has proven to be considerably faster than Line when using only small templates, we have also demonstrated that Line outperforms Dot with respect to the combination of recognition rate and speed especially when using large templates in heavily cluttered environments. Furthermore, we have shown that using multiple different modalities drastically improves the recognition performance and significantly reduces the number of false positives in regard to single modality template matching.

Finally, we gave an outlook to future work based on the approaches presented here in this thesis. Among others, we discussed the partial occlusion problem, the missing depth value problem and the scaling issue. We also proposed an detection framework for general texture-less objects based on synthetic 3D models and showed some preliminary results.

LIST OF FIGURES

AUTHORED AND CO-AUTHORED PUBLICATIONS

[Alt et al., 2010] Alt, N., Hinterstoisser, S., and Navab, N. (2010). Rapid selection of reliable templates for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Georgel et al., 2007] Georgel, P., Schroeder, P., Benhimane, S., Hinterstoisser, S., Appel, M., and Nassir, N. (2007). An Industrial Augmented Reality Solution For Discrepancy Check . In *International Symposium on Mixed and Augmented Reality*.

[Hinterstoisser et al., 2008a] Hinterstoisser, S., Benhimane, S., Lepetit, V., Fua, P., and Navab, N. (2008a). Simultaneous Recognition and Homography Extraction of Local Patches With a Simple Linear Classifier. In *British Machine Vision Conference*.

[Hinterstoisser et al., 2007] Hinterstoisser, S., Benhimane, S., and Navab, N. (2007). N3M: Natural 3D Markers for Real-Time Object Detection and Pose Estimation. In *IEEE International Conference on Computer Vision*.

[Hinterstoisser et al., 2008b] Hinterstoisser, S., Benhimane, S., Navab, N., Fua, P., and Lepetit, V. (2008b). Online Learning of Patch Perspective Rectification for Efficient Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Hinterstoisser et al., 2011] Hinterstoisser, S., Cagniart, C., Holzer, S., Ilic, S., Konolige, K., Navab, N., and Lepetit, V. (2011). Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *IEEE International Conference on Computer Vision*.

[Hinterstoisser et al., 2012] Hinterstoisser, S., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. (2012). Gradient response maps for real-time detection of texture-less objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Hinterstoisser et al., 2009] Hinterstoisser, S., Kutter, O., Navab, N., Fua, P., and Lepetit, V. (2009). Real-Time Learning of Accurate Patch Rectification. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Hinterstoisser et al., 2010a] Hinterstoisser, S., Lepetit, V., Benhimane, S., Fua, P., and Navab, N. (2010a). Learning Real-Time Perspective Patch Rectification. *International Journal of Computer Vision*.

[Hinterstoisser et al., 2010b] Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., and Navab, N. (2010b). Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Holzer et al., 2009] Holzer, S., Hinterstoisser, S., Ilic, S., and Navab, N. (2009). Distance Transform Templates for Object Detection and Pose Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

REFERENCES

[1] Amit, Y., Geman, D., and Fan, X. A Coarse-To-Fine Strategy for Multi-Class Shape Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2004).

[2] Baker, S., Datta, A., and Kanade, T. Parameterizing Homographies. Tech. rep., CMU, 2006.

[3] Baker, S., and Matthews, I. Lucas-Kanade 20 Years On: A Unifying Framework. *International Journal of Computer Vision 56*, 3 (March 2004), 221–255.

[4] Ballard, D. H. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition* (1981).

[5] Baumberg, A. Reliable Feature Matching Across Widely Separated Views. In *IEEE Conference on Computer Vision and Pattern Recognition* (2000), pp. 774–781.

[6] Bay, H., Tuytelaars, T., and Van Gool, L. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision* (2006).

[7] Beis, J., and Lowe, D. Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In *IEEE Conference on Computer Vision and Pattern Recognition* (1997).

[8] Benhimane, S., and Malis, E. Homography-Based 2D Visual Tracking and Servoing. *International Journal of Robotics Research 26*, 7 (July 2007), 661–676.

[9] Berg, A., and Malik, J. Geometric Blur for Template Matching. In *IEEE Conference on Computer Vision and Pattern Recognition* (2002).

[10] B.J.Frey, and Jojic, N. Transformation invariant clustering using the em algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2003).

[11] Blake, G., Dreslinski, R., and Mudge, T. A Survey of Multicore Processors. *IEEE Signal Processing Magazine 26* (November 2009).

[12] Borgefors, G. Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1988).

[13] Bosch, A., Zisserman, A., and Munoz, X. Image Classification Using Random Forests. In *IEEE International Conference on Computer Vision* (2007).

[14] Calonder, M., Lepetit, V., and Fua, P. Keypoint Signatures for Fast Learning and Recognition. In *European Conference on Computer Vision* (2008).

[15] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision* (2010).

[16] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 6 (1986).

[17] Carvey, A., Gouldstone, J., Vedurumudi, P., Whiton, A., and Ishii, H. Rubber Shark as User Interface. In *International Conference for Human-Computer Interaction* (2006).

[18] chandrasekhar, V., Takacs, G., Chen, D., S.Tsai, Grzeszczuk, R., and Girod, B. CHoG: Compressed Histogram of Gradients A Low Bit-Rate Feature Descriptor. In *IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[19] Chum, O., and Matas, J. Geometric Hashing with Local Affine Frames. In *IEEE Conference on Computer Vision and Pattern Recognition* (2006), pp. 879–884.

[20] Cooley, J. W., and Tukey, J. W. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation* (1965).

[21] Dalal, N., and Triggs, B. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (2005).

[22] Davison, A. Real-Time Simultaneous Localisation and Mapping With a Single Camera. In *IEEE International Conference on Computer Vision* (October 2003), pp. 1403–1410.

[23] Eade, E., and Drummond, T. Unified Loop Closing and Recovery for Real Time Monocular SLAM. In *British Machine Vision Conference* (2008).

[24] Enzweiler, M., Eigenstetter, A., Schiele, B., and Gavrila, D. M. Multi-cue pedestrian classification with partial occlusion handling. In *IEEE Conference on Computer Vision and Pattern Recognition* (2010).

[25] Ess, A., Leibe, B., and Gool, L. J. V. Depth and appearance for mobile scene analysis. In *IEEE International Conference on Computer Vision* (2007).

[26] Fergus, R., Perona, P., and Zisserman, A. Weakly Supervised Scale-Invariant Learning of Models for Visual Recognition. *International Journal of Computer Vision* (2006).

[27] Ferrari, V., Jurie, F., and Schmid, C. From Images to Shape Models for Object Detection. *International Journal of Computer Vision* (2009).

[28] Fischler, M., and Bolles, R. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography. *Communications ACM 24*, 6 (1981), 381–395.

[29] Freidman, J. H., Bentley, J. L., and R. A. Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software* (1977).

[30] Gavrila, D., and Philomin, V. Real-Time Object Detection for "Smart" Vehicles. In *IEEE International Conference on Computer Vision* (1999).

[31] Gavrila, D. M., and Munder, S. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision* (2007).

[32] Goedeme, T., Tuytelaars, T., and Van Gool, L. Fast Wide Baseline Matching for Visual Navigation. In *IEEE Conference on Computer Vision and Pattern Recognition* (2004).

[33] Grabner, H., and Bischof, H. On-line boosting and vision. In *IEEE Conference on Computer Vision and Pattern Recognition* (2006).

[34] Grabner, M., Grabner., H., and Bischof, H. Learning features for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition* (2007).

[35] Grabner, M., Grabner, H., and Bischof, H. Tracking Via Discriminative Online Learning of Local Features. In *IEEE Conference on Computer Vision and Pattern Recognition* (2007).

[36] Grabner, M., Leistner, C., and Bischof, H. Semi-Supervised On-Line Boosting for Robust Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[37] Haralick, R. M. Statistical and structural approaches to texture. In *Proceedings of IEEE* (1979).

[38] Harris, C., and Stephens, M. A Combined Corner and Edge Detector. In *Fourth Alvey Vision Conference, Manchester* (1988).

[39] Hinterstoisser, S., Benhimane, S., Lepetit, V., Fua, P., and Navab, N. Simultaneous Recognition and Homography Extraction of Local Patches With a Simple Linear Classifier. In *British Machine Vision Conference* (2008).

[40] Hinterstoisser, S., Benhimane, S., and Navab, N. N3M: Natural 3D Markers for Real-Time Object Detection and Pose Estimation. In *IEEE International Conference on Computer Vision* (2007).

[41] Hinterstoisser, S., Benhimane, S., Navab, N., Fua, P., and Lepetit, V. Online Learning of Patch Perspective Rectification for Efficient Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[42] Hinterstoisser, S., C.Cagniart, S.Holzer, S.Ilic, K.Konolige, N.Navab, and V.Lepetit. Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes. In *IEEE International Conference on Computer Vision* (Submitted 2011).

[43] Hinterstoisser, S., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. Gradient response maps for real-time detection of texture-less objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Submitted 2010).

[44] Hinterstoisser, S., Kutter, O., Navab, N., Fua, P., and Lepetit, V. Real-Time Learning of Accurate Patch Rectification. In *IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[45] Hinterstoisser, S., Lepetit, V., Benhimane, S., Fua, P., and Navab, N. Learning Real-Time Perspective Patch Rectification. *International Journal of Computer Vision* (2011).

[46] Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., and Navab, N. Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects. In *IEEE Conference on Computer Vision and Pattern Recognition* (2010).

[47] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Konolige, K., Bradski, G., and Navab, N. Anonymous Title. In *Submitted to ACCV* (2012).

[48] Holzer, S., Hinterstoisser, S., Ilic, S., and Navab, N. Distance Transform Templates for Object Detection and Pose Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[49] Huang, C., Ai, H., Li, Y., and Lao, S. Vector boosting for rotation invariant multi-view face detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (2005).

[50] Huttenlocher, D., Klanderman, G., and Rucklidge, W. Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1993).

[51] Jurie, F., and Dhome, M. A Simple and Efficient Template Matching Algorithm. In *IEEE International Conference on Computer Vision* (July 2001).

[52] Jurie, F., and Dhome, M. Hyperplane Approximation for Template Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 7 (2002), 996–100.

[53] Kadir, T., Zisserman, A., and Brady, M. An Affine Invariant Salient Region Detector. In *European Conference on Computer Vision* (2004).

[54] Klein, G., and Murray, D. Parallel tracking and mapping for small AR workspaces. In *IEEE International Symposium on Mixed and Augmented Reality* (Nara, Japan, November 2007).

[55] Koeser, K., Beder, C., and Koch, R. Conjugate Rotation: Parameterization and Estimation from an Affine Feature Correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[56] Lampert, C. H., Blaschko, M. B., and Hofmann, T. Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In *IEEE Conference on Computer Vision and Pattern Recognition* (June 2008).

[57] Lepetit, V., and Fua, P. Towards Recognizing Feature Points Using Classification Trees. Technical report, EPFL, 2004.

[58] Lepetit, V., and Fua, P. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. *Foundations and Trends in Computer Graphics and Vision 1*, 1 (October 2005), 1–89.

[59] Lepetit, V., Lagger, P., and Fua, P. Randomized Trees for Real-Time Keypoint Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (June 2005).

[60] Liebelt, J., and Schmid, C. Multi-view object class detection with a 3d geometric model. In *IEEE Conference on Computer Vision and Pattern Recognition* (2010).

[61] Lindeberg, T. Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales. *Journal of Applied Statistics 21*, 2 (1994), 224–270.

[62] Liu, M. Y., Tuzel, O., Veeraraghavan, A., Chellappa, R., Agrawal, A., and Okuda, H. Pose estimation in heavy clutter using a multi-flash camera. In *International Conference on Robotics and Automation* (2010).

[63] Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision 20*, 2 (2004), 91–110.

[64] Matas, J., Chum, O., Martin, U., and Pajdla, T. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference* (September 2002), pp. 384–393.

[65] Mikolajczyk, K., and Schmid, C. An Affine Invariant Interest Point Detector. In *European Conference on Computer Vision* (2002), pp. 128–142.

[66] Mikolajczyk, K., and Schmid, C. Scale and affine invariant interest point detectors. *International Journal of Computer Vision* (2004).

[67] Mikolajczyk, K., and Schmid, C. Performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2005).

[68] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. A Comparison of Affine Region Detectors. *International Journal of Computer Vision* (2005).

[69] Mohr, D., and Zachmann, G. Continuous edge gradient-based template matching for articulated objects. In *International Conference on Computer Vision Theory and Applications* (2009).

[70] Molla, E., and Lepetit, V. Augmented reality for board games. In *IEEE International Symposium on Mixed and Augmented Reality* (2010).

[71] Molton, N., Davison, A., and Reid, I. Locally Planar Patch Features for Real-Time Structure from Motion. In *British Machine Vision Conference* (2004).

[72] Muja, M., Rusu, R., Bradski, G., and Lowe, D. REIN - a Fast, Robust, Scalable REcognition INfrastructure. In *International Conference on Robotics and Automation* (2011).

[73] Navab, N., Benhimane, S., and Hinterstoisser, S. Computer vision cad models. Patent Pending, 2010. EP 08 83 7578.7.

[74] Neubert, J. J., and Drummond, T. Using Backlight Intensity for Device Identification. In *IEEE International Symposium on Mixed and Augmented Reality* (2006).

[75] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *IEEE International Symposium on Mixed and Augmented Reality* (2011).

[76] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. DTAM: Dense Tracking and Mapping in Real-Time. In *IEEE International Conference on Computer Vision* (2011).

[77] Nister, D., and Stewenius, H. Scalable Recognition With a Vocabulary Tree. In *IEEE Conference on Computer Vision and Pattern Recognition* (2006).

[78] Obdržálek, Š., and Matas, J. *Toward Category-Level Object Recognition*. J. Ponce, M. Herbert, C. Schmid, and A. Zisserman (Editors). Springer-Verlag, 2006.

[79] Olson, C. F., and Huttenlocher, D. P. Automatic Target Recognition by Matching Oriented Edge Pixels. *IEEE Transactions on Image Processing 6* (1997).

[80] Ozuysal, M., Calonder, M., Lepetit, V., and Fua, P. Fast Keypoint Online Learning and Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2009).

[81] Ozuysal, M., Fua, P., and Lepetit, V. Fast Keypoint Recognition in Ten Lines of Code. In *IEEE Conference on Computer Vision and Pattern Recognition* (June 2007).

[82] Ozuysal, M., Lepetit, V., Fleuret, F., and Fua, P. Feature Harvesting for Tracking-by-Detection. In *European Conference on Computer Vision* (2006).

[83] Ozuysal, M., Lepetit, V., and Fua, P. Pose Estimation for Category Specific Multiview Object Localization. In *IEEE Conference on Computer Vision and Pattern Recognition* (June 2009).

[84] Pan, Q., Reitmayr, G., and Drummond, T. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *British Machine Vision Conference* (2009).

[85] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. Object Retrieval With Large Vocabularies and Fast Spatial Matching. In *IEEE Conference on Computer Vision and Pattern Recognition* (2007).

[86] Richards, W., and Polit, A. Texture matching. In *Kybernetic* (1974).

[87] Rosten, E., and Drummond, T. Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision* (2006).

[88] Rothganger, F., Lazebnik, S., Schmid, C., and Ponce, J. Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints. *International Journal of Computer Vision 66*, 3 (2006), 231–259.

[89] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. Orb: An efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision* (2011).

[90] Rucklidge, W. Efficiently Locating Objects Using the Hausdorff Distance. *International Journal of Computer Vision* (1997).

[91] Salzmann, M., Pilet, J., Ilić, S., and Fua, P. Surface Deformation Models for Non-Rigid 3D Shape Recovery. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007).

[92] Serre, T., and Riesenhuber, M. Realistic Modeling of Simple and Complex Cell Tuning in the HMAX Model, and Implications for Invariant Object Recognition in Cortex. Tech. rep., MIT, 2004.

[93] Shaer, O., and Hornecker, E. Tangible User Interfaces: Past, Present, and Future Directions. *Foundation and Trends in Human-Computer Interaction* (2009).

[94] Sklansky, J. Image segmentation and feature extraction.

[95] Smith, S. M., and Brady, J. M. SUSAN – A New Approach to Low Level Image Processing. Tech. rep., Oxford University, 1995.

[96] Stark, M., Goesele, M., and Schiele, B. Back to the future: Learning shape models from 3d cad data. In *British Machine Vision Conference* (2010).

[97] Steger, C. Occlusion Clutter, and Illumination Invariant Object Recognition. In *International Archives of Photogrammetry and Remote Sensing* (2002).

[98] Stöttinger, J., Hanbury, A., Gevers, T., and Sebe, N. Lonely but attractive: Sparse color salient points for object retrieval and categorization. In *Workshop on Feature Detectors and Descriptors: The State of the Art and Beyond, in conjunction with IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[99] Sun, M., Bradski, G. R., Xu, B.-X., and Savarese, S. Depth-encoded hough voting for joint object detection and shape recovery. In *European Conference on Computer Vision* (2010).

[100] Tamura, H. S., and Ymawaki, Y. Textural features corresponding to visual perception.

[101] Taylor, S., and Drummond, T. Multiple Target Localisation at Over 100 FPS. In *British Machine Vision Conference* (2009).

[102] Taylor, S., Rosten, E., and Drummond, D. Robust Feature Matching in 2.3μs. In *Workshop on Feature Detectors and Descriptors in conjunction with IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[103] Tola, E., Lepetit, V., and Fua, P. DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2009).

[104] Tola, E., Strecha, C., and Fua, P. Efficient Large Scale Multi-View Stereo for Ultra High Resolution Image Sets. *Machine Vision and Applications* (Submitted 2010).

[105] Triggs, B. Detecting Keypoints with Stable Position, Orientation and Scale under Illumination Changes. In *European Conference on Computer Vision* (2004).

[106] Tuytelaars, T., and Van Gool, L. Wide Baseline Stereo Matching Based on Local, Affinely Invariant Regions. In *British Machine Vision Conference* (2000), pp. 412–422.

[107] Tuytelaars, T., and Van Gool, L. Matching Widely Separated Views Based on Affine Invariant Regions. *International Journal of Computer Vision 59*, 1 (2004), 61–85.

[108] Viola, P., and Jones, M. Robust Real-Time Object Detection. *International Journal of Computer Vision* (2001).

[109] Viola, P., and Jones, M. Fast Multi-view Face Detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (2003).

[110] Weise, T., Wismer, T., Leibe, B., and Gool, L. V. In-hand Scanning with Online Loop Closure. In *International Workshop on 3-D Digital Imaging and Modeling* (2009).

[111] Williams, B., Klein, G., and Reid, I. Real-Time SLAM Relocalisation. In *IEEE International Conference on Computer Vision* (2007).

[112] Wojek, C., Walk, S., and Schiele, B. Multi-cue onboard pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[113] Zucker, S. W., and Kant, K. Multiple-level representations for texture discrimination. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1981).