Augmented Reality for Non-Rigid Surfaces

THÈSE N^{O} 4192 (2008)

PRÉSENTÉE LE 31 OCTOBRE 2008

À LA FACULTE INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE VISION PAR ORDINATEUR

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Julien PILET

Ingénieur informaticien diplômé EPF
de nationalité suisse et originaire de Rossinière (VD)

acceptée sur proposition du jury:

Prof. D. Thalmann, président du jury
Prof. P. Fua, directeur de thèse
Dr A. Bartoli, rapporteur
Prof. O. Bimber, rapporteur
Prof. R.D. Hersch, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2008

Abstract

Augmented Reality (AR) is the process of integrating virtual elements in reality, often by mixing computer graphics into a live video stream of a real scene. It requires registration of the target object with respect to the cameras. To this end, some approaches rely on dedicated hardware, such as magnetic trackers or infra-red cameras, but they are too expensive and cumbersome to reach a large public. Others are based on specifically designed markers which usually look like bar-codes. However, they alter the look of objects to be augmented, thereby hindering their use in application for which visual design matters. Recent advances in Computer Vision have made it possible to track and detect objects by relying on natural features. However, no such method is commonly used in the AR community, because the maturity of available packages is not sufficient yet. As far as deformable surfaces are concerned, the choice is even more limited, mainly because initialization is so difficult.

Our main contribution is therefore a new AR framework that can properly augment deforming surfaces in real-time. Its target platform is a standard PC and a single webcam. It does not require any complex calibration procedure, making it perfectly suitable for novice end-users. To satisfy to the most demanding application designers, our framework does not require any scene engineering, renders virtual objects illuminated by real light, and let real elements occlude virtual ones. To meet this challenge, we developed several innovative techniques.

Our approach to real-time registration of a deforming surface is based on wide-baseline feature matching. However, traditional outlier elimination techniques such as RANSAC are unable to handle the non-rigid surface's large number of degrees of freedom. We therefore proposed a new robust estimation scheme that allows both 2–D and 3–D non-rigid surface registration.

Another issue of critical importance in AR to achieve realism is illumination handling, for which existing techniques often require setup procedures or devices such as reflective spheres. By contrast, our framework includes methods to estimate illumination for rendering purposes without sacrificing ease of use.

Finally, several existing approaches to handling occlusions in AR rely on multiple cameras or can only deal with occluding objects modeled beforehand. Our requires only one camera and models occluding objects at runtime.

We incorporated these components in a consistent and flexible framework. We used it to augment many different objects such as a deforming T-shirt or a sheet of paper, under challenging conditions, in real-time, and with correct handling of illumination and occlusions. We also used our non-rigid surface registration technique to measure the shape of deformed sails. We validated the ease of deployment of our framework by distributing a software package and letting an artist use it to create two AR applications.

Keywords: Augmented Reality, non-rigid surfaces registration, deformable object detection, geometric and photometric camera calibration, occlusion segmentation.

Réalité augmentée pour surfaces non rigides

Le terme réalité augmentée désigne l'intégration d'éléments virtuels dans la réalité. Dans la majorité des cas, elle consiste à insérer des images de synthèse dans une vidéo filmée en temps réel. La réalité augmentée implique le repérage spatial de l'objet à augmenter par rapport à la caméra. Des capteurs spécifiques, comme des suiveurs magnétiques ou des caméras infra-rouges, peuvent effectuer ce repérage. Cependant, ce matériel est coûteux et compliqué à utiliser. Il reste donc difficile d'accès pour le grand public. D'autres méthodes se basent sur des marqueurs d'apparence similaire à des code-barres. Elles présentent néanmoins un inconvénient majeur lorsque l'aspect visuel est important, car elles altèrent l'apparence des objets à augmenter. Dans ce contexte, des progrès récents en vision par ordinateur rendent possible le suivi et la détection d'objets en exploitant uniquement leurs caractéristiques propres. Cette technique reste encore peu répandue en réalité augmentée, à cause de la maturité encore insuffisante des logiciels disponibles. En outre, aucune de ces méthodes ne permet le traitement efficace de surfaces déformables, l'initialisation étant particulièrement problématique.

La contribution principale de nos travaux est donc un nouveau système de réalité augmentée gérant correctement, et en temps réel, des surfaces qui se déforment. Le matériel nécessaire se réduit à un ordinateur standard et à une webcam. De plus, notre système ne requiert pas de procédure de calibration compliquée, il convient donc parfaitement au grand public. Notre système peut aussi satisfaire les concepteurs d'applications les plus exigeants. En effet, il ne nécessite aucune modification de la scène à augmenter, affiche les objets virtuels éclairés par la lumière réelle et permet à des éléments réels de masquer les éléments virtuels.

Pour atteindre cet objectif ambitieux, nous avons développé plusieurs techniques innovantes :

- Notre approche de détection en temps réel d'une surface déformée se base sur la mise en correspondance de points entre une image modèle et l'image provenant de la caméra. La mise en correspondance est automatique, mais imparfaite, et exige donc l'élimination des données aberrantes. Mais les algorithmes traditionnels, comme RANSAC, sont incapables de gérer le nombre important de degrés de liberté d'une surface déformable. Nous proposons donc un nouvel algorithme d'estimation robuste qui permet le repérage spatial de surfaces déformables, à la fois en deux et en trois dimensions.
- Le réalisme en réalité augmentée exige également une bonne gestion de l'éclairage des objets virtuels par la lumière réelle. Les techniques existantes ont souvent besoin de procédures compliquées de mise en place, ou d'outils comme des sphères réfléchissantes. Notre système intègre également des méthodes d'estimation de l'éclairage, mais sans sacrifier sa facilité d'utilisation. La réalité augmentée résultante est donc à la fois accessible au grand

public et réaliste.
- Notre approche permet de masquer partiellement l'objet virtuel lorsqu'un élément réel le cache. Les approches habituelles de gestion d'occultations se basent sur plusieurs caméras ou ne peuvent gérer que des occultations causées par des objets modélisés à l'avance. La nôtre ne nécessite par contre qu'une seule caméra et modélise en temps réel les objets occultants.

Nous avons intégré ces trois composants dans un système cohérent et flexible. Nous l'avons utilisé pour augmenter des objets variés, comme un T-shirt ou une feuille de papier. Notre système augmente des surfaces pendant qu'elles se déforment, dans des conditions difficiles, en temps réel et avec une gestion correcte de l'éclairage et des occultations. Nous avons également utilisé notre technique de detection de surfaces non rigides pour mesurer la forme de voiles déformées. Finalement, nous avons validé la facilité de déploiement de notre système en distribuant un logiciel. Il a été utilisé entre autres par une artiste pour créer deux applications de réalité augmentée.

Mots-clés : Réalité augmentée, repérage spatial de surfaces non rigides, détection d'objets déformables, calibration géométrique et photométrique de caméra, segmentation d'occultations.

Acknowledgments

The job of EPFL professor is far from easy. A professor is supposed to be a good researcher, a good teacher, a good chief, a good advisor and a good writer. Although human and therefore not perfect, Prof. Pascal Fua has all these skills and directs the Computer Vision Laboratory with talent. I am grateful for his highly valuable advice and for the liberty he gave me.

I express my gratitude to the thesis committee members: Prof Oliver Bimber, Dr Adrien Bartoli, Prof. Roger D. Hersch, and Prof. Thalman for accepting to evaluate this work. I particularly appreciated Adrien's thorough proofreading and precious comments.

How can I thank Vincent Lepetit? His intelligence and his creativity were always so helpful and so inspiring to me. I based almost all my work on his. In short, Vincent is my idol.

I am grateful to Christoph Strecha who helped me to clean and formalize my latest developments.

I have to thank Pascal Vuilliomenet for his efficient and friendly coordination of the collaborations with both Alinghi and Hydroptère teams.

Many thanks to Mustafa for the nice and friendly office sharing, for the many inspiring discussions, for supporting my mess, and for helping me digging Luca's scratch paper stack.

Josiane deserves special thanks for her efficiency, for her patience, and her support.

I thank all the lab members that made the atmosphere pleasant and motivating. Collaboration with my colleagues has always been a great pleasure.

I thank Irène, Windekind, Luxemarie, Blue Bird, and their respective crews for the fun I had sailing with them. I am also grateful to all my loyal friends. Their friendship is a lighthouse allowing me to find my bearing in my life.

I thank all my family, especially Patrick and Martine for giving me the chance to live during more than three years in a huge and amazingly comfortable apartment.

Many thanks to Christiane and Marjo for their love, for their generosity, and for their unconditional support during my long studies.

At last, my greatest gratitude goes to my best friend, my lover, my partner, my wife: Nelly. The fantastic love we share makes my life happy and joyful. I'm especially grateful for her exceptional patience, tolerance, and support before paper deadlines and during the writing of this thesis.

Contents

Notations

Scalar i, r

Point Individual 3–D or 2D points are noted with Roman characters: p, v

Correspondence A single correspondence between 2–D points c_0 and c_1 is written: $c = \{\mathrm{c}_0, \mathrm{c}_1\}$

Set of correspondences capital: $C = \{c_1, ..., c_{|C|}\}$

Vector Bold: $\mathbf{v}$

Matrix Roman capital: M

θ Parameter vector

v_i Vertex number i

ε Energy function

$\mathrm{q} = T_\theta(\mathrm{p})$ The surface to screen transformation $T_\theta : \Re^2 \rightarrow \Re^2$ sends the point p lying on the original surface to screen location q. The transformation is parameterized by θ.

$b_i(\mathrm{p})$ Barycentric coordinates of point p on the undeformed model.

r Radius of confidence

$\rho(\delta, r)$ Robust estimator

$\mathbf{m}$ The model image, also called reference or template image. The pixel at location p is written $\mathbf{m}_\mathrm{p}$.

$\mathbf{u}$ The input image.

$\mathbf{s}$ The augmented image showing synthetic geometry or surface albedo.

e_p The irradiance reaching surface point p.

List of Abbreviations

AR	Augmented Reality
CG	Computer Graphics
CVPR	Computer Vision and Pattern Recognition
DoF	Degrees of Freedom
E-M	Expectation-Maximization
FPS	Frame Per Second
GLSL	OpenGL Shading Language
GMM	Gaussian Mixture Model
GPL	GNU General Public License
GPU	Graphic Processing Unit
ICCV	International Conference on Computer Vision
ISMAR	International Symposium on Mixed and Augmented Reality
PC	Personal Computer
PDA	Personal Digital Assistant
pdf	Probability Density Function
RANSAC	Random Sample Consensus
ROC	Receiver Operating Characteristic
SLAM	Simultaneous Localization and Mapping
VR	Virtual Reality

1. Introduction

The popularity of Augmented Reality (AR) has been growing during the last two decades, helped along by the increasing power of computer graphics hardware, the decreasing prices of webcams and camcorders, and, last but not least, the increasing robustness of computer vision algorithms. This is particularly true of recent approaches to rigid object detection and tracking for which many effective and real-time solutions have been proposed [69, 63, 88, 61]. These methods have reached the maturity necessary for augmented reality applications that require automation, robustness, accuracy, and speed. By contrast, sound registration of non-rigid surfaces still lags behind, which limits the range of objects that can be handled. Non-rigid tracking solutions exist [5, 40, 24, 21, 3], but they tend to be slow and automated initialization remains an issue for which there is no convincing real-time solution.

This is the problem we address in this thesis. We propose a framework for augmenting images of a non-rigid surface filmed with a standard camera. Because geometric registration is only one of the steps required by augmented reality, our framework includes vision-based tools not only for geometric registration of deformable surfaces, but also for recovering illumination, allowing real light to affect virtual objects, and for handling occlusion to allow real objects to hide virtual ones.

The resulting framework lets us augment fast and realistically images of non-rigid surfaces acquired by an ordinary camera. We took a particular care to ensure its genericity and ease of deployment. It was successfully used by augmented reality application designers, such as artists, allowing them to concentrate on application design, rather than dealing with technical issues [95]. Furthermore, the algorithms we developed are applicable in a wider context that includes 3–D surface measurement and video surveillance.

1.1. Contributions

Prior to our work, no AR system could properly augment a deformable surface seen by a single ordinary camera. Our key contribution is therefore to make AR possible and realistic in such a challenging context: we augment in real-time images of deforming surfaces using a camera as the sole sensor. We achieve this ambitious goal by developing several new algorithms, which, together, form a consistent and flexible AR framework. Its main components are geometric registration, both in two and three dimensions, illumination handling, and occlusion handling. Compliance to a common set of constraints ensures good interoperability between the components. They can all handle both rigid and non-rigid surfaces. They do not require any engineering of the scene. They run in real-time. They require a single camera but can take advantage of additional ones, if available. Finally, they deliver high visual quality without sacrificing ease of use, both for end-users and for application designers.

Achieving this has required a number of technical advances because state-of-the-art techniques were not up to the challenge:

Registration We propose a method to fast and robust non-rigid surface detection. To run in real-time, our method relies on wide-baseline matching of 2–D feature points. Given such correspondences, if the target object were rigid, detecting it and estimating its pose could be implemented using a robust estimator such as RANSAC [34]. However, for a deformable object, the problem becomes far more complex because not only pose but also a large number of deformation parameters must be estimated. We therefore contribute a robust optimization scheme designed to work in these conditions, resulting in an accurate, fast, and robust non-rigid surface registration algorithm.

Illumination We propose new approaches to allowing virtual objects to reflect real light. For full 3–D virtual objects, we construct a radiance map. For a virtual 2–D layer lying on the real surface, we locally estimate illumination. The novelty of these approaches comes from the fact that neither requires specific devices or complex calibration steps, as most state-of-the-art approaches do. Therefore, our methods allow novice end-users to enjoy virtual objects shaded by real illumination.

Occlusion We push visual quality for AR even further by allowing real objects to hide virtual ones. Our contribution consists in a new solution to occlusion segmentation that does not require multiple views or any other device, and that can handle unmodelled occluding objects.

Combining these methods results in a framework able to augment non-rigid surfaces, reflecting real illumination on virtual elements, and respecting the occlusions caused by real objects.

1.2. Organization of this Document

The thesis is organized as follows. The next chapter reviews AR basics, investigates its requirements, and sketches a generic framework. Chapter 3 discusses the importance of augmenting non-rigid surfaces and defines assumptions on top of which we designed solutions that fit into our framework. Chapter 4 introduces our approach to geometric registration and calibration, notably our robust estimation scheme. Chapter 5 covers illumination issues. Chapter 6 shows how real objects can be made to occlude virtual ones. Chapter 7 presents the results obtained with our framework. Chapter 8 discusses the impact and limitations of our work and concludes the thesis.

As opposed to a traditional habit, we review related work at the beginning of each chapter, rather than in a separate one. Thus, a broad overview of state-of-the art methods for AR is presented in Section 2.3, while more Computer Vision related methods for registration are reviewed in Section 4.1, for handling illumination in Section 5.1, and for handling occlusions in Section 6.1.

2. About Augmented Reality

Virtual Reality has become popular and widespread during the 20 last years, thanks in particular to the rapid improvement of computer graphics. It is now mature enough to immerse users in virtual environments and allows video-game players to evolve in complex and realistic but entirely artificial worlds. By contrast, AR combines real and virtual by adding synthetic elements to the real world, or by altering or hiding real ones. More specifically, in video-based AR, a camera captures a scene, a computer then analyzes the images and renders the virtual objects at the correct location and under the appropriate lighting conditions. The resulting image is shown to the user who can then interact with both real and virtual parts of the scene.

2.1. Applications

We begin by presenting the most representative applications of this technology. Some of them have already been demonstrated and used. We believe that the others soon will be.

Home Entertainment A recent example illustrates the interest of this business for augmented reality: The video game *The Eye of Judgment* published in 2007 by Sony Computer Entertainment Inc for the PlayStation 3 console (Figure 2.1). In this game, playing cards are augmented with virtual creatures. The software relies on black and white patterns printed on cards border to detect them and compute their orientation. Using such patterns is acceptable in this context because the target audience is fond of emerging technology.

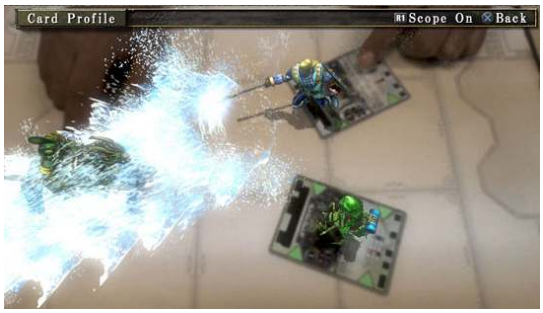


Figure 2.1.: *The Eye of Judgment,* a Sony game that mixes virtual creatures and cards using augmented reality. A camera films the game board on which the user disposes cards. On the TV screen, virtual characters come to life and stand on the real cards.

Figure 2.2.: Mobile phones and other embedded systems can also augment reality. Courtesy of Daniel Wagner [115].

Mobile Applications As depicted by Figure 2.2, the recent increase in performance of mobile devices, such as PDAs or mobile phones, allows them to support AR applications [114]. AR reaching such common and ubiquitous devices will lead to numerous applications. Imagine a mobile device acting as an electronic guide for tourists, that could augment street signs with their translations, or monuments with visual annotations. The tourist could also use a conventional paper map showing static information at a very high resolution, and see an augmented object by pointing the device towards the place of interest. In other words, AR is a powerful way to supplement a paper map with dynamic and interactive content.

Augmented reality on a mobile platform also offers advertisement opportunities. By pointing the device towards an advertisement displayed in a street, a user could see a live augmentation presenting further information. This would turn a simple static display into an animated and interactive one.

Tourism and Culture In theme parks and in museums, visitors can be made to experience virtual objects or characters in a real environment. Augmenting antique ruins with a virtual reconstruction of the original building can help visitors to grasp the magnificence of past cities, without altering the ruins [113]. Virtual characters dressed in historical costumes playing in an ancient residence can also greatly improve visitor's experience, as demonstrated in the Norwegian Museum of Cultural History. A method to superimpose pictorial artwork with projected imagery is proposed in [10]. It allows overlaid graphics and animations to tell stories about a painting.

The French company Total Immersion designed a show called The Future is Wild for the theme park Futuroscope, in Poitiers, France. It allows visitors to experience a virtual safari, set in the world as it might be 200 millions years from now. The animals of the future are superimposed on reality. They come to life in their surrounding and react to visitors gestures.

Augmented reality has a strong potential to impress spectators. Therefore, it can be integrated into company presentations, university lectures, or other live communication events. For example, a narrator can present a real object which is then augmented live in the background. It is precisely what Intel did in January 2008 at the Consumer Electronic Show, Las Vegas, to unveil a range of new processors, including chips designed for so-called "mobile Internet devices". The impact of the demonstration was guaranteed because the technology did not require any marker

Figure 2.3.: The depicted AR system is commercially available and allows training firemen in realistic yet safe situation. Users see in their head mounted display harmless and virtual fire and smoke. Courtesy of Resolve Fire & Hazard Response Inc.

or visible elements beside a PDA equipped with a camera.

For such applications, an unintrusive and realistic augmented reality is best. The better real visual effects such as illumination and occlusion are reproduced on virtual objects, the more natural will be user's feeling.

TV and Movie Industry Adding virtual elements in TV shows is very popular. For instance, the company Orad proposes solutions for augmenting sports events with advertisement or additional information, such as distance measures on a soccer field.

Augmented reality is a convenient way of providing real-time special effects preview for film making. Thus, directors can quickly have a clear idea of the final result, without having to wait that the computer graphics team finishes its work.

Training, Visualization and Maintenance Training firefighters, policemen, or first-aid workers to handle emergency situations can also benefit from AR. As depicted by Figure 2.3, the trainees can be shown catastrophic but virtual events superimposed on the real environment, such as a virtual fire in a real building, or a virtual flood in a real city.

Augmented reality can also be used to visualize inaccessible objects. For example, city workers could see augmented pipes in the streets before starting to dig a hole. In another context, augmenting a patient's skin with virtual organs and tumor could help a surgeon in his work, as depicted by Figure 2.4.

Augmented reality can also be used to give maintenance instruction. The instructions to repair a complex device such as a printer or a car engine can be provided through AR, as depicted by Figure 2.5. The direct overlay of instructions over the part of interest can be much more efficient than reading documentation on paper.

This AR application list is of course not exhaustive, and many more are still to be invented. Because many fields can benefit from AR and because young and immature technology already

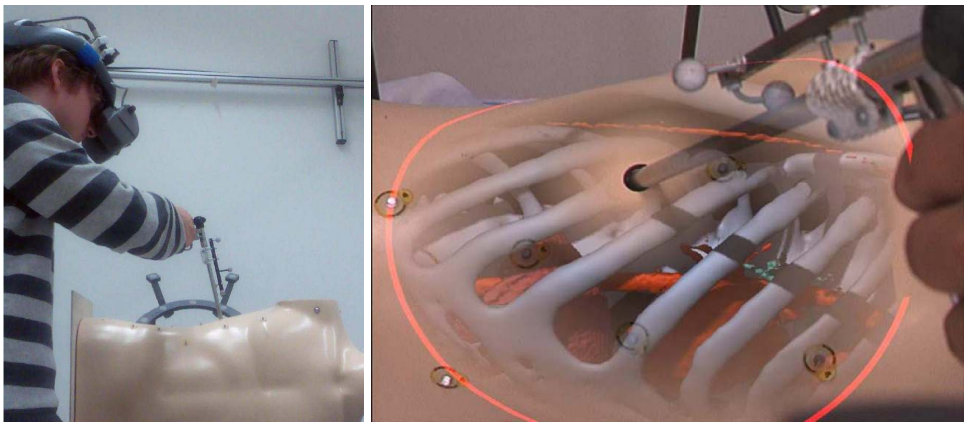


Figure 2.4.: An application of AR to surgery. This prototype proposes to use AR to display 3–D data during endoscopic surgery. Courtesy of Christoph Bichlmeier [9].

Figure 2.5.: An application using AR to show maintenance instructions directly on the broken device. Courtesy of Gerhard Reitmayr [87].

meets success, we believe that AR techniques will improve in the next years, allowing larger diffusion and popularity.

2.2. Challenges

In spite of all these potential applications, AR is not yet as ubiquitous as Virtual Reality, mostly because the technology is far less mature and AR research, unlike VR research, still focuses more on developing basic techniques rather than their applications. This is because integrating virtual elements in a real picture is far more complex than synthesizing an image from scratch: The real world is uncontrolled and has to be sensed and "understood", while virtual worlds are simply built to suit the application's requirements. This requirement for automated machine understanding of reality is one of the great challenges of AR and is worth explaining in more details.

A typical video-based AR system repeatedly performs the following steps: Image acquisition, scene modeling, virtual object rendering, and display. While the loop is running, the user can interact with real objects and see results on both real and virtual elements. As we will see, additional sensors can be used to make the job easier, but the camera remains the most important one.

Given these sensory inputs, the AR system must perform three critical tasks:

Registration To properly draw a virtual object in a picture, it is necessary to know where to draw it, at what size, and under which perspective. Therefore, an AR system must constantly model the position of the scene with respect to the camera. This is known as registration and must be accurate because little errors quickly produce an unrealistic effect.

Illumination Modeling Rendering a virtual object in a real scene will lack realism if it does not reflect the real illumination environment. Therefore, a practical AR solution must be able to model real illumination conditions.

Occlusion Handling If something real lies between the camera and the virtual object, the latter should only appear partially in the augmented image. To this end, the system either needs knowledge of the 3–D scene, which is costly to provide, or a way to decide at runtime what parts of the virtual object are occluded.

Registration, illumination modeling, and occlusion handling have to be performed on-line, since camera motion, illumination and object visibility can change non deterministically. They also have to be obtained in real time and with delays as short as possible to permit comfortable interaction. In most cases, if the user had to wait even half a second for the system to react, he would probably quickly loose interest in using it.

Figure 2.6.: An AR application based on ARTag [32] that register bar code-like markers. The user wears a special costume which is augmented with a virtual armor.

2.3. Related Work

Having identified three challenges that video-based AR systems must meet, we briefly review current techniques dedicated to solving them.

Registration An abundant literature is available about registration. Our goal here is not to give a complete review but rather to describe classes of solutions.

To measure the pose of a moving object or camera, one can attach to it a dedicated sensor to measure pose and orientation. Among such sensors that have successfully been used for AR purposes are magnetic sensors, inertial measurement units that combine accelerometers and gyroscopes, and retro-reflective markers filmed using infra-red cameras. They offer robustness. However, they are expensive, cumbersome to setup, and they might limit the volume in which tracking is possible. Moreover, their accuracy is not always sufficient for AR, and video-based techniques have to complement them.

Computer vision based approaches directly analyze the image content to infer the camera pose and rotation with respect to an object visible in the image. An option is therefore to introduce in the scene one or more object that are specifically designed for automated detection. Many augmented reality systems rely on such markers that often look like bar-codes. Registration of natural and known objects is also possible, often by exploiting their 3–D model. Some approaches use the edges of a CAD model [57]. Others are based on the texture naturally present on the object [63].

Some tracking algorithms do not even need to know what they are looking at; They simultaneously track the camera movements and reconstruct a 3–D representation of the scene [58].

These approaches solve a problem often called Simultaneous Localisation and Mapping and can only deal with rigid environments.

Illumination Inlaying a virtual object into a real picture requires some knowledge of the real illumination. In the pioneering work of [78], geographic position, date and time of picture acquisition was used to determine sun's orientation, allowing for realistic shading of outdoor scenes. Some approaches estimate the environment beforehand, requiring a full 3–D model of the visible scene, which usually implies some manual intervention [27, 68, 41]. Others rather make use of specific instruments. Debevec takes several pictures of a spherical mirror under different exposures [23]; Sato et al use a pair of images taken with stereo wide-angle cameras [93]. Bimber et al use projectors and cameras to capture reflectance information from diffuse real objects and to illuminate them under new synthetic lighting conditions [11].

On-line estimation of dynamic illumination environment is also possible, for example by combining a spherical mirror on a marker. Image analysis of such a device provides both the pose of the virtual object and its light environment [55].

Occlusion One possible way AR systems have to let a real object occlude a virtual one is to use a depth sensor attached to the camera. The depth information is then used as a Z-buffer: Pixels showing a real object closer than the virtual one are left unmodified. Depth sensors include stereo pairs [118, 42], trifocal cameras, and laser range scanners.

Other methods assume that potentially occluding objects are known and modeled in 3–D. Their registration allows the system to compute a Z-buffer, which solves the problem. However, this presumes that the environment is static and has been completely modeled beforehand.

Finally, some approaches do not rely on computing depth buffer. Instead, they use reasoning on contours [8] or semi-interactive outlining of occluding objects [60].

2.4. A Generic Framework for Augmented Reality

We have seen that a variety of approaches is possible to address AR issues, each with strengths and weaknesses. When designing a specific AR application, the developer has several options to choose from. This choice should be conditioned by the answers to the following questions:

Devices What are the available devices? Is a single, uninstrumented camera the only available device for registration? Or is it possible to use multiple ones, and dedicated sensors?

Scene engineering Is it permissible to engineer or instrument the scene to augment, for example by painting a marker on the augmented object?

Visual quality How important is it to account accurately for illumination effects and occlusions?

Ease of use How knowledgeable are the end users expected to be?

	Single camera	No scene engineering	Visual Quality	Ease of use
Home applications	++	+	+	++
Museums	+	++	+	++
TV/Movie		+	++	+
Mobile	++	+	+	++
Shows		++	+	
Training, Visualization		+		++

Table 2.1.: The requirements of augmented reality applications. A "++" in a cell denotes that the corresponding criterion is necessary for the application. A "+" denotes an optional criterion: The application would benefit from satisfying the constraint, but it is possible without it. An empty cell denotes an irrelevant criterion. Obviously, real-time registration is a requirement shared by all augmented reality applications.

In this work, we did not attempt to design a single AR application. Instead, we created generic building blocks that could serve in any AR context. Our goal is to allow AR application designers to concentrate on authoring rather than on solving technical issues by letting them use generic off-the-shelf solutions that are as widely applicable as possible. With this goal in mind, we answered the questions introduced above in the following manner:

- We want to develop methods that work without specialized registration devices. Only a single standard camera is required. If more cameras are available, they can improve accuracy, but that must remain optional.
- We want to handle natural objects without having to instrument the scene.
- To achieve realism, the visual quality should be high and we should model the effect of real light on virtual objects and their occlusion by real objects.
- The end users should not have to be experts and should not have to follow complex calibration procedures or to perform manual tracker initialization.

On top of these requirements remains of course the time constraint. AR application should be interactive, completely automated, and real-time.

An AR system that could do all this reliably would certainly be generic. Table 2.1 highlights this by listing the requirements of different application classes and by rating the importance of each one of these constraints. It states for example that a single camera is the only acceptable device for home and mobile applications, for practical and economic reasons. A discreet AR technology that does not require visible markers is useful for all kind of applications, but specially for museums and for shows. In a museum, the original objects should not be altered. For best impact during a show, AR technology should remain invisible to give more visibility to the result. Visual quality is similarly important everywhere, except for technical visualization which prefers clarity to realism. Ease of use is important in all application involving untrained users.

Table 2.1 summarizes the requirements and confirms that our target AR system would suit many application classes. In this work, we therefore tried to keep these constraints in mind when developing algorithms.

From the AR application author point of view, such genericity can have a cost: An overconstrained system might not reach the performances of an ad hoc solution. However, we will show that this performance loss can be minimized by using well-designed algorithms. Furthermore, the ease of deploying a generic system as opposed to developing a specific solution usually outweighs any drawbacks, since it allows designers to spend time on authoring the application, rather than solving its technical issues.

3. Augmenting Deformable Surfaces

By contrast to AR involving rigid objects, we are not aware of any interactive AR application involving deformable surfaces that predate our publication in 2005 [84]. Yet, it is an important topic, as evidenced by the many papers published on this topic since then [116, 66, 28, 47, 40, 123, 50].

In this chapter, we start by distinguishing 2–D from 3–D approaches to deformable surface augmentation and discuss their respective strengths and weaknesses. We then review the difficulties that must be overcome to achieve real-time AR for non-rigid surfaces and then list its most important applications. Finally, we define the goals and assumptions we base our work on.

3.1. The Third Dimension

Augmented reality can be achieved either in two or in three dimensions. In 2–D, the goal is to modify the color and texture of real objects without changing their geometry. In 3–D, the augmented objects do not have to lie on existing ones: Both geometry and texture can be virtual. Figures 3.1 and 3.2 depict 2–D and 3–D augmented reality.

Choosing two or three dimensions has strong implications while designing an augmented reality system, most obviously for registration. Recovering a 2–D geometric transformation from an image is far less ambiguous than a 3–D one. In pathological cases, several very different 3–D configurations can lead to very similar image projections, making recovery by minimization subject to many local minima. Moreover, a 2–D geometric registration does not require camera calibration, because it handles pose and projection together, as a single problem. In 3–D both are dissociated, and recovering pose and shape requires the intrinsic parameters of the camera,

Figure 3.1.: Augmenting a deforming sheet of paper with a 2–D cartoon character. Left: Input image. Right: Augmented image. The virtual drawing bends with the paper, is shaded appropriately and is partially occluded by the hand holding the page.

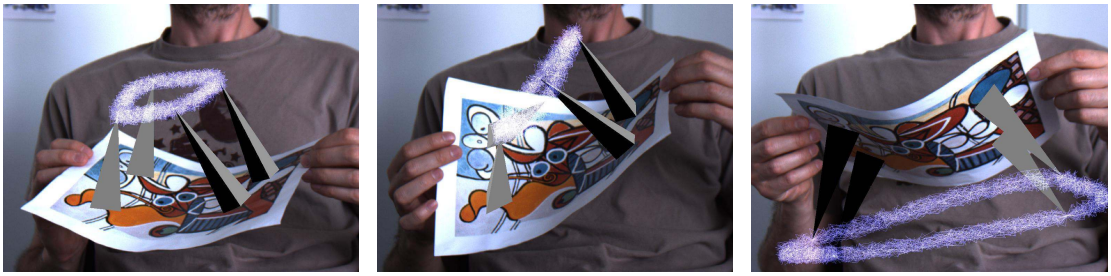


Figure 3.2.: Augmenting a deforming sheet of paper with 3–D objects. In this application, the user can change the path of the electric arc connecting the virtual poles by deforming the sheet of paper.

adding complexity. However, working in object space rather than in image space also has advantages. 3–D surface deformations have a true physical meaning, unlike their 2–D projection. This makes handling self-occlusions much easier. A 3–D model will be able to correctly predict and locate hidden areas just by observing visible ones. By contrast, a 2–D model will often let the hidden part float.

Illumination handling is also easier in 2–D, because illumination information is present in each frame. The real surface required for retexturing naturally reflects real light. Thus, when augmenting a single pixel showing a particular surface area, illumination can be deduced from the original pixel itself. In 3–D, by contrast, virtual objects do not have a real counterpart and therefore do not reflect real light. In this case, an estimate of the illumination environment is necessary to illuminate virtual objects appropriately.

Handling occlusions is also more complex in 3–D than in 2–D. When a real object occludes a virtual one that lies on a real surface, observing whether the surface is visible or not is sufficient. In 3–D, there is no real surface to observe. Virtual and real objects can intersect. Thus, a depth estimation is required for proper occlusion handling.

Because the choice of working in two or three dimensions depends on the application, we present methods for both. In cases for which modifying the texture of real objects is enough, our 2–D algorithms provide efficient registration, illumination, and occlusion to virtual elements. In cases of augmentation with a 3–D virtual object whose geometry does not match any real one, our set of 3–D algorithms can be used.

3.2. Required Developments

To understand the implications of augmenting deformable surfaces, we first review the difficulties to overcome and then see what are applications of their solutions.

Registration To the best of our knowledge, no algorithm published before 2005 could properly detect and register non-rigid surfaces. Some could *track* them, given the position in the previous frame, but all require manual intervention or pre-defined starting location for initial-

ization [19, 4, 3, 17, 26, 43]. A few could *detect* a non-rigid surface and register it without an initial estimate, but not accurately and efficiently enough for augmented reality purposes [5, 31]. Fast and automated initialization is crucial for AR, since it is key to fluent interaction. This unresolved initialization issue might seem surprising, given that registration of rigid or articulated objects has been extensively studied. We believe this is because non-rigid surfaces raise several difficult issues, the most important of which is the large number of degrees of freedom required to represent their deformations. Furthermore, recovering the configuration of a known, deformable surface from a single view is ambiguous because different shapes might produce similar image projections.

What can we rely on to address these issues and achieve geometric registration fast enough for interactive applications? Modeling possible shape deformations with physical surface properties or statistical learning greatly help reducing the number of degrees of freedom. The original snakes and their extensions to surfaces did this by introducing quadratic regularization terms [56, 38]. An alternative is to use dimensionality reduction techniques to learn shape models from training databases [20]. In this work, we used both methods. Quadratic regularization terms are sufficient for 2–D non-rigid registration. Training from automatically generated databases allows handling of the third dimension.

We also relied on some Computer Vision advances, such as fast wide-baseline feature matching techniques that have been successfully used for rigid object detection [63]. The process of establishing correspondences does not change much when going from rigid objects to deformable ones. However, a novel approach was required to actually register the surface based on these correspondences, because traditional robust pose estimation algorithms used in the rigid case do not trivially extend to the non-rigid one. In Chapter 4, we introduce this new approach to robust estimation that can handle the added degrees of freedom.

Illumination Recovering illumination for augmented reality is not an easy task. In the case of a deforming object, the problem is made even more complex by the fact that surface orientation changes inhomogeneously. The object may also cast complex shadows on itself. Moreover, we operate in a framework that forbids dedicated hardware for illumination estimation, which eliminates many existing techniques.

Because we use a single camera as sensor, we do not attempt to build a full illumination model. Instead, we simply observe illumination effects and handle them as they are. This is particularly appropriate for retexturing, in which augmentation is achieved by only modifying the texture of real objects. In that case, the geometry of real and virtual scenes are identical and illumination at a particular surface point can be recovered by examining the corresponding pixel. Illumination can then be reproduced on the augmentation to realistically change the object color. This simple approach, presented in Chapter 5, is well adapted to monocular augmentation of non-rigid surfaces, because it can handle complex effects.

Occlusion Because we want a system that can work using a single camera, obtaining depth data for occluding virtual objects is difficult. However, given a known surface that we can register, rendering it produces an image comparable to the camera output. By comparing these two images, we can segment occluded areas and occlude virtual objects appropriately. In Chapter 6,

Figure 3.3.: Our non-rigid augmented reality technology can be used to augment the same logo printed on different surfaces. Here, the ICCV 2003 logo is detected as well on a T-Shirt as on a mug.

we introduce the corresponding segmentation algorithm.

3.3. Applications

In the previous chapter, we have introduced a number of existing AR applications. We now discuss a range of new ones that our ability to handle deformable surfaces will make possible.

Medical Applications Medical imaging techniques such as Computer Tomography or Magnetic Resonance Imaging have changed the way surgery is planned and executed. They reveal, without intrusion, the inside of a patient, resulting in large amounts of 3–D data whose visualization is an important issue. Due to data size and complexity, surgeons can only observe the images during the operation preparation phase. During surgery, the practitioner has to mentally link the images with the patient body, as a traveller recognizes landmarks to locate himself on a map. This process might lack accuracy and, in case of endoscopic surgery, is made even more difficult because the only available view is limited. As already explored more than ten years ago [2, 44] and more recently [35], AR can play an important role in this domain by showing the surgeon information automatically registered with the patient body. He could for instance virtually see organs through the patient skin, thereby helping him to drive his endoscope towards the target location.

Visualization of medical data with AR has the advantages that registration can be automated and that dynamic data can be displayed live. However, registration is complex because a body is not rigid. Even under complete narcosis, breath causes organs to move within the body. Therefore, proper augmentation requires non-rigid skin tracking, from which the location of organs can be deduced. In [79], an experiment conducted in real conditions showed that an average precision of 9.5mm can be reached with a rigid model. Using non-rigid registration techniques such as ours could clearly improve accuracy of registration, thereby making AR practical for medical operations implying a deforming body.

Augmenting the human body has more medical applications, many of them are only practical if deformations are properly handled.

Advertisement Live insertion of advertisement during sport event broadcasts represents an important market. Companies such as Orad proposes commercial solutions to this end. However, their system is limited to planar area of uniform color. Our approach to non-rigid AR could extend target surfaces to player clothes. For instance, T-shirts of football players could be augmented with dynamic advertisement.

AR also offers advertisement opportunities through mobile phones. Nowadays, almost every mobile phone is equipped with a camera. Users could point their handy to an advertisement board showing a particular logo. After detection and registration, the device could show virtual elements enhancing the original advertisement. Detection in this case is made difficult by the fact that the logo could be printed on a planar board, a cylindrical one or other merchandising goods such as a T-shirt or a mug. However, our non-rigid AR framework can treat the logo as deformable and detect it wherever printed. Figure 3.3 depicts how our system detected and registered a logo printed both on a mug and on a T-shirt.

Entertainment Video games such as *The Eye of Judgment* demonstrated the entertainment capability and commercial viability of traditional AR. Augmenting non-rigid surfaces such as clothes or deforming paper is then a natural extension. During webcam-based communication, users could wear virtual elements such as jewels, company logo, or animated cartoons. Video games could also animate pictures on paper, mimicking the moving pictures of *the Daily Prophet* newspaper, in the Harry Potter movies. The possibility of augmenting non-rigid surfaces opens many new perspectives.

Deformation Measurement Our non-rigid detection algorithm has potential applications beside augmented reality. It can serve as a measurement system to compute the shape of a non-instrumented, known, deformable surface, simply from pictures. An example of such a use is at the origin of this work: Measuring the shape of a spinnaker during navigation. This is challenging, because the spinnaker is about 500 square meters. It keeps deforming. It is partially elastic. It is not possible to instrument it, because instrumentation weight would bias the measures. Lasers have difficulties to handle sun light and the sail's semi-transparency. It is also difficult to find a viewing angle to shoot the spinnaker entirely on the same image, due to its semi-circular shape. However, thanks to our non-rigid detection algorithm, measuring such a sail is now possible for Alinghi, the Swiss team for the America's cup. As depicted by Figure 3.4, an off-line study of pictures can bring accurate but sparse measures for improving sail design. On-board, real-time measurements can help sailors to optimize trimmings. Our method is adapted to both cases.

Sails are not the only surfaces whose deformation can be measured. A car during a crash test can also be considered as a deformable surface. An airplane's wings also deform during flight. Automatic observation of watch straps during wear tests is another potential application. Our vision-based approach is a powerful deformation measurement tool. It has the important advantage not to require any instrumentation of the observed surface.

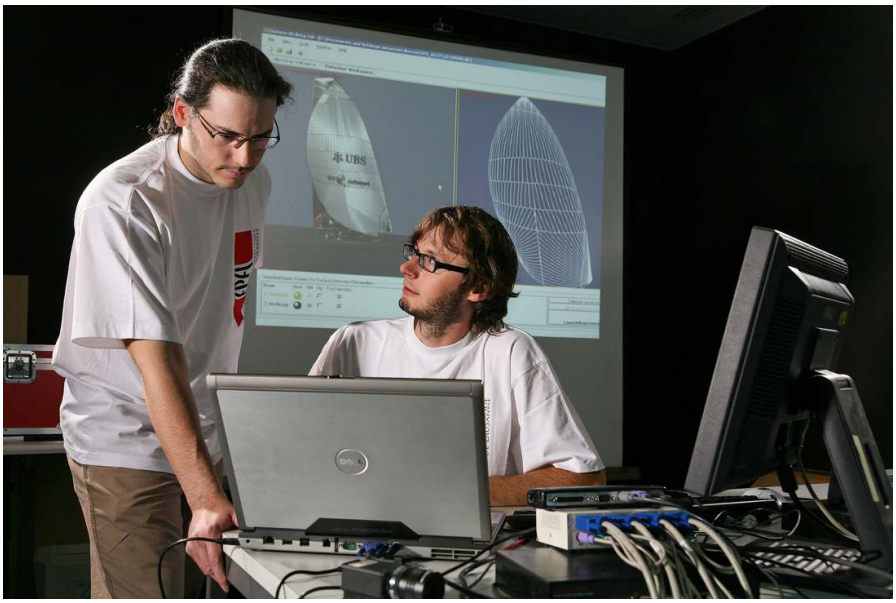


Figure 3.4.: The author and his colleague André Mazzoni measuring the shape of a Spinnaker. Photo: Alain Herzog.

As we have seen, several fields can directly benefit from augmenting non-rigid surfaces. Since we now have a rough idea of how it extends the AR perspectives, we can define our goals more precisely.

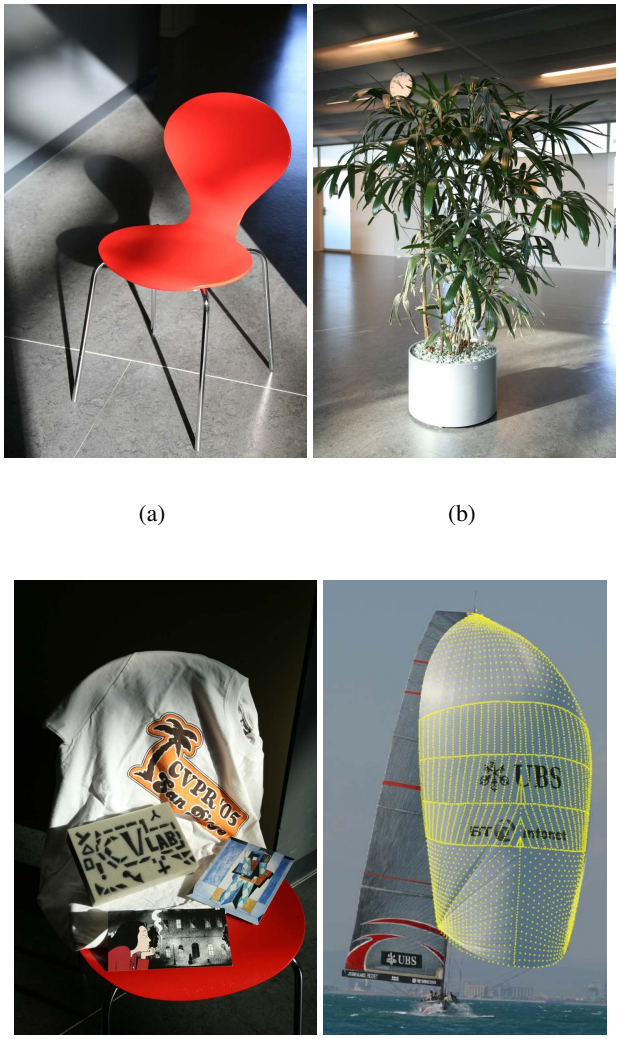
3.4. Thesis Goal and Basic Assumptions

The goal of this Thesis is to solve the registration, illumination, and occlusion problems for non-rigid surfaces augmentation under the constraints stated in Section 2.4: Using a single camera and, optionally, additional ones, not engineering the scene, providing both visual quality and ease of use. We made our design choices to come up with solutions that satisfy these constraints. The first choice is to rely on fast wide-baseline matching as a basis for registration [63, 81, 80, 62]. Since it requires a textured model to train the system, we focus on a single known object[1]. We then achieve tracking by detecting the pattern in each frame, thus avoiding manual initialization and drift.

To establish feature correspondences, the object has to be textured. To satisfy our constraints, the procedure to create the textured model should be easy. In our implementation, this is achieved by taking a single picture of the surface in a known configuration, which is easy enough for a novice user.

To handle non-rigid surfaces effectively, we place a few constraints on their geometry. The first is linked to wide-baseline matching; It requires the surface to be locally planar, which is true

[1]This contrasts with Simultaneous Localization and Mapping (SLAM) approaches that build a world representation at runtime [58], but usually assume the world to be rigid and would typically discard non-rigid surfaces as outliers.



(a) (b)

(c) (d)

Figure 3.5.: Suitability of some objects for augmentation. (a) Our framework might have trouble with the lack of texture of the chair. (b) Considering the plant as locally planar is not a good approximation; our framework is thus unable to augment it. (c) and (d) Several common objects that perfectly fulfill conditions for augmentation or measurement.

for most non-rigid surfaces. The second constraint applies to the surface topology: We work with surfaces without holes. Such surfaces have some interesting properties. They are continuous and their span of possible shapes can be modeled efficiently with regular hexagonal 2D meshes or with a linear combination of deformation modes. These choices are generic enough to cover a wide range of objects, such as those depicted in Figure 3.5(b).

In short, we assume that the texture of the target surface is known and abundant and that its geometry is know, locally planar, and continuous. These assumptions are sufficient to create methods that works for most applications we presented above.

4. Geometric Registration

Our goal is to superimpose virtual material over a real and potentially non-rigid object in such a way that the virtual elements appear to be glued to the real ones. Achieving this requires perfect recovery of the 2–D or 3–D motions and deformations of the target object. This is the issue we address in this chapter, bearing in mind that we want solutions that fit in the framework we defined in Section 2.4. It states that the only mandatory registration device is a single camera. Multiple ones, if available, can improve accuracy but remain optional. It also assumes that the object geometry and its texture are known. Moreover, augmented reality requires real-time registration. Thanks to recent computer vision developments, this challenging problem can be addressed using wide-baseline feature matching.

In the following, we review related work and explain the general principles of using wide-baseline correspondences to detect objects. We then present a robust estimation scheme that is able to eliminate outliers while handling the large number of degrees of freedom of deformable objects. We will see that it makes both 2–D and 3–D non-rigid surface detection possible. As we have seen in Section 3.1, the choice of working with two or three dimensions depends on the application. It has a significant impact on the algorithms and we therefore present different ones for both cases.

4.1. Related work

Many approaches to registering a model on an image have been proposed. Some feature-based algorithms first establish correspondences and then find the best transformation explaining them, while eliminating outliers. Others simultaneously solve for both correspondence and registration, without the need for correspondences and with or without using feature characterization. Finally, some techniques do not even rely on features. We review related techniques briefly below and discuss why they have not yet been shown to be suitable for real-time detection of deformable objects.

4.1.1. Feature-Based Registration

These approaches rely on establishing correspondences between image-features of the target object and those that can be found in an input image in which it is to be detected. These correspondences are then used to estimate the transformations. Figure 4.1 depicts detected features on an image, and Figure 4.2 the correspondences obtained by matching them with the ones of another view.

Figure 4.1.: Detecting feature points on an image. The circles' size represents the pyramid level at which points have been detected.

Establishing Correspondences Our method relies on establishing wide-baseline correspondences between a training image and an input one. To be useful, correspondences have to be insensitive to light and viewpoint changes, as well as to some amount of non-rigid deformation.

Among the many matching techniques that exist, we tested three: SIFT [69], shape context descriptors [7], and a classification-based method detailed in Appendix A and published in [80, 62, 64].

Even if these algorithms differ in speed, number of correspondences, and quality, our experiments show that the effectiveness of our robust estimation scheme is independent from the specific technique used to establish the point correspondences. However, only the classification based technique has proved fast enough for our purpose, real-time detection without loss of accuracy.

From Correspondences to Detection Whatever the matching technique used, the correspondences can then be used to detect the object in different ways. Since matching is never perfect, they require robustness to potential outliers.

The simplest is to eliminate outliers and find a globally consistent interpretation using a robust estimator. Having each local match vote for a global transformation is the approach used by the Hough transform and its many variations. This is effective for rigid objects but impractical for deformable ones because it would require far too many degrees of freedom to represent all possible transformations into a vote accumulator. The same can be said of the popular RANSAC algorithm [34]: With 25% of outliers and 100 degrees of freedom, 10^{12} samples are required to guarantee with 90% probability that at least one sample does not contain outliers [48]. As illustrated by Figure 4.3, RANSAC complexity is exponential with respect to degrees of freedom, because it relies on drawing a minimal group of correspondences, free of error.

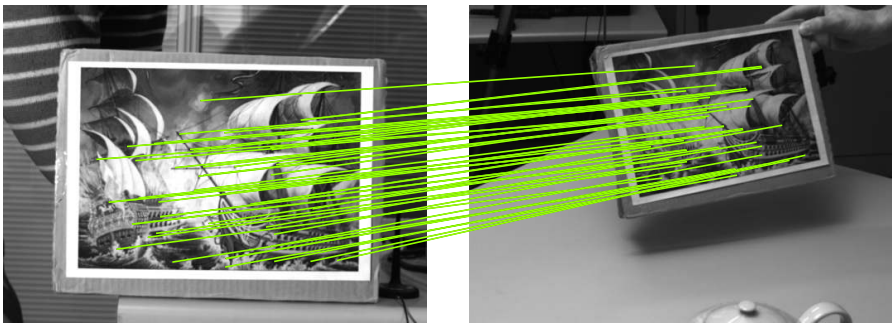


Figure 4.2.: Correspondences established automatically and validated with a RANSAC process that finds the homography relating the model image (left) and the target image (right).

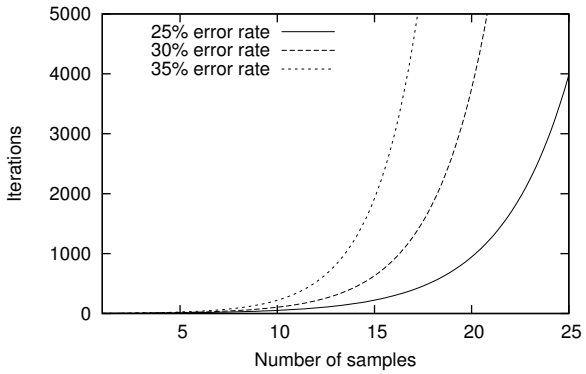


Figure 4.3.: This plot shows the number of RANSAC iterations required to reach a 95% probability of success, as a function of the minimum number of samples required to estimate model parameters. The three curves account for 25%, 30%, and 35% of outliers in the correspondence set. In practice, we often have to deal with even higher rate.

An alternative strategy is to proceed iteratively. TPS-RPM (thin plate spline - robust point matching, [17]) and EM-ICP (expectation maximization - iterative closest point, [43, 26]) are two well-known representatives of the family of algorithms that simultaneously solve for both correspondence and transformation using an iterative process. At each step, the current transformation estimate is first used to establish correspondences and assign weights to them, and, then, is refined using those correspondences. These methods use an entropy term—be it called temperature parameter, scale or blurring factor, or variance—that is progressively reduced. It controls the assignment of weights to the correspondences and has an important role in insuring convergence towards a desirable solution. As will be discussed in more detail in Section 4.2.2, our algorithm follows a similar strategy but makes use of local characterization to reduce the correspondence problem difficulty and to achieve real-time performance.

In [7], a method designed to compute a distance between shapes is presented. Shape context descriptors provide correspondences which are established one to one using bipartite graph matching. Although this method copes with some outliers and slightly different numbers of features detected on both shapes, it is not designed to extract objects from a cluttered background or to handle scale changes.

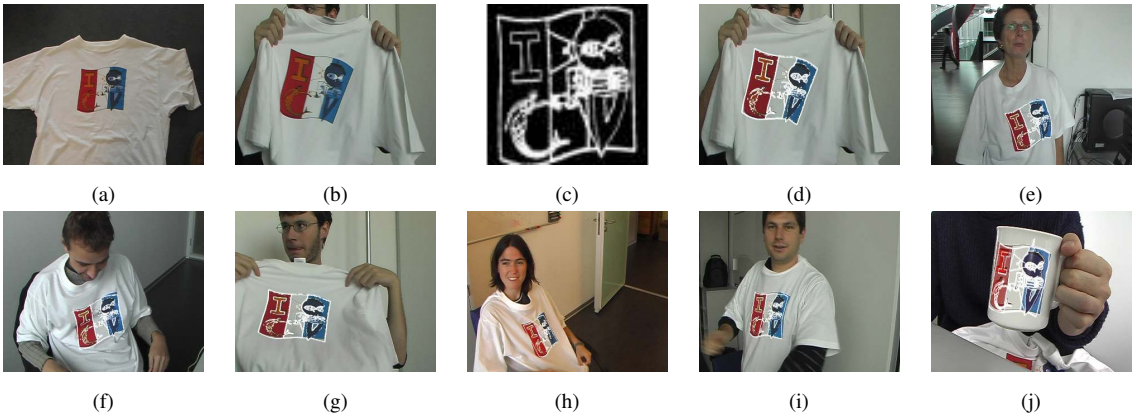
Image exploration [31] is another strategy that hooks on a first set of correspondences and then gradually explores the surrounding area, trying to establish more matches. It can handle deformable objects but this complex process is slow and takes several minutes on a 1.4 GHz computer.

4.1.2. Pixel Level Registration

For objects such as faces whose deformations are well understood and can be modeled in terms of a relatively small number of deformation parameters, fitting directly to the image data without using features is an attractive alternative to using correspondences because it allows the use of global constraints to guide the search. This has been successfully demonstrated in the context of non-rigid tracking [24, 21, 3, 97, 40] but typically requires a good initialization because the criteria being minimized tend to have many local minima.

These methods are complementary to the one proposed here: They exploit more of the texture and therefore tend to be more accurate. However, they require the initial estimate such as the one our algorithm can provide. There are in fact relatively few others that can do this for deformable objects. One of them has been proposed in [46] but requires that the whole outline be detected, which severely limits its scope. Another is the tracking of [66] that exploits the repeating properties of a near regular texture to discover new texture tiles in new frames.

Finally, the recent work presented in [116] is related to ours in two ways. First, it registers a texture composed of a few colors, typically 3 or 4, by comparing color histograms. Then, it modifies the texture on the deformed surface, while handling illumination changes. This approach to retexturing differs from ours in that we avoid limiting the number of colors present on the surface by introducing some irradiance smoothing, which yields real-time performance on both color or gray level images.



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

(f) (g) (h) (i) (j)

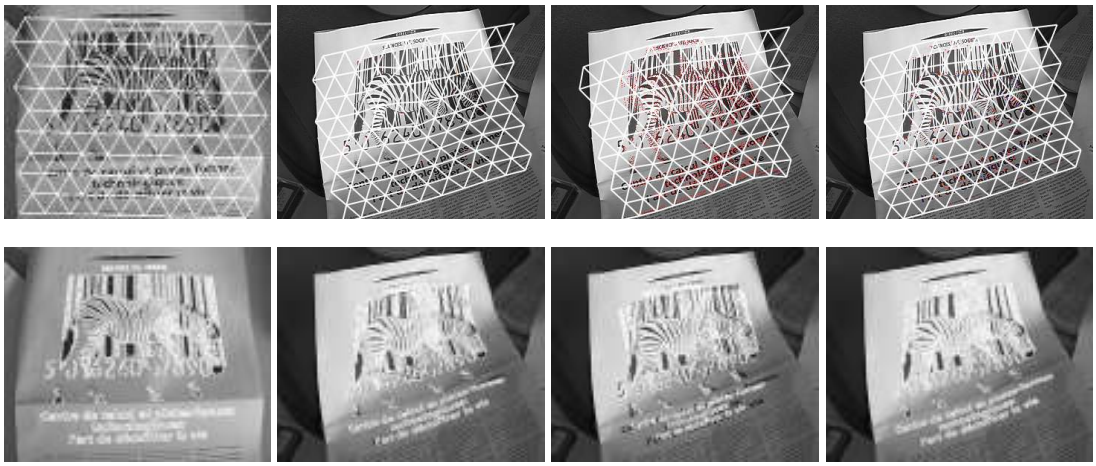
Figure 4.4.: In order to achieve surface detection, we use a model image (a). Then, our method computes a function mapping the model to an input image (b). To illustrate this mapping, we find the contours of the model using a simple gradient operator and we use them as a validation texture (c) which is overlaid on the input image using the recovered transformation (d). Additional results are obtained in different conditions (e to i). Note that in all cases, including the one where the T-shirt is replaced by a cup (j), the white outlines project almost exactly at the right place, thus indicating a correct registration and shape estimation. The registration process, including image acquisition, takes about 100 ms and does not require any initialization or *a priori* pose information.

4.2. 2–D Non-rigid Surface Detection

To detect a potentially deformable object, we rely on establishing correspondences between a model image in which the deformations are small and an input image in which they may be large. To this end, we use the fast wide-baseline matching algorithm discussed in Appendix A. Given a set $\mathcal{C}$ of correspondences between the two images, many of which might be erroneous, our problem can be formally stated as follows: We are looking for the transformation $T_{\mathbf{S}}$ mapping the undeformed model surface M into the deformed target one $T_{\mathbf{S}}(M)$ and for the subset $\mathcal{G} \subset \mathcal{C}$ of correct matches such that the sum of the squared distances between corresponding points in $\mathcal{G}$ is minimized while the deformations remain as smooth as possible. Figure 4.4 depicts some of such transformations our system is able to recover.

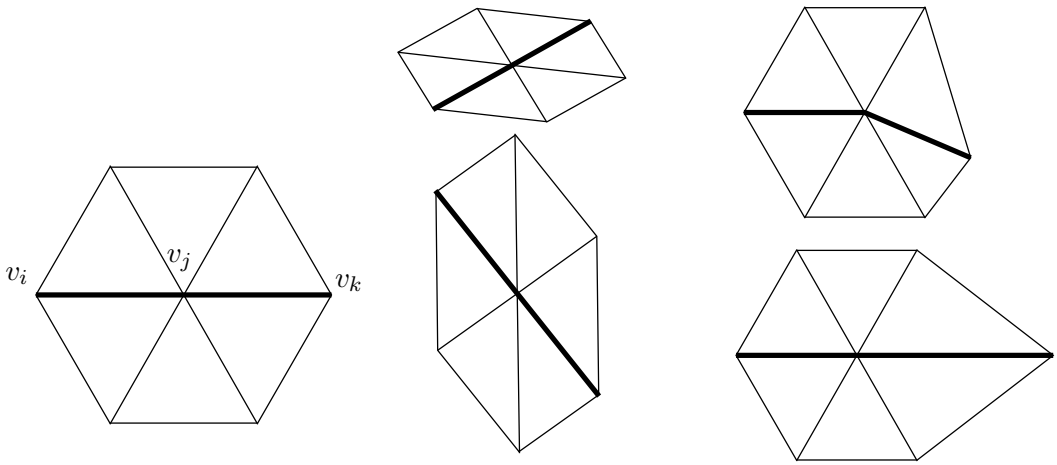
4.2.1. 2–D Surface Meshes

We represent our model M as a triangulated 2–D mesh of hexagonally connected vertices such as the one shown in Figure 4.5. The position of a vertex v_j is specified by its image coordinates (x_j, y_j). The overall shape is therefore controlled by a state vector θ that is the vector of all $\mathbf{x}$ and $\mathbf{y}$ coordinates. Given θ and the barycentric coordinates $b_i(\mathrm{p})$ of image point p that belongs



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

Figure 4.5.: Comparing three different keypoint matching algorithms. (a) Model image and validation texture shown in white. Results using: (b) Real-time classification trees, (c) shape context descriptor reimplementation, and (d) SIFT.



(a) (b) (c)

Figure 4.6.: 2D mesh models. (a) Vertex neighborhood in an undeformed hexagonal mesh. (b) Two deformations that are not penalized. (c) Two penalized deformations. Deformations resulting from perspective projection resemble those in (b) and are therefore much less severely penalized than those resulting from erroneous matches.

to a specific facet $(\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3)$ of the undeformed mesh, we define the mapping

$$T_\theta(\mathrm{p}) = \sum_{i=1}^{3} b_i(\mathrm{p}) \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad , \tag{4.1}$$

where x_i and y_i are vertex coordinates of the deformed mesh.

The mesh deforms to minimize the objective function

$$\varepsilon(\theta) = \lambda_D \varepsilon_D(\theta) + \varepsilon_C(\theta) \quad , \tag{4.2}$$

where ε_C is a data term that takes point correspondences into account, ε_D is a deformation energy that should be rotationally invariant and tend to preserve the regularity of the mesh, and λ_D is a constant discussed later. We give in Section 4.2.4 a Bayesian interpretation of $\varepsilon(\theta)$ and the following derivations.

We take $\varepsilon_D(\theta)$ to be an approximation of the sum over the surface of the square second derivatives with respect to the $\mathbf{x}$ and $\mathbf{y}$ coordinates. More specifically, let $\mathcal{E}$ be the set of vertex index triplets (i, j, k) such that $(\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3)$ form two connected and collinear edges, as illustrated by Figure 4.6(a). Since the undeformed mesh M has equidistant vertices, we have

$$\forall (i, j, k) \in \mathcal{E} : \mathrm{v}_i - \mathrm{v}_j = \mathrm{v}_j - \mathrm{v}_k \quad , \tag{4.3}$$

and therefore write

$$\varepsilon_D(\theta) = \frac{1}{2} \sum_{(i,j,k)\in\mathcal{E}} (-x_i + 2x_j - x_k)^2 + (-y_i + 2y_j - y_k)^2 \quad . \tag{4.4}$$

$\varepsilon_D(\theta)$ approximates the squared directional curvature of the surface as long as the vertices remain roughly equidistant and its value grows with the length difference of every two collinear connected edges.

This regularization term serves a dual purpose. First it *convexifies* the energy landscape and improves the convergence properties of the optimization procedure. Second, in the presence of erroneous correspondences, some amount of smoothing is required to prevent the mesh from overfitting the data, and wrinkling the surface excessively. As illustrated by Figure 4.6(b,c), ε_D is appropriate for this purpose because it tolerates 2–D affine motions but penalizes shape deformations. Of course, both those produced by perspective distortions and by the actual surface deformation tend to increase ε_D. However, this increase is insignificant when compared to those that spurious deformations resulting from erroneous matches could produce.

Equation 4.4 can be rewritten in matrix form as

$$\varepsilon_D(\theta) = \frac{1}{2} \left(\mathbf{x}^T \mathbf{K}'^T \mathbf{K}' \mathbf{x} + \mathbf{y}^T \mathbf{K}'^T \mathbf{K}' \mathbf{y} \right) \quad , \tag{4.5}$$

where $\mathbf{K}'$ is a matrix containing one row per triplet in $\mathcal{E}$ and one column per mesh vertex. The row corresponding to triplet (i, j, k) is filled with zeroes except for locations i, j and k that contain -1, 2, and -1, respectively. By replacing $\mathbf{K} = \mathbf{K}'^T \mathbf{K}'$ in Equation 4.5, we have:

$$\varepsilon_D(\theta) = \frac{1}{2} (\mathbf{x}^T \mathbf{K} \mathbf{x} + \mathbf{y}^T \mathbf{K} \mathbf{y}) \quad . \tag{4.6}$$

To minimize $\varepsilon(\theta)$, we use the semi-implicit scheme so successfully introduced in the original snake paper [56]: We look for a minimum of the energy and therefore for solutions of

$$\begin{aligned} 0 &= \frac{\partial \varepsilon}{\partial \mathbf{x}} = \frac{\partial \varepsilon_C}{\partial \mathbf{x}} + \mathbf{K}\mathbf{x} \ , \\ 0 &= \frac{\partial \varepsilon}{\partial \mathbf{y}} = \frac{\partial \varepsilon_C}{\partial \mathbf{y}} + \mathbf{K}\mathbf{y} \ . \end{aligned} \tag{4.7}$$

Since $\mathbf{K}$ is positive but not definite, given initial vectors $\mathbf{x}_0$ and $\mathbf{y}_0$, this can be solved by introducing a viscosity parameter α and iteratively solving at each time step the two coupled equations

$$\begin{aligned} \mathbf{K}\mathbf{x}_t + \alpha(\mathbf{x}_t - \mathbf{x}_{t-1}) + \left.\frac{\partial \varepsilon_C}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_{t-1},\mathbf{y}=\mathbf{y}_{t-1}} &= 0 \ , \\ \mathbf{K}\mathbf{y}_t + \alpha(\mathbf{y}_t - \mathbf{y}_{t-1}) + \left.\frac{\partial \varepsilon_C}{\partial \mathbf{y}}\right|_{\mathbf{x}=\mathbf{x}_{t-1},\mathbf{y}=\mathbf{y}_{t-1}} &= 0 \ , \end{aligned}$$

which implies

$$\begin{aligned} (\mathbf{K} + \alpha\mathbf{I})\mathbf{x}_t &= \alpha\mathbf{x}_{t-1} - \left.\frac{\partial \varepsilon_C}{\partial X}\right|_{\mathbf{x}=\mathbf{x}_{t-1},\mathbf{y}=\mathbf{y}t-1} \ , \\ (\mathbf{K} + \alpha\mathbf{I})Y_t &= \alpha\mathbf{y}_{t-1} - \left.\frac{\partial \varepsilon_C}{\partial \mathbf{y}}\right|_{\mathbf{x}=\mathbf{x}_{t-1},\mathbf{y}=\mathbf{y}_{t-1}} \ . \end{aligned}$$

Because $\mathbf{K}$ is sparse and regular, solving these linear equations using LU decomposition is fast and upon convergence $\mathbf{x}_t \approx \mathbf{x}_{t-1}$ and $\mathbf{y}_t \approx \mathbf{y}_{t-1}$. This iterative scheme therefore quickly yields a solution of Equation 4.7, even when starting with completely random guesses for $\mathbf{x}_0$ and $\mathbf{y}_0$ as will be shown in Section 4.2.5.

4.2.2. Correspondence Energy

Minimizing ε_C, the data term of Equation 4.2, tends to deform the mesh so that it matches the target object in the input image. This is achieved as follows.

Let $\mathcal{C}$ be a set of *correspondences* between the model and the input image. Its elements are of the form $c = \{\mathrm{c}_0, \mathrm{c}_1\} \in \mathcal{C}$, where c_0 represents the 2–D coordinates of a feature point in the model image and c_1 the coordinates of its match in the input image. For the sake of generality, we allow potential matches between a point in the first image and multiple points in the second, so that the corresponding c_0 may appear in several elements of $\mathcal{C}$. We write

$$\varepsilon_C = -\sum_{c \in \mathcal{C}} w_c \rho\left(\left\|\mathrm{c}_1 - T_\theta\left(\mathrm{c}_0\right)\right\|, r\right) \ , \tag{4.8}$$

where ρ is a robust estimator whose *radius of confidence* is r and $w_c \in [0, 1]$ a weight associated to each correspondence. In our experience the choice of ρ is critical to ensure the elimination of outliers and convergence towards the desired minimum while the choice of the w_c has much less impact, as will be discussed in Section 4.2.5.

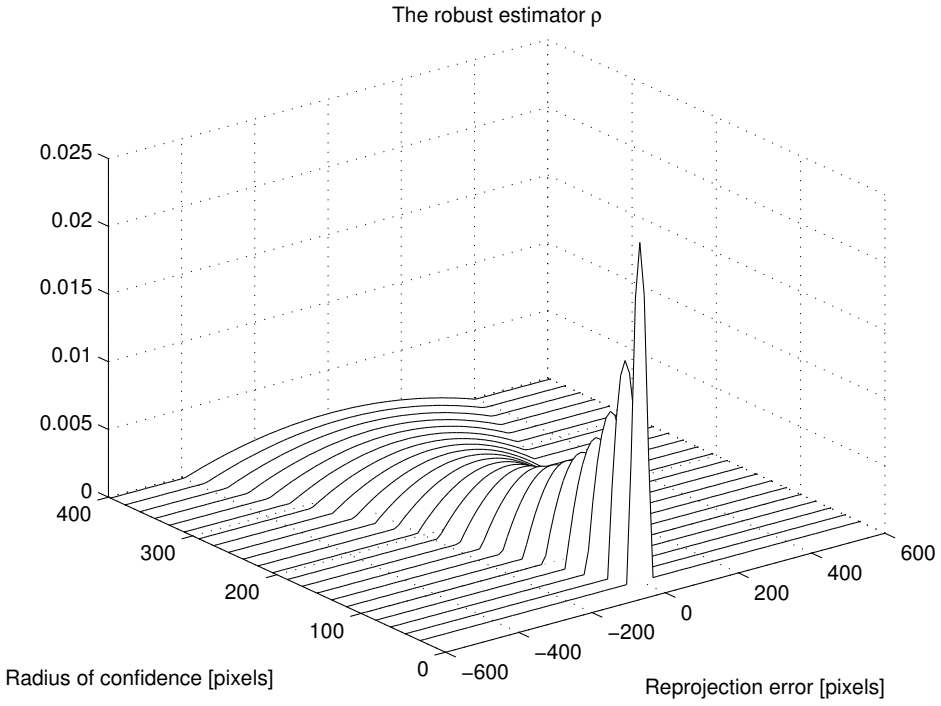


Figure 4.7.: The ρ function of Equation 4.9 is quadratic for distances smaller than the radius of confidence, elsewhere it is zero.

We take the robust estimator to be

$$\rho(\delta, r) = \begin{cases} \frac{3\left(r^2-\delta^2\right)}{4r^3} & \text{if } \delta < r \\ 0 & \text{otherwise} \end{cases} . \tag{4.9}$$

As shown in Figure 4.7, its shape is that of a quadratic ridge that gets narrower and taller when r decreases. In other words, r acts as a confidence measure. When it is large, most correspondences, potentially including poor ones, fall within this broad *ridge of confidence* and are given some weight. As r diminishes, ρ becomes more peaked and selective. This formulation has the following advantages:

- The quadratic behavior of ρ within the ridge of confidence yields a function ε_C that is easy to minimize.
- ρ is normalized so that $\int_{-\infty}^{\infty} \rho(x, r)dx = 1 \quad \forall r > 0$, which means that the ε_C term computed with any r values remain commensurate to the $\lambda_D \varepsilon_D$ term of Equation 4.2. Therefore, we do not need to adjust either the λ_D parameter or the w_c weights of Equation 4.8. This is in contrast to methods such as SoftAssign [17, 117] in which the surface rigidity must be progressively reduced according to a schedule that is not necessarily easy to synchronize with the annealing of r and may change from case to case.
- ρ has finite support so that correspondences that fall outside the radius of confidence are completely ignored and can be tagged as invalid.

These properties of the ρ estimator are what make the straightforward approach to optimization described below so effective.

4.2.3. Optimization Schedule

Minimizing ε therefore results in a mesh that moves towards the desired solution but whose progression can be blocked by outliers. To overcome this, we introduce a simple optimization schedule in which the initial *radius of confidence* $r_0 = 1000$[pixels] is progressively reduced at a constant rate $\eta = 0.5$: $r_t = \eta r_{t-1}$. For each value of r, we minimize ε and use the result as the initial state for the next minimization.

As discussed in Section 4.2.1, at each iteration of our semi-implicit optimization scheme, we evaluate the derivatives of ε_C. In this context, the fact that ρ has derivatives whose magnitude is inversely proportional to r is very beneficial: At the beginning when r is large, the gradients of ε_D are comparatively larger than those of ε_C, thus preventing erroneous matches from crumpling the surface while allowing correct and consistent ones to produce the right global deformation. As the optimization progresses and r decreases, the ρ derivatives and consequently the gradients of ε_C become larger. The triangulation starts bending as appropriate and the influence of the outliers progressively decreases.

The algorithm stops when r reaches a value close to the expected precision of the matches expressed in pixels, typically one or two. Such a deterministic algorithm is guaranteed to converge but the result might be wrong, for example because the target object is completely occluded. To

decide whether or not to believe the result, we simply count the number of correspondences that fall within the ridge of confidence of our ρ estimator. As will be shown in Section 4.2.5, this criterion is surprisingly effective at distinguishing successes from failures.

4.2.4. A Probabilistic Interpretation

We compute the T_θ mapping from the undeformed model surface into the deformed target one by minimizing the $\varepsilon(\theta)$ energy of Equation 4.2. From a strictly theoretical point of view, this is the right thing to do if $\varepsilon(\theta)$ is proportional to the negative log-likelihood of $p\left(T_\theta \mid \mathbf{I}\right)$, where $\mathbf{I}$ represents the current image.

In practice, there is no feasible way to precisely estimate $p\left(T_\theta \mid \mathbf{I}\right)$ in general. However we show that under a very reasonable and limited set of hypotheses $\varepsilon(\theta)$ is indeed a good approximation of the log-likelihood. Since they are close to being satisfied in the real world, this clarifies the assumptions that our algorithm makes and helps explain why it actually works.

Approximating the Log-Likelihood Bayes' formula yields

$$p\left(T_\theta \mid \mathbf{I}\right) \propto p\left(\mathbf{I} \mid T_\theta\right) p(T_\theta), \tag{4.10}$$

$$-\log p\left(T_\theta \mid \mathbf{I}\right) = -\log p\left(\mathbf{I} \mid T_\theta\right) - \log p(T_\theta), \tag{4.11}$$

up to a constant. Taking the $-\log p(T_\theta)$ to be the ε_D deformation energy of Equation 4.6 amounts to giving a higher prior to smooth surfaces, which is standard practice when modeling deformable surfaces. The relationship between the $-\log p\left(\mathbf{I} \mid T_\theta\right)$ term and the ε_C energy of Equation 4.8 is less obvious and needs to be examined more carefully.

Since the position of the keypoints in the input image depends only on the deformation that the surface undergoes, it is legitimate to treat each point c_1^i on the input image as independent from each other given T_θ. If the target object is sufficiently textured, we can also assume that the information provided by the keypoints extracted from $\mathbf{I}$ is sufficient to condition it. We can therefore write

$$p\left(\mathbf{I} \mid T_\theta\right) = p\left(\mathrm{c}_1^1, ..., \mathrm{c}_1^n \mid T_\theta\right) = \prod_i p\left(\mathrm{c}_1^i \mid T_\theta\right). \tag{4.12}$$

Each $p\left(\mathrm{c}_1^i \mid T_\theta\right)$ term can be computed as the disjunction over the matching possibilities :

$$\begin{array}{l} p\left(\mathrm{c}_1^i \mid T_\theta\right) = \sum_j p\left(\mathrm{c}_1^i,\, \mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^j \mid T_\theta\right) + \\ p\left(\mathrm{c}_1^i, \mathrm{c}_1^i \text{ unmatchable} \mid T_\theta\right), \end{array} \tag{4.13}$$

where $\mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^j$ denotes the binary variable "is c_1^i corresponding to the point c_0^j on the model?". c_1^i can either correspond to any keypoint extracted from the model image or be a keypoint that is on the background or on the target object but not detected in the model image. We write $p(c_1^i, c_1^i \text{ unmatchable} \mid T_\theta)$ as a uniform distribution of parameter λ which allows for outlier matches. We have

$$\begin{array}{rl} & p\left(\mathrm{c}_1^i,\, \mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^j \mid T_\theta\right) \\ = & p\left(\mathrm{c}_1^i \mid \mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^j,\, T_\theta\right) \times p\left(\mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^j \mid T_\theta\right). \end{array} \tag{4.14}$$

In our implementation, we take the second term $p\left(\mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^j \mid T_\theta\right)$ to be equal to 1 when it is likely that c_1^i corresponds to c_0^j, 0 otherwise. This choice is based solely on appearance and is made by our feature point recognizer [63]. A more complex expression measuring how similar the points look could have be retained, but we chose this one for simplicity. Note that several c_0^j points can yield $p\left(\mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^j \mid T_\theta\right)$ since we retain multiple correspondences for a single point.

The first term $p\left(\mathrm{c}_1^i \mid \mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^j, T_\theta\right)$ is taken to be equal to $\mathcal{N}(T_\theta(\mathrm{c}_0^j) \mid \mathrm{c}_1^i, \mathbf{\Sigma}_{c_1})$, and models the geometric relation and represents the uncertainty resulting from mapping c_0^j on c_1^i according to T_θ as a normal distribution of covariance $\mathbf{\Sigma}_{\mathrm{c}_1}$ and mean c_1^i. Its theoretical value could be derived using standard covariance propagation formulas as $\mathbf{\Sigma}_{\mathrm{c}_1} = \mathbf{J}_{T_\theta}\mathbf{\Sigma}_{T_\theta}\mathbf{J}_{T_\theta}^\top + \mathbf{J}_{\mathrm{c}_0}\mathbf{\Sigma}_0\mathbf{J}_{c_0}^\tau + \mathbf{\Sigma}_0$, where $\mathbf{J}_{\mathrm{c}_0} = \frac{\partial \mathrm{c}_1}{\partial \mathrm{c}_0}$, $\mathbf{J}_{T_\theta} = \frac{\partial \mathrm{c}_1}{\partial \theta}$, and $\mathbf{\Sigma}_0$ is the expected uncertainty of the extracted keypoint locations. Representing the uncertainty of the current estimate of T_θ as a normal distribution of covariance $\mathbf{\Sigma}_{T_\theta}$ then yields an analytical expression of $\mathbf{\Sigma}_{c_1}$ as a function of $\mathbf{\Sigma}_{T_\theta}$. In short, we can rewrite the logarithm of the probability of Equation 4.13 as

$$-\log p\left(\mathrm{c}_1^i \mid T_\theta\right) = \sum_{k=1}^{N^i} \log \mathcal{N}(T_\theta(c_0^{j_k}) \mid \mathrm{c}_1^i, \mathbf{\Sigma}_{\mathrm{c}_1^i}) + \lambda \tag{4.15}$$

where the $\mathrm{c}_0^{j_k}$ are the potential matches of c_1^i.

The expression of Equation 4.15 being unwieldy for optimization purposes, we seek an approximation with a quadratic formulation to ensure the good behavior of the minimization process.

Let us first consider the case where c_1^i has only one potential match, which means $N^i = 1$ in Equation 4.15. The log-likelihood can then be approximated either by the Tukey robust estimator [51] or even more simply as

$$-\log p\left(\mathrm{c}_1^i \mid T_\theta\right) \propto \begin{cases} \|c_1^i - T_\theta(c_0^{j_1})\|^2 & \text{if } \|c_1^i - T_\theta(c_0^{j_1})\|^2 < r^2 \\ r^2 & \text{otherwise} \end{cases}, \tag{4.16}$$

where r is a threshold that depends on Σ_{c_1}. This is justified by the fact that when computed sufficiently far from the mean, the Gaussian value is negligible with respect to λ. Because Σ_{c_1} is depending on values difficult to estimate, namely Σ_0 and Σ_{T_θ}, it is difficult to have a good estimate of $\mathbf{\Sigma}_{\mathrm{c}_1}$, and in practice, a fixed threshold is used. However it is expected that the error on T_θ represented by $\mathbf{\Sigma}_{T_\theta}$ is reduced after each iteration. Our method, by regularly reducing the radius of our robust estimator, enforces this fact while ensuring convergence. Our robust estimator is a scaled and translated version of the classic function of Equation 4.16. Large radius values correspond to spread, low Gaussian distribution and large errors for T_θ, and small radius values correspond to peaked, high Gaussian distribution and small errors for T_θ.

When c_1^i can potentially have multiple matches, which means $N^i > 1$ in Equation 4.15, we have shown experimentally that all of the schemes discussed in Section 4.2.5 yield similar results. They correspond to different ways of computing the probability $p\left(\mathrm{c}_1^i \mid T_\theta\right)$. It is relatively easy to see that taking all the weights w_i equal to 1 assumes that the different Gaussians do not

overlap when they are not negligible compared to λ. This is obviously wrong particularly at the beginning of the minimization when the radius is large, both in practice and in our synthetic experiments. Apparently it does not really affect the minimization. The ICP and EM-ICP methods both take into account the different hypotheses for one model point: ICP takes the input image feature closest to the reprojected model points. The EM-ICP version normalizes the probabilities such that $\forall m : \sum_i p\left(\mathrm{c}_1^i \leftrightarrow \mathrm{c}_0^m \mid T_\theta\right) = 1$. In SoftAssign, the normalization is not only done over all image points, but also over all model points, thus enforcing one to one correspondences. This method is more accurate but takes more time to compute.

Validity of our Hypotheses The above derivation highlights the implicit hypotheses that our choice of energy function entails. Some are standard and widely accepted: For textured objects, the image information can be summarized by keypoints that are assumed to be independent from one another given a transformation, and the prior on the shape parameters favors the smoothest deformations. In fact, we effectively make only two non-standard assumptions:

1. We assume that the uncertainty on the estimate of T_θ regularly decreases as the minimization progresses, whereas previous approaches aim at estimating this uncertainty from the optimization result. Not only is it legitimate to assume the uncertainty is reduced during optimization, but our experiments make us think it actually helps the optimization to converge. This is corroborated by the results in [88].

2. Our implementation implicitly assumes that the uncertainties associated to several possible matches of a keypoint are independent, which clearly is too strong an assumption. However, as shown by the experiments of Section 4.2.5, weakening it increases the computational cost without significantly improving performance.

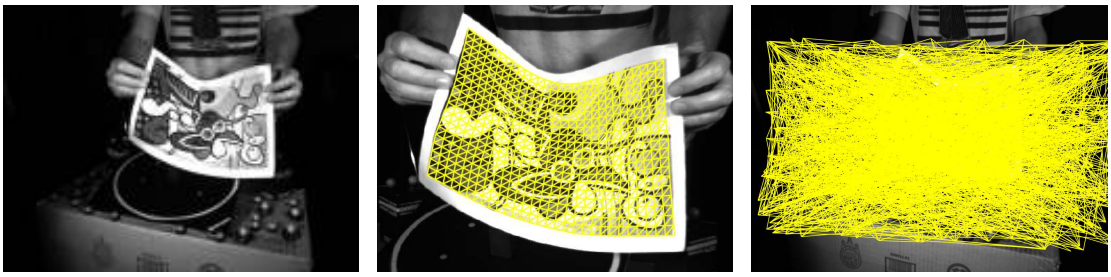
4.2.5. Algorithm Properties

In this section, we use synthetic data to illustrate the effectiveness of our implementation choices. More specifically we show that our algorithm is insensitive to parameter choices, insensitive to initial conditions, and effective at rejecting false matches.

Figure 4.8 depicts our approach to creating synthetic data for these experiments. We fed our algorithm with manually established correspondences between a model image in which the sheet of paper is flat, and the image of Figure 4.8(a) until we obtained the 600-vertex deformed mesh of Figure 4.8(b), which projects correctly over its whole surface. We treat this mesh as our reference, which can be viewed as the ideal result that can be expected from our algorithm. In the remainder of this section we will use different sets of correspondences, randomized initial conditions, and modified parameter settings. They produce different results that can then be compared to our reference. Proceeding in this manner ensures that the deviations we measure are strictly related to what we are trying to measure, as opposed to pose dependent problems.

Measuring Success We define three objective success criteria:

$\mathbf{C}_1$ 90% of the mesh vertices are within 2 pixels of those in the reference mesh.



(a) (b) (c)

Figure 4.8.: Image and meshes used for our synthetic experiments. (a) Original image. (b) Reference mesh computed using hand-picked correspondences. (c) A random initial configuration.

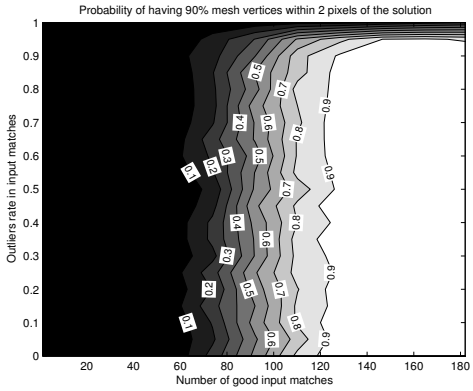
$\mathbf{C}_2$ 50% of the mesh vertices are within 2 pixels of those in the reference mesh.

$\mathbf{C}_3$ At least 90% of the valid correspondences given as input are correctly labeled as such by the robust estimator, as discussed in Section 4.2.2.

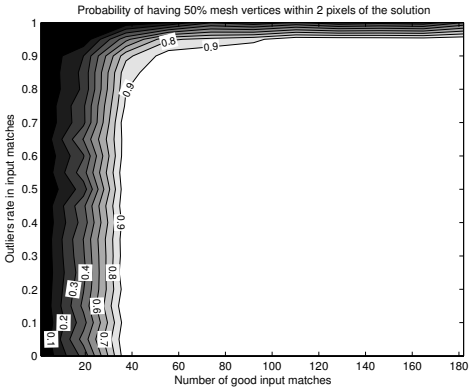
Given that the test image is of dimension 1024×768, $\mathbf{C}_1$ and $\mathbf{C}_2$ rate the algorithm's accuracy and $\mathbf{C}_3$ its ability to discriminate valid correspondences from spurious ones. The 90% figure in $\mathbf{C}_1$ eliminates cases where a substantial part of the mesh is incorrectly reconstructed, even though the algorithm may have done a good job on the rest, a case that $\mathbf{C}_2$ labels as correct.

To test our algorithm, we ran it about one hundred thousand times with random initial conditions, such as the one of Figure 4.8(c) that is very far from the solution, and using synthetic sets of correspondences containing varying numbers of valid matches and percentages of erroneous ones.

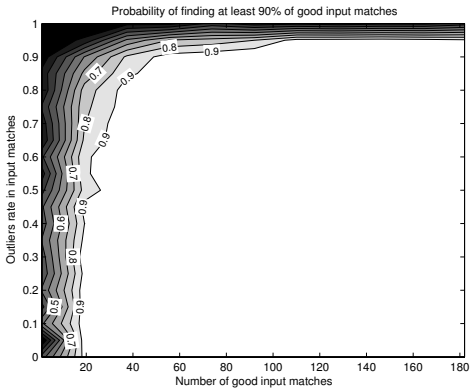
Accuracy and Robustness Figure 4.9 depicts the success rates according to the $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$ criteria introduced above as a function of the number of valid correspondences and of the outlier rate. In each plot, the color depicts the percentage of results that meet the corresponding criterion. The black wiggly lines represent level lines in this probability landscape. Note that in all three plots, they are nearly vertical for outlier rates up to 90%, thus indicating that the performance does not significantly degrade before then. Given that $\mathbf{C}_1$ is much more stringent than $\mathbf{C}_2$, it is natural that it requires more valid matches to achieve the required level of precision. To recover 50% mesh vertices location, 40 matches are enough and 120 for 90%. This is very encouraging considering that the mesh has 600 vertices, which would imply 1200 degrees of freedom in the absence of regularization constraints. $\mathbf{C}_3$ is the less demanding of the three criteria and requires less than 15 to 20 valid correspondences. However, success in terms of $\mathbf{C}_3$ does not guarantee accuracy for large outlier rates because, even though the algorithm still finds most of the inliers, it starts mistakenly tagging outliers as valid matches, which degrades the precision.



(a) $\mathbf{C}_1$ criterion



(b) $\mathbf{C}_2$ criterion



(c) $\mathbf{C}_3$ criterion

Figure 4.9.: Probability of success according to the three criteria of Section 4.2.5 as a function of the number of valid input matches, on the horizontal axis, and the outlier rate, on the vertical axis. White indicates values close to one and black close to zero.

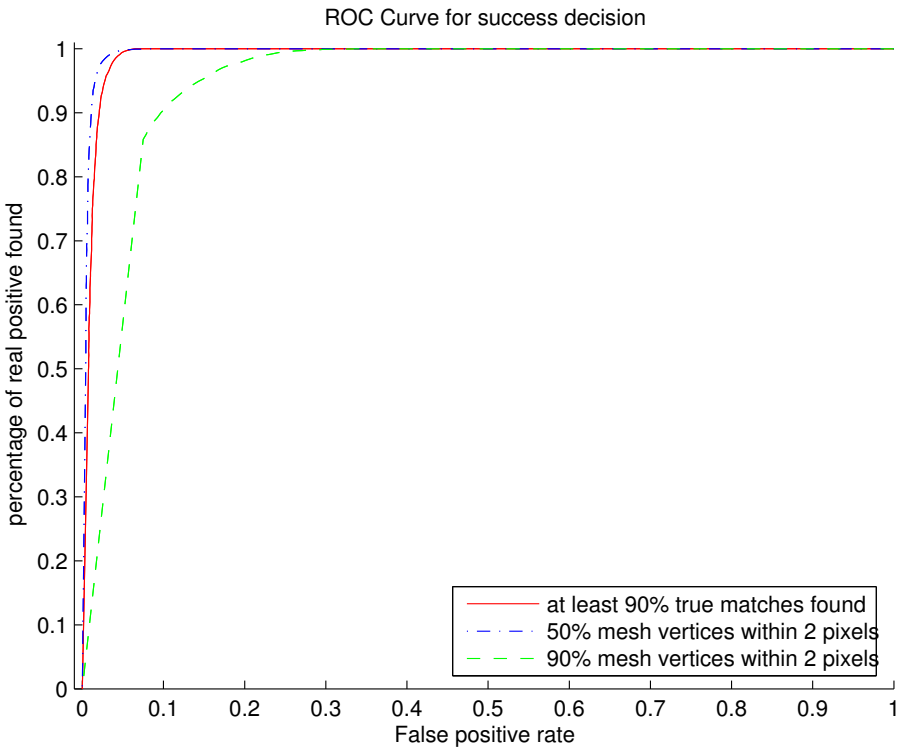


Figure 4.10.: Self diagnostic ROC curves. We accept a detection result based on the number of matches tagged as valid by our robust estimator. We plot one curve for each one of the three criteria of Section 4.2.5. For each one, as the threshold for accepting a result is lowered, both the false-positive rate, on the x-axis, and the true-positive rate, on the y-axis, increase.

Self Diagnostic So far, we have compared our results against a manually obtained reference mesh. In practice, the algorithm has to self-diagnose its own successes and failures in the absence of any such reference. As a substitute, we use the absolute number of matches that are tagged as valid by the ρ estimator of Equation 4.9 as a measure of success. In other words, our algorithm declares a successful detection when the number of valid matches is above a given threshold. In Figure 4.10, we plot the corresponding ROC curves according to $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$. These curves indicate an excellent correlation between "objective" success, as measured by comparison to a reference result, and "subjective" success, as measured by the number of matches tagged as valid.

One limitation of the current approach, however, is that we only implemented a global success measure: The surface is either completely found or not at all. An interesting extension would be to measure partial success, for example in cases where the surface is partly occluded, by checking sub-areas as opposed to the whole surface.

Disambiguating Multiple Matches Recall from Section 4.2.2 that a point from the model image can have several potential matches in the input image. One can simply rely on the progressively decreasing r radius of confidence of the ρ estimator to disambiguate those cases. Alternatively one could use a more sophisticated weighting scheme, an option we explore here by setting the w_c weights of Equation4.8 in one of the five following ways:

1. $w_c = 1$ for all correspondences,
2. $w_c = 1$ for the closest match, and zero to all others as in ICP,
3. as in EM-ICP [43], with $\sigma = \frac{r}{3}$:
$$w_c = \frac{\exp\left(-\|\mathrm{c}_1 - T_\theta(\mathrm{c}_0)\|^2/2\sigma^2\right)}{\sum_{d\in\mathcal{C},\mathrm{d}_o=\mathrm{c}_o}\exp\left(-\|\mathrm{d}_1 - T_\theta(\mathrm{d}_0)\|^2/2\sigma^2\right)},$$
4. a variation of EM-ICP in which the Gaussian is replaced by ρ:
$$w_c = \frac{\rho(\|\mathrm{c}_1 - T_\theta(\mathrm{c}_0)\|, r)}{\sum_{d\in\mathcal{C},\mathrm{d}_o=\mathrm{c}_o}\rho(\|\mathrm{d}_1 - T_\theta(\mathrm{d}_0)\|, r)},$$
5. a weight computed by normalizing rows and columns of the correspondence matrix, as in SoftAssign [17].

Figure 4.11 summarizes the result of this experiment. We used 150 valid matches and a variable number of spurious matches. We plot success rates according to the $\mathbf{C}_1$ criterion as a function of the percentage of outliers. Note that these curves correspond to a vertical slice of Figure 4.9(a) and are very close to each other. For our specific purpose, but obviously not in a more general context, their respective performances are almost indistinguishable, but not their computational costs. In our real-time implementation, we therefore use the simplest one and set all w_c to one.

Parameters and Initial Conditions

We now turn to the influence of our parameter choices and of the initial conditions. We show that they influence the speed at which the algorithm converges much more than its final result.

Regularization Weight In the $\varepsilon(\theta)$ total energy of Equation 4.2, the relative influence of the regularization and observation terms is controlled by the λ_D parameter. It represents surface stiffness: The larger it is, the more deformations are penalized. If it is too large, legitimate bending might be prevented. If it is too small, the mesh may wrinkle excessively and treat some spurious correspondence as valid. In Figure 4.12, we again use a fixed number of valid matches and plot success rates according to the $\mathbf{C}_1$ criterion as a function of the percentage of outliers and of the λ_D value used to perform the computation. For outlier rates below 60%, and even up to 80%, λ_D can be chosen in a very wide range without significantly affecting the results. As the outlier rate increases, larger λ_D values appear to give better results. It is to emphasize these large values of λ_D that we chose to plot $\frac{1}{\lambda_D}$ on the vertical axis of the graph.

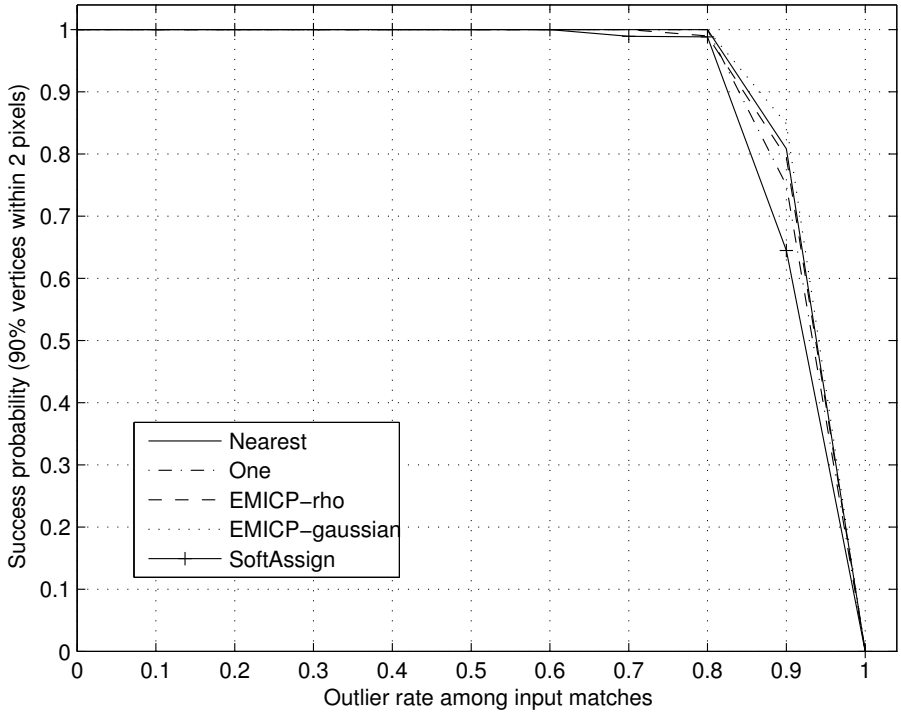


Figure 4.11.: Comparing weighting schemes. Success rate as a function of erroneous correspondences percentage, for each one of the five schemes described in Section 4.2.5.

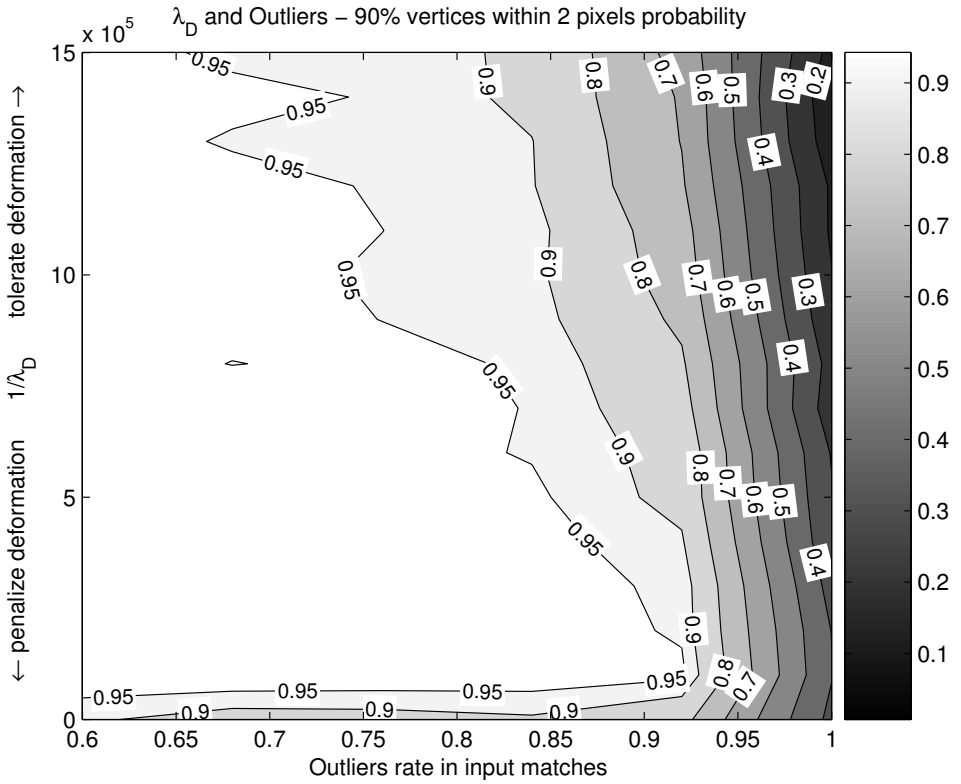
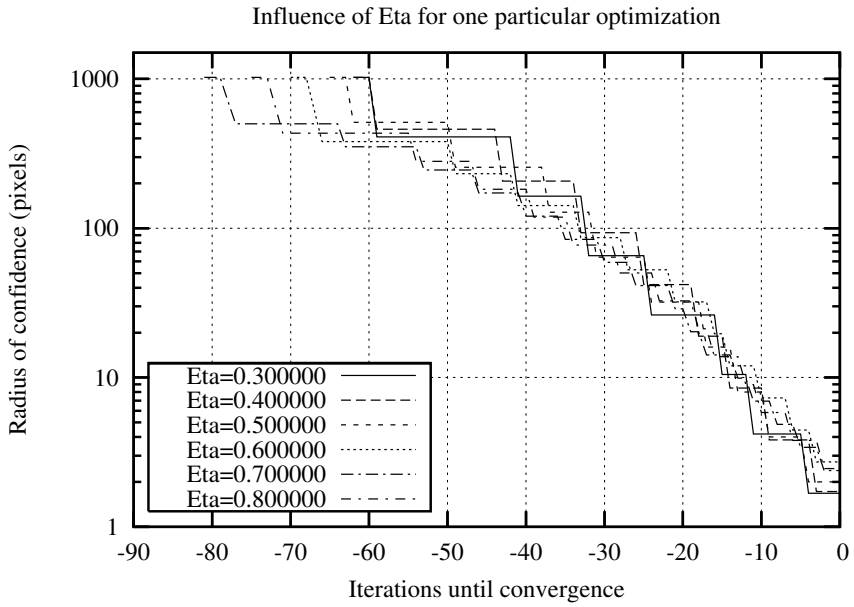
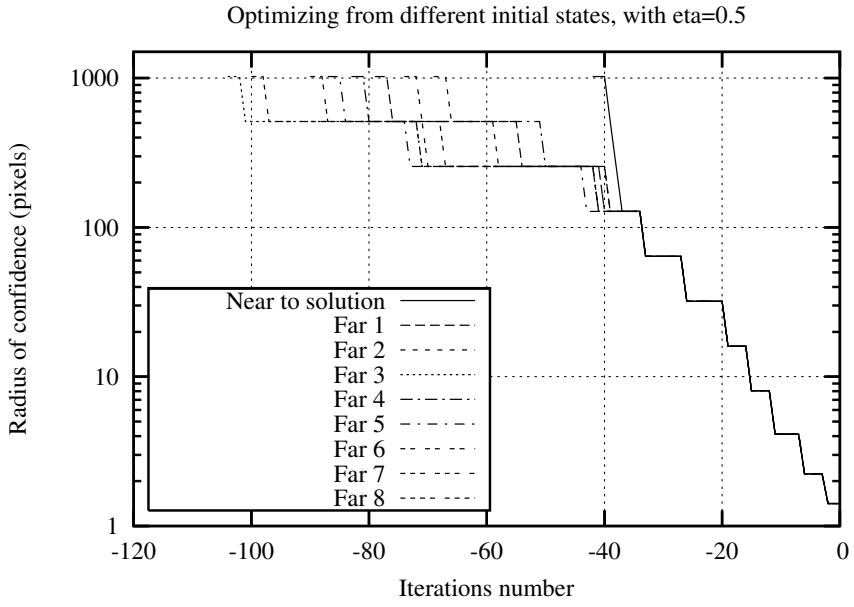


Figure 4.12.: Probability of success given $\frac{1}{\lambda_D}$ and the input outlier rate. The choice of λ_D is only important for very high outlier rates.



(a)



(b)

Figure 4.13.: Number of individual Levenberg-Marquardt steps to achieve convergence. (a) For several values of η, number of steps required to minimize the total-energy of Equation 4.2 for each successive value of r. (b) Similar plot for nine initial states, chosen to be increasingly far from the reference mesh.

Deterministic Radius Reduction In our algorithm, the confidence radius r of Equation 4.9 is decreased by a factor η after each minimization. Even though this deterministic approach might seem simplistic, we prefer it because, in practice, it is very hard to evaluate a new radius from currently valid matches. For example, a similar problem is solved in [88] using an expectation minimization (E-M) approach and the authors have to "*give a kick downwards*" to their blurring factor when E-M converges too early. Here we show that the value of η has relatively little impact on the optimizer's behavior.

To this end, we randomly chose one particular set of correspondences and ran the optimizer several times using η values ranging 0.3 and 0.8. In Figure 4.13(a), for each trial, we plot the number of individual optimization steps performed to minimize the total-energy of Equation 4.2 for each successive value of r. Note how similar the curves are. The total number of optimization steps required to locate the surface does not change much. If the radius decreases slowly, the optimizer will use more radius values but will require fewer iterations at each. If the radius decreases faster, the situation is reversed but the global outcome is similar.

Sensitivity to Initial Conditions Because our algorithm appears to be very effective at avoiding local minima, the choice of initial condition has little bearing on success or failure. It does however have an influence on the time required to achieve convergence.

To demonstrate this, we again randomly picked a set of correspondences and ran the optimizer several times using nine different initial conditions, chosen to be increasingly far from the reference mesh. The algorithm yielded the same result in all cases and Figure 4.13(b) depicts the number of individual optimization steps performed for each successive value of r during each run. Starting close to the solution saves iterations for large values of r but not for small ones. Nevertheless, this behavior could obviously be exploited in a tracking context where a good initial estimate is usually available.

4.2.6. Results

The method has been tested in conjunction with three different feature point recognizers: The publicly available SIFT implementation [69], a reimplementation of shape context characterization [7], and a classification-based method [63]. SIFT provide fewer but more accurate matches than shape contexts. The classification-based approach produces correspondences comparable to SIFT but does it faster. Because our technique is robust, the results are almost indistinguishable whatever the matching method used, as shown in Figure 4.5. However, because the classification-based method is much faster than the others, it is only when using it that we obtain true real-time performance. In this example, the algorithm runs at 10 frames per second on a 2.3 GHz laptop. Furthermore, because the point matcher is relatively insensitive to light changes and motion blur, they do not hinder the registration process.

Since we work in each frame individually, we can find objects as soon as they become visible and our method is robust to both perspective distortion and severe deformations. In the example of Figure 4.4, the ICCV logo on the shirt is detected very quickly and well before its deformation has become roughly planar. Similarly, the logo is equally well detected when worn by different

people or seen on the ICCV mug. Figure 4.14 depicts similar speed and robustness to deformations when detecting a piece of foam. For well textured objects, we get no false positives and only false negatives when the deformations or occlusions are so severe that the target object is almost impossible to make out.

Figure 4.14.: Deforming a piece of foam. The model image (top left) was used to produce the validation texture (top right) which is overlaid on the results of the right column.

4.3. 3–D Non-rigid Surface Detection

We will show that the robust estimation scheme of previous section is not restricted to 2–D. It is possible to replace the 2–D hexagonal mesh of Section 4.2.1 with a 3–D surface parameterization and a camera projection model, resulting in a system that fully recovers 3–D deformations. However, without a strong model, 3–D detection and shape recovery of a non-rigid surface from monocular images is a severely underconstrained problem.

First, the three-dimensional problem has many more degrees of freedom. The six camera pose parameters add to the ones accounting for 3–D shape. Second, different 3–D shape and pose configuration can lead to very similar 2–D projections, making detection from a single view ambiguous.

Extension to 3–D therefore requires a sound deformation model. It reduces the deformation search space, ideally restraining it to the only physically possible shapes. Hereafter, we combine Salzmann's surface parameterization first published in [90] and our robust estimation scheme. It yields an efficient algorithm that is still sometimes underconstrained if the number of wide-baseline matches is too low, for example because some surface area lacks texture. We quickly explain how to use frame-to-frame feature matching and silhouette detection to improve results. Finally, we present evaluation and results at the end of the section.

4.3.1. Related Work

Detecting and tracking 3–D surface deformations in monocular video sequences requires deformable models to constrain the search space and make the problem tractable.

Such models have been created for feature point-based structure from motion [108, 107, 67, 120] by tracking feature points and using them to learn both shape and motion. While effective, these algorithms assume that a fixed set of feature points can be reliably tracked and are not designed to exploit other sources of image information or to use known surface properties to recover the shape far away from those feature points. This typically requires explicit surface modeling using as few degrees of freedom as possible.

One way to achieve this is to only consider the motion of a few control points. Free-form deformations [98, 30, 77] are a good example of this kind of approach but there is currently no automated way to create appropriate sets of deformation modes or control points. Physics-based models are potentially more generic. The original ones [56] were 2–D and have been shown to be effective for 2–D deformable surface registration [5]. They were soon adapted for 3–D surface modeling purposes by using deformable superquadrics [106, 75], triangulated surfaces [18], or thin-plate splines [74]. In this framework, modeling generic 3–D surfaces often requires many degrees of freedom that are coupled by regularization terms. In practice, this coupling implicitly reduces the number of degrees of freedom, which makes these models robust to noise and is one of the reasons for their immense popularity. This reduction can also be explicitly achieved via modal analysis [82, 18, 25]. The cartographic work of [38] represents 3–D surfaces as hexagonal meshes that deform to minimize an energy that was the sum of an image-data term and a quadratic regularization term. This proves very effective for cartographic modeling, which is essentially 2.5–D as opposed to fully 3–D. But, it turns out to be insufficient for robust monocular video-based tracking of deformable surfaces.

Since accurately capturing the physics of deformable surfaces in a dynamical model is difficult, example-based approaches are an attractive alternative. They involve creating a database of representative shapes and using them in conjunction with a dimensionality reduction technique to learn a low-dimensional model. Active appearance models [19] pioneered this approach in the 2–D case and have since been extended to 3–D [73]. Morphable models [14] rely on the same philosophy to build 3–D face models: The database is made of 3–D meshes that were fitted to laser scans and then registered to each other. Similar approaches were successfully used to learn models of articulated motion [13, 101]. However, in all these cases, gathering and registering enough examples to build a meaningful database represented a very significant amount of work. The difficulties involved in creating the databases have limited the spread of these example-based approaches.

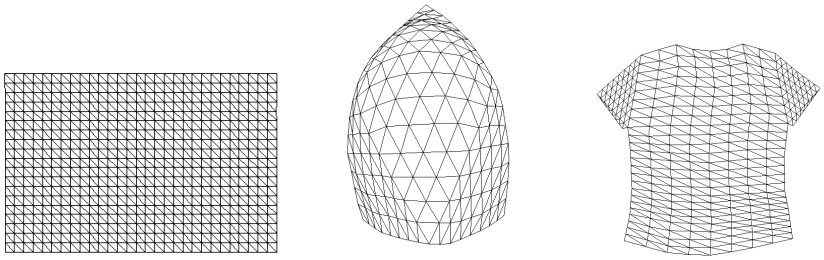
4.3.2. 3–D Surface Parameterization

Dimensionality reduction is key to make 3–D non-rigid registration tractable and greatly helps towards canceling the ambiguities inherent in monocular 3–D recovery. The approach of [92] creates low-dimensional models of deformable 3–D surfaces that can be represented as 3–D meshes of genus zero. Given the possibly non-planar rest shape of the mesh, constraining its edges to retain their original lengths implies that all possible deformations are entirely specified by a small subset of the angles between its facets. This implies that the manifold of all possible deformations can be effectively sampled by randomly setting a limited number of angles. This, in turn, generates a database of deformed shapes with identical topologies. A standard dimensionality reduction technique produces the low-dimensional 3–D deformation models that we need for tracking and detection purposes.

The inextensible triangulations we consider can be thought of polyhedra made of metal plates and whose edges have been replaced by hinges. Such polyhedra have been extensively used in the classroom to teach elementary geometry but not in our field. Nevertheless, they can assume a surprisingly large range of shapes and, therefore, produce representative shape databases. Thus this approach can be used to recover the deforming 3–D shape of such diverse objects as a T-shirt, a sheet of paper, a sail, or an elastic surface. Even though these have very different physical properties, the model has the right degrees of freedom to capture their deformations, even when they are not isometric [54], and to take full advantage of available image information.

The angle-based parameterization of [92] reduces the number of parameters required to specify the shape of an inextensible mesh. However, it is not particularly well adapted to fitting surfaces to images. It is therefore only used as an intermediate representation that samples the set of possible shapes by randomly drawing the angles from a uniform distribution between two bounds. Since all the resulting deformed meshes have the same topology, we form a $3V$ vector for each one by concatenating the coordinates of its V vertices in the vector $\mathbf{v}$. By running PCA on these vectors and retaining only the first $N_c << 3V$ principal components, we can approximate the vector of coordinates of any mesh as

$$\mathbf{s}(\alpha_1, .., \alpha_{N_c}) = \bar{\mathbf{s}} + \sum_{k=1}^{N_c} \alpha_k \mathbf{s}_k \quad , \tag{4.17}$$



(a) (b) (c)

Figure 4.15.: Hexagonal triangulations. (a) Rectangular mesh used to model the piece of paper. (b) Triangular mesh used to model the spinnaker. (c) Stitching a rectangular patch for the body part and two triangular ones for the sleeves lets us model the T-shirt.

where $\bar{\mathbf{s}}$ is the vector corresponding to an undeformed mesh, the $\mathbf{s}_k$ are the principal components or modes, and the α_k are weights that specify the surface shape. In other words, the shape of a mesh can now be expressed as a function of the vector $\theta_\alpha = \{\alpha_1, ..., \alpha_{N_c}\}$.

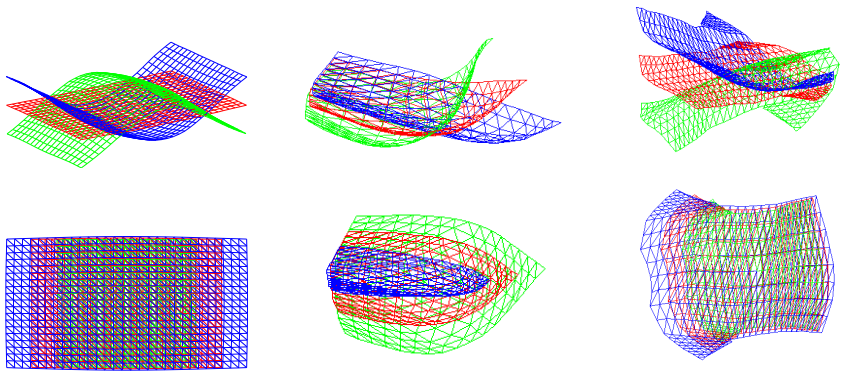
Figure 4.16 depicts the influence of two of the most significant modes in the case of the meshes of Figure 4.15. Giving weight to the first produces bending and, to the second, extension. The presence of extension modes may seem surprising since all the samples we used to learn the model are instances of the same inextensible mesh. However, given that the deformations are not linear when expressed in terms of 3–D coordinates, there is no reason for the manifold of all resulting shapes to lie on a hyperplane. Intuitively, by using PCA, we consider the N_c-dimensional ellipsoid that includes this manifold without being limited to it. This produces not only extension modes but also rigid modes that we discard.

In practice, the presence of these extension modes makes the method more general: On the one hand, if the surface whose deformations we seek to recover is truly inextensible, we can incorporate a term that prevents extension or shrinking into our optimization scheme. On the other hand, the presence of the extension terms lets us effectively model stretchable materials using a low-dimensional deformation model.

Any point p lying on the rest surface $\bar{\mathbf{s}}$ can be expressed with its barycentric coordinates $b_i(\mathrm{p})$ and the three vertices s_i forming the facet $\mathcal{F}_\mathrm{p}$ on which p lies. Given some deformation parameter θ_α, the point p follows the surface to reach the point

$$f_{\theta_\alpha}(\mathrm{p}) = \sum_{i \in \mathcal{F}_\mathrm{p}} b_i(\mathrm{p}) \mathrm{s}_i(\theta_\alpha) \quad . \tag{4.18}$$

Since $\mathbf{s}(\theta_\alpha)$ forms a linear combination (see Equation 4.17), $\mathbf{s}(\mathbf{0}) = \bar{\mathbf{s}}$ and $f_\mathbf{0}(\mathrm{p}) = \mathrm{p}$. The function f, parameterized with θ_α, densely transforms all points p on the rest surface to some location of the deformed surface. Next section shows how to integrate this parameterization into an optimization scheme that recovers the shape from one or more views of the surface.



(a) (b) (c)

Figure 4.16.: The deformation modes of the meshes of Figure 4.15, computed by PCA of generated examples [92]. In all figures the average mesh is the middle one. The other two are obtained by taking a single α_k to be non zero. Positive and negative values of that α_k yields the shown meshes. (a) Bending and extension modes of the flat rectangular mesh, (b) the triangular spinnaker, and (c) the T-shirt.

4.3.3. Optimization Scheme

The shape of the mesh is controlled by the θ_α vector of weights assigned to the PCA modes. To handle the potentially moving camera, or cameras, we introduce a vector of extrinsic parameters κ for each one and define the state vector

$$\theta_3 = (\kappa_1, \ldots, \kappa_C, \theta_\alpha)^T \quad , \tag{4.19}$$

where C is the number of cameras being used. Note that this formulation can handle both one single camera and multiple cameras that may move with respect to each other. A camera i, whose rotation and translation are parameterized by κ_i, can project a 3–D point p on the image point $\mathrm{q} = \kappa_i(\mathrm{p})$. Therefore, the projection of the deformed point $f_{\theta_\alpha}(\mathrm{p})$ on camera i is

$$T^i_{\theta_3}(\mathrm{p}) = \kappa_i\left(f_{\theta_\alpha}(\mathrm{p})\right) = \kappa_i\left(\sum_{j\in\mathcal{F}_\mathrm{p}} b_j(\mathrm{p})\mathrm{s}_j(\theta_\alpha)\right) \quad , \tag{4.20}$$

which is a function mapping surface points to image points. It plays the same role as the T_θ function of Equation 4.1 in the 2–D case. The expression of the correspondence energy term in the 3–D case is then:

$$\varepsilon_\mathcal{C}(\theta_3) = -\sum_{c\in\mathcal{C}} w_c \rho\left(\left\|\mathrm{c}_1 - T_{\theta_3}\left(\mathrm{c}_0\right)\right\|, r\right) \ ,$$

where $\rho()$ and r are the robust estimator and the deterministically reduced radius of confidence, both discussed in Section 4.2.

A deformation energy term is required to favor, within the linear space of deformations, the ones that do not stretch or compress the surface. As we saw in previous section, a linear combination of principal components can result in a mesh that expands or shrinks. To model surfaces

```
boolean detect_surface_on_frame(f)
{
    wbm = compute_wide-baseline_matches(f)
    lfm = compute_last_frame_matches(f)
    for (r = 1000; r>1; r = 0.5*r)
    {
        sm = compute_silhouette_matches(f)
        m = apply_rho(r, wbm + lfm + sm)
        optimize_with_LevenbergMarquardt(m)
    }
    if (size(m) > threshold)
        return success
    else
        return failure
}
```

Algorithm 4.1: Overview of the 3–D complete detection algorithm.

that do not stretch, a deformation energy term favors, within the linear space of deformations, the ones that preserve edge lengths:

$$\varepsilon_D(\theta_3) \quad = \quad \sum_{(i,j)\in\mathcal{E}} \left(\|\mathrm{s}_i(\theta_\alpha) - \mathrm{s}_j(\theta_\alpha)\| - L_{i,j}\right)^2 \ , \tag{4.21}$$

where $\mathcal{E}$ represents the set of all edges and $L_{i,j}$ is the initial edge length. As in the 2–D case, the final energy term is defined by:

$$\varepsilon(\theta) = \lambda_D \varepsilon_D(\theta) + \varepsilon_C(\theta) \ . \tag{4.22}$$

The non-linear nature of the deformation energy term is incompatible with the semi-implicit optimization algorithm we used in the 2–D case (Section 4.2.1). We therefore use the Levenberg-Marquardt algorithm to optimize $\varepsilon(\theta)$ [71]. As we have seen in Section 4.2.3, mismatches elimination and optimization are alternated, with a radius of confidence r reduced at each iteration. Algorithm 4.1 summarizes the process.

The Levenberg-Marquardt optimization of ε_D is, compared to the 2–D case, more sensitive to initialization issues, because the problem is more ambiguous. Fortunately, wide-baseline matching eases convergence, even when starting with a very fuzzy initialization. We used several approaches. One of them is to assume a standard initial pose. As soon as the surface gets close enough to it, detection is successful and, since optimization starts from the previous frame configuration, the user can bring the target object far from its initial pose. Another strategy is to compute an initial guess using the P3P algorithm, assuming a rigid surface [39, 86]. If the deformation is not too large, the result is close enough to let the optimization converge to the global minimum. The last strategy, very recently developed by Salzmann et al, consist in a closed form computation of the pose and shape [91].

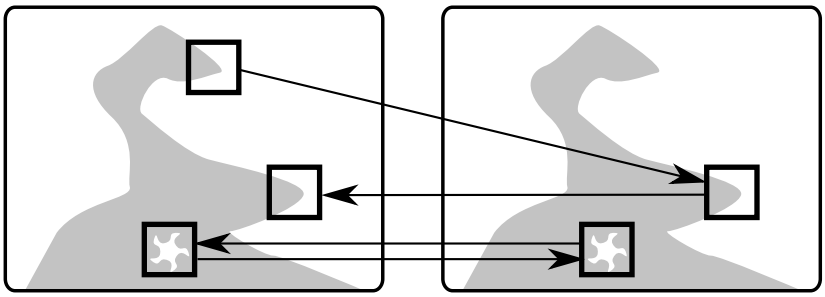


Figure 4.17.: Bidirectional feature matching. When matching two consecutive frames, matches are established back and forth, and only agreeing matches are kept.

Figure 4.18.: Frame-to-frame correspondences, as detailed in Section 4.3.4. Only inliers kept within the final radius of confidence are shown.

4.3.4. Temporal Consistency

Up to now, feature points are the only exploited image information. However, features are sparse. Despite the shape parameterization of previous section, recovering the 3–D pose of a monocular camera, together with the surface shape, is sometimes ambiguous: Several 3–D poses and shapes could explain the observed image. In that case, if detection is made on a video sequence without any temporal consistency, the system will hesitate between possible solutions. A very small change in image can cause the optimization to jump to another solution. The resulting camera pose estimation thus presents jitter. Therefore, enforcing temporal consistency is required. One possibility is to measure motion flow.

On the one hand, wide-baseline correspondences are sparse but provide very strong clues and are not subject to drift. On the other hand, frame-to-frame matching of features estimates motion flow, is easier to compute, denser, and induces drift. Both techniques are complementary and their combining, when detecting an object in a video sequence rather than in a single image, leads to a stable and jitter free tracking. In ambiguous cases, introducing frame-to-frame correspondences in the set of correspondences $\mathcal{C}$ forces the optimizer to stick to a particular solution, rather than jumping from one to the other at each frame. Of course, there is no guarantee that

the chosen solution is the correct one. Fortunately, tracking is stable in the sense that even if it drifts slightly, the wide-baseline correspondences will eventually bring the tracking back to the correct position.

Frame-to-frame correspondences are obtained by detecting features on two consecutive frames, and by comparing features neighborhood. The matching process is very similar to the one described in [122], except that epipolar geometry is meaningless in the presence of a non-rigid surface. As illustrated by Figure 4.17, features are matched back and forth: Frame $t-1$ is compared to frame t, and then frame t with $t-1$. Incompatible matches are eliminated. Fig 4.18 depicts typical correspondences obtained when matching two consecutive frames.

Once a correspondence is established, its location c_{t-1} on the last frame is backprojected on the surface, to obtain the 3–D coordinate p_{t-1}. Obviously, the projection of p_{t-1} on frame t should lie near c_t. This constraint can be enforced by the correspondence energy $\varepsilon_{\mathcal{C}}$ of Equation 4.8, by inserting all frame-to-frame correspondences into the set $\mathcal{C}$ of wide-baseline correspondences. Our robust estimation scheme takes care of eliminating outliers.

4.3.5. Exploiting Silhouettes

When projecting a 3–D surface on a planar image, the boundary between the background and the surface forms a silhouette on the image. Back projecting the silhouette on the 3–D surface yield a curve separating visible and hidden areas. The silhouette of the real object often correspond to a change in appearance on the image. Therefore, aligning the projection of the 3–D occluding contour with image transitions can improve 3–D shape recovery. In our framework, this process fits well the optimization, by summing an additional energy term in the objective function.

The approach works as follow. The first step consists in determining the 3–D occluding contour, based on the current 3–D shape and camera pose. This could be achieved accurately with implicit meshes [52], or more naively by simply computing the boundary between visible and hidden facets.

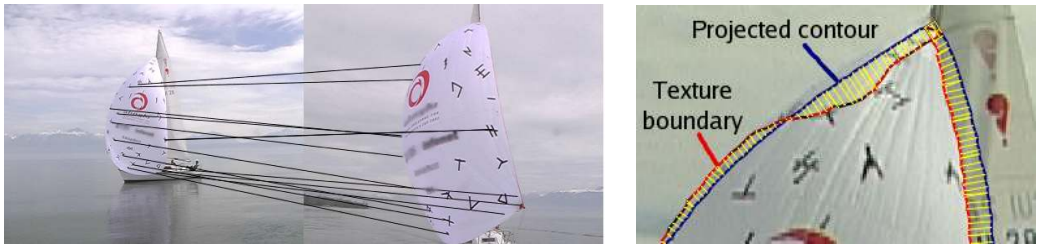
The 3–D occluding contour is then projected on the image, yielding a 2D curve which is regularly sampled. Texture changes are then searched from each sample, perpendicularly to the curve. One way of performing this search, proposed in [99], use a statistical model of texture features. A simpler and faster approach just searches for intensity transitions. Figure 4.19(c) shows an example of such silhouette scans.

An energy term ε_S is taken to be the sum of square Euclidean distances between the edge projections and their corresponding image transition. The robust estimator ρ of Equation 4.9 eliminates outliers. As explained in [112], keeping multiple hypotheses can also improve results.

For speed considerations, the edge matching is not done each time the minimizer updates the state vector. Instead, it is done each time the radius of confidence r is decreased.

4.3.6. Results

We tested our deformation measurement method in two contexts: Video sequences and still pictures. On the one hand, the goal of working with a video sequence is to analyze the behavior of a surface in time, while it is deforming. On the other hand, working with just a few pictures allows to have a snapshot of a surface shape at the time the picture was taken. From a computation point



(a) (b) (c)

Figure 4.19.: Image data. (a) An image from an input sequence. (b) One of 15 images used to build a textured 3–D model of the spinnaker. For our experiments, we added black scotch tape on the otherwise white parts of the sail to help our wide-baseline algorithm to find correspondences between model and input images such as those depicted by the black lines. (c) Contours detected as texture boundaries. Even though the boundary is not correct everywhere, thanks to the model and robust estimation we still recover the correct shape.

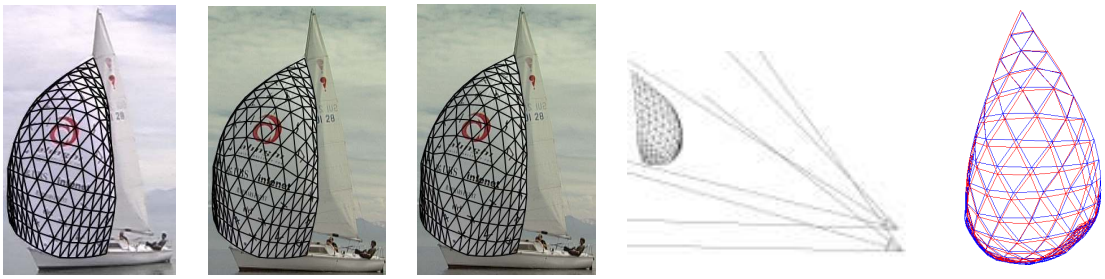
of view, both problem are very similar, except that frame-to-frame matches are unavailable in still images. From an application point of view, there is a trade-off between spacial and temporal resolution. We also tested the suitability of our technique to augmented reality by implementing a toy application in which the user deforms a paper augmented with virtual poles producing electric arcs. This application is presented in Chapter 7.

Deformation Recovery from Video Sequences Here we demonstrate the capabilities of our system using the four very different kinds of objects depicted by Figures 4.20, 4.21 and 4.22. They cover a wide range of physical properties and the images have been acquired using ordinary camcorders. In all four cases, we first created a 3–D textured model offline using one or more static images acquired independently of the videos.

We then optimized the criterion of Equation 4.22 using model-to-image correspondences, correspondences with the previous image, and optionally silhouette information. As shown in the figures of this section, the resulting shapes are accurate enough for correct reprojection.

We represent both the sheet of paper and the elastic surface, which was cut out of an inflatable balloon, of Figure 4.21 and 4.22 as a 30×20 rectangular grid. We model the spinnaker using a 153-vertex triangle. The T-shirt mesh is made of 2 sleeves that are 45-vertex equilateral triangle attached to a 9×25 rectangular grid. We used 45 PCA modes to track the sheet of paper, 10 for the balloon, 40 for the spinnaker and 50 for the T-Shirt.

The rest shape of the spinnaker is not planar and Voiles Phi, a Swiss sailmaking company, gave us the CAD model that was used to design it, thus allowing us to fit a triangular mesh to it. This gave us the initial shape from which we computed the deformation modes. To create the textured model needed for automated run-time operation, we developed a software tool that allows us to manually supply a few image-to-model correspondences in images such as those of Figure 4.23, which do *not* belong to the test video. By feeding these correspondences along with automatically detected silhouettes into the optimization framework of Section 4.3.3, we recover



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

Figure 4.20.: Tracking a spinnaker with either one or two cameras. (a,b) Two synchronized images from independently moving cameras, with recovered spinnaker reprojection. (c) Tracking using only one camera. Note that once reprojected on the images, the results are almost indistinguishable. (d) 3–D results with two cameras. Both camera positions are also retrieved. (e) Superposed 3–D shapes retrieved using either one (red) or two (blue) cameras. Note that both shapes are very similar, which indicates that the deformation model provides a good approximation when data are missing.

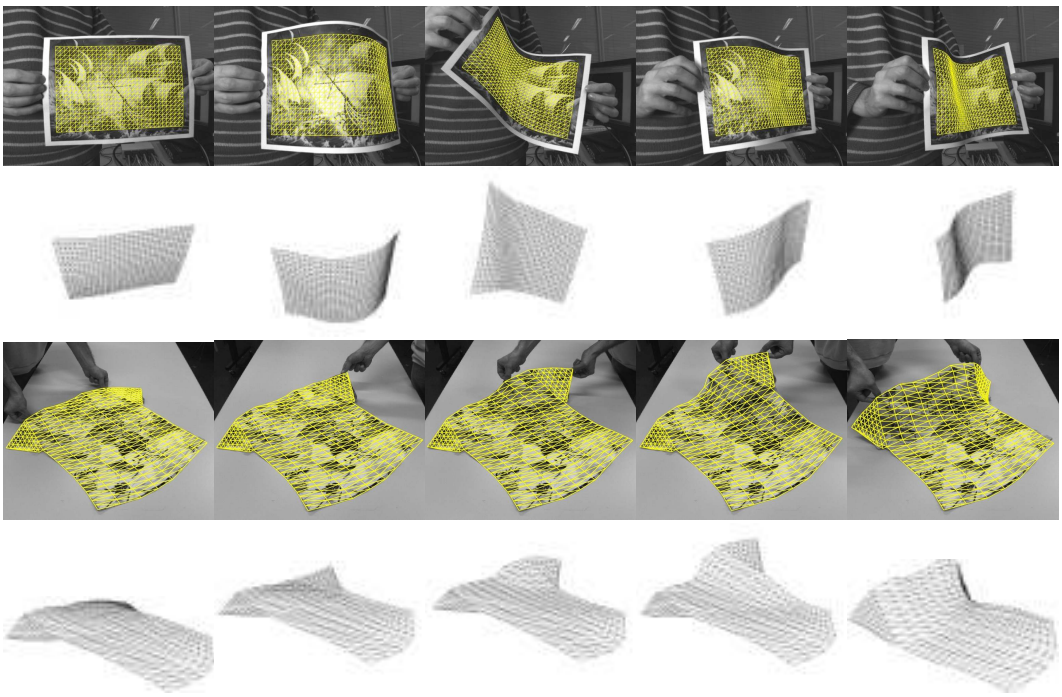


Figure 4.21.: Tracking a deforming sheet of paper and T-shirt. In both cases, we show the deformed 3–D mesh overlaid on the original images in the top row and then seen from a different viewpoint in the bottom row.

the spinnaker's shape into the images and, thus, a textured model. In the specific case depicted by Figure 4.23, we only supplied 10 correspondences per image, which did not take long to do. In short, our deformation modes not only lead to robust and automated run-time operation but can also be used to limit the required amount of manual intervention during model building.

These experiments display several strengths of our method. First, as shown in the videos corresponding to the deforming sheet of paper of Figure 4.21 and spinnaker of Figure 4.20, our system is robust enough to process sequences of more than 1500 frames acquired both indoors and outdoors without getting lost or drifting. When the image data is too weak, 3–D shape recovery becomes temporarily less accurate but the system soon recovers.

Deformation Recovery from Still pictures Our system also works with still pictures, such as the one of Figure 4.24. We developed a graphical user interface to quickly click correspondences between a 3–D CAD model and pictures, allowing quick deformation measure and shape comparison, as illustrated by Figure 4.25.

Detection precision increases with the number of simultaneous views used. To estimate our system's accuracy, we used a finite element simulation to generate a possible sail shape. We then generated a set of synthetic pictures on which we applied our method. To compare shapes, we first reduce them to a restricted set of parameters: Draft and camber at curves 25% and 50% of the spinnaker's height, as depicted by Figure 4.26. Sailmakers are used to deal with this kind of measures. We can then compare draft and camber measurement with our ground truth and use it as an estimation of the system's accuracy. The comparison is visible on Figure 4.27. Unsurprisingly, the accuracy improves when using multiple views. Results obtain from a single image are noisy but still meaningful.

4.4. Conclusion

This chapter presented a method making fast and robust non-rigid surface detection possible. Based on wide-baseline feature matching, our robust estimation scheme is able to simultaneously eliminate outliers and to compute deformation parameters. Since our approach requires a geometric model, we combined it with a 2–D regular hexagonal mesh. It allowed us to conveniently model non-rigid surfaces and required little computation costs. However, since its vertices lie in the image plane, no depth or orientation is computed, making impossible augmentation with 3–D objects. Therefore, we also tested a 3–D geometric model that both handles camera pose and a linear deformation basis. It allowed 3–D augmented reality on deforming surfaces. However, AR does not reduce to geometric registration and the next chapter proposes solutions to improve virtual object rendering by appropriately handling illumination.

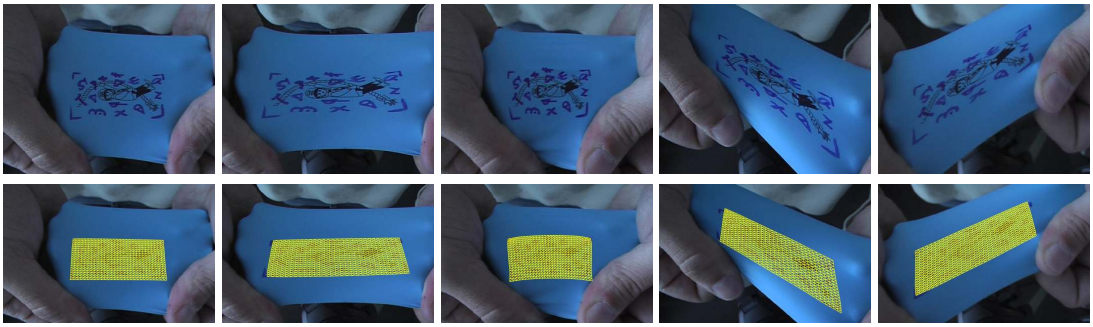


Figure 4.22.: Tracking an extensible surface undergoing anisotropic deformations. In the top row, we show the original images and, in the bottom row, we overlay the recovered 3–D grid that stretches appropriately.

Figure 4.23.: 3–D model of the spinnaker overlaid on the three images used to compute its reference texture. In each image, we specified 10 correspondences with a CAD model of the spinnaker and used them, along with automatically detected silhouettes, to deform it. The texture is then exploited to measure the shape in the remaining of the video sequence.

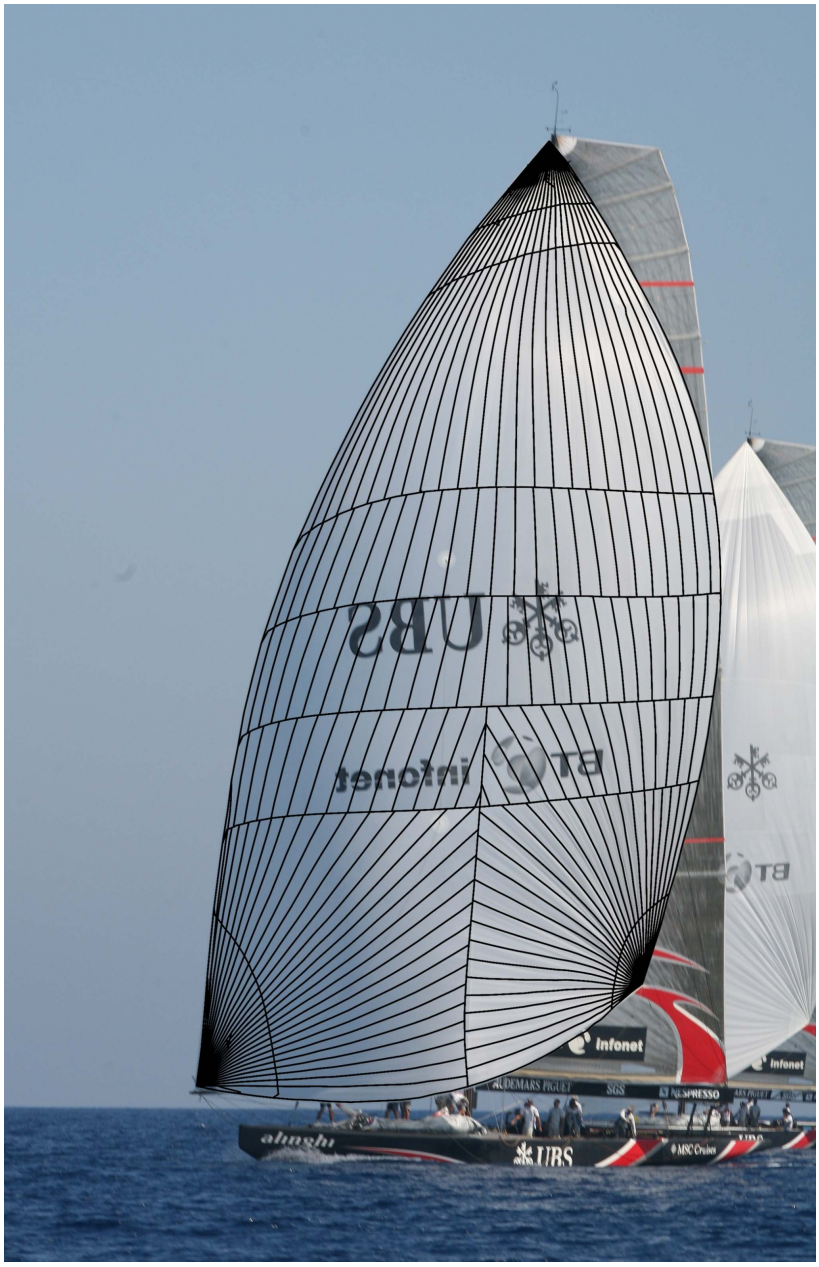


Figure 4.24.: Measuring the shape of a spinnaker. The black lines show the deformed and reprojected design shape.

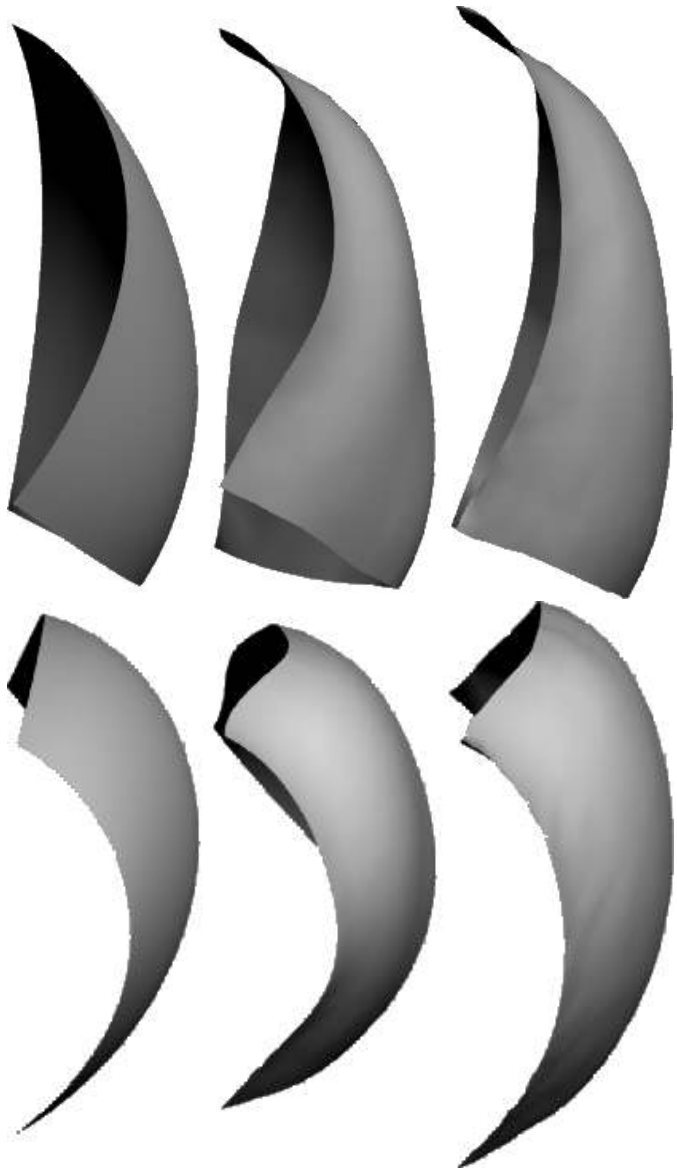


Figure 4.25.: Comparing three shapes of the same spinnaker. The design shape is on the left. Our system computed the shape of the two others, one of which is visible on Figure 4.24. They are manufactured with different materials.

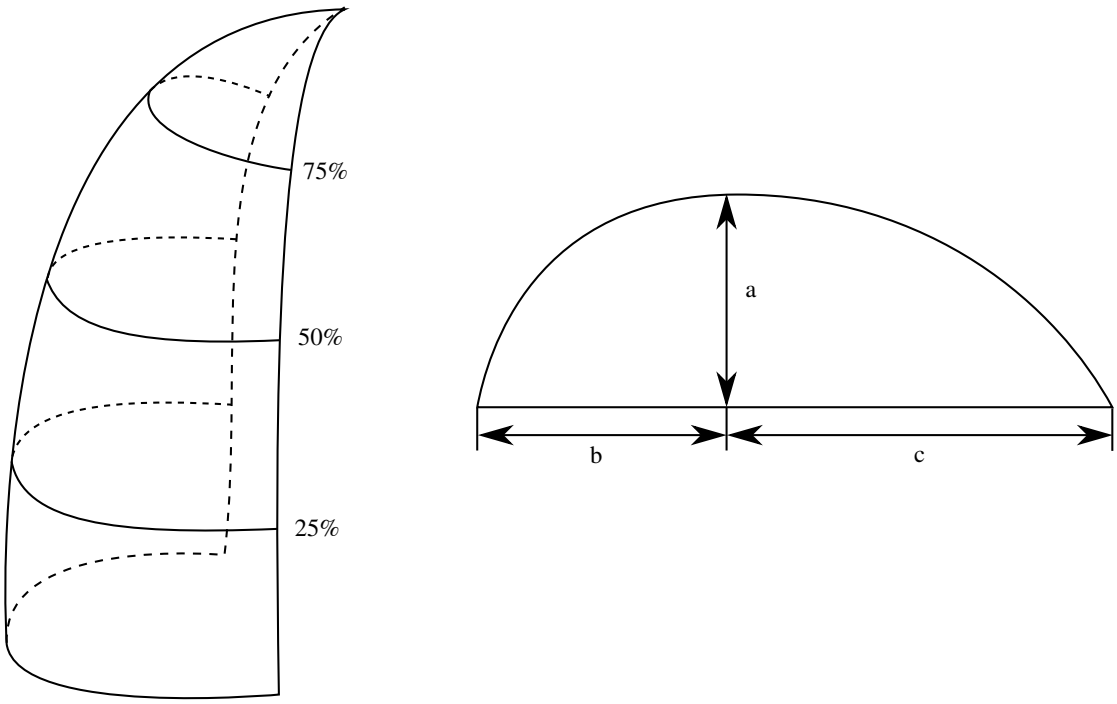


Figure 4.26.: To evaluate the shape of a spinnaker with only a few measures, sailmakers define curves at different height, such as the ones depicted at 25%, 50% and 75%. From these curves, camber is defined as $\frac{a}{b+c}$ and draft as $\frac{b}{b+c}$.

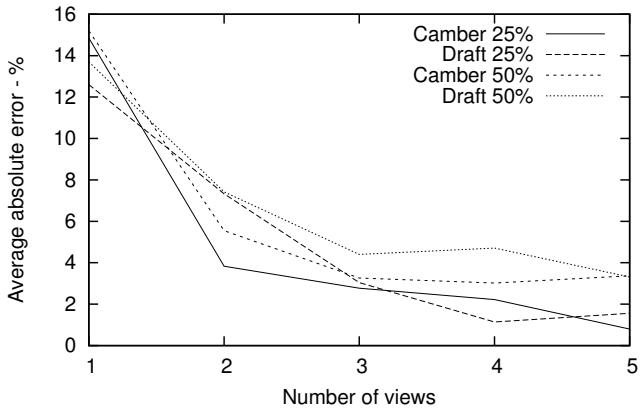


Figure 4.27.: Average absolute error of draft and camber measured from a varying number of synthetic images. As expected, the error decreases quickly when using multiple views. Monocular measurement, although noisy, is still meaningful.

5. Handling Illumination

Convincingly adding virtual objects into a scene not only requires proper geometric registration, which is what the techniques of Chapter 4 provide, but also photometric modeling so that they can be relighted correctly and properly blend into the environment. We propose to differentiate two cases: Retexturing, which augments a surface by altering its appearance while preserving its geometry, and 3–D augmented reality for which virtual geometry is added. Both cases require different approaches. In the retexturing case, since the geometry of the scene is preserved, the lighting information required to augment a particular surface point is present on the observed image. Therefore, the process first evaluate the changes in pixel intensity between the reference image and the input image and reproduce them on the virtual texture. The information about where the light is coming from is not necessary, as opposed to the case of added geometry. When rendering a virtual facet without a real counterpart, the system needs a representation of the real illumination environment for realistic shading. It also requires knowledge of the target object 3–D pose and orientation, as well as geometric calibration: The intrinsic and extrinsic parameters of the camera.

Most existing techniques perform geometric and photometric calibration as independent steps, which is incompatible with our framework of Section 2.4. By contrast, our approach relies on the very same set of images to perform both kinds of calibration, the geometric part of which is relatively standard and detailed in Appendix B. For each camera, as long as the lighting and camera settings do not change, the pixel intensities within the calibration pattern depend only on its normal. In other words, each image in which the pattern is detected provides a number of samples corresponding to one individual surface orientation. Because we can easily and automatically collect many such samples, we can simultaneously recover the gain and bias of each camera and create an environmental lighting model that can be used for relighting purposes.

In the rest of the chapter, we review the related work, present the retexturing case and finally the photometric calibration required for adding virtual geometry.

5.1. Related Work

A pioneering approach to realistic shading for AR relies on geographic location and date to compute the sun position [78] and to insert virtual buildings in a picture of a real landscape. Real-world illumination can be captured using calibration objects such as reflective spheres. This can be done as a preprocessing step [23] if the illumination remains unchanged. It can also be done in real-time [55] using a calibration object built by adding a mirror ball to a 2–D square marker. While effective, this approach is much more difficult to deploy than ours since constructing such an object is much more involved than simply using a textured pattern. Similarly, [93] relies on omni-directional stereo cameras, which requires specialized hardware

instead of the ordinary cameras we use. The photometric calibration we get may not be as accurate as those obtained with such lighting probes but we will show that it is amply sufficient to synthesize realistic augmented images while being much more light-weight.

Another class of approaches [27, 41] focuses on interactive rendering of illumination changes caused by virtual objects in the real scene. However, this typically involves a 3–D scene model, which is cumbersome to acquire.

Our approach to creating an environmental lighting map is related to the inverse lighting framework of [72] that involves an object of constant albedo and known shape. If the object surface contains a sufficiently large number of differently oriented normals, it is possible to relate a lighting contribution to each direction of a discretized lighting sphere. Regularized deconvolution then allows us to estimate the light sources position. In our case, as the grid moves in front of the cameras, it samples the spaces of normals. Since we precisely compute those normals, we can directly exploit the observed changes in pixel intensities to model the illumination using a very similar approach. However, we require a textured surface rather than a constant albedo.

5.2. Illumination Model

In practice, if we wish to build a versatile system that can be demonstrated in uncontrolled environments, we cannot make strong assumptions about light sources that are present when acquiring the input video. There can be many and their respective intensities and spectral properties are unknown, which can result in complex shading, shadowing, and color effects. To avoid the latter, we work independently on the red, green, and blue bands of color images.

We consider a potentially infinite number of sources that we assume to be both directional and outside the capture volume. For our purposes, this provides a satisfactory approximation of extended light sources.

The irradiance e at surface point p and time t can therefore be written as

$$e_{\mathrm{p},t} = \sum_{l=0}^{L} o_{l,\mathrm{p},t} \max\left(\mathbf{n}_{\mathrm{p},t}\mathbf{d}_l, 0\right) \Omega_l \,, \tag{5.1}$$

where Ω_l represents the radiosity, or power, of source l, $\mathbf{d}_l$ its direction, and $o_{l,\mathrm{p},t}$ a binary variable that takes the value 1 if and only if light source l is visible from point p at time t. We assume a Lambertian surface and a linear response for the cameras and write the pixel intensity $I(c, \mathrm{p}, t)$ of point p in the image acquired by camera c at time t as

$$I(c, \mathrm{p}, t) = g_c a_{\mathrm{p}} e_{\mathrm{p},t} + b_c \,, \tag{5.2}$$

where a_{p} is the surface albedo at point p, g_c the camera gain, and b_c its bias.

We propose two ways of exploiting this illumination model for augmented reality. We will first show how to create an augmented image by modifying the surface albedo a_{p}, while preserving the surface normal and position. We then explain how to compute relative gains and biases of several cameras and how to compute the irradiance e_t as a function of the surface normal. It results in a light map that can shade virtual objects.

Equation 5.2 assumes that all pixels have the same gain and bias, which is wrong in case of vignetting: Poor lenses tend to make image corners darker than its center. This issue is not a problem in the retexturing case, since the irradiance can be evaluated for each pixel and can therefore account for vignetting. However, when inserting new geometry in the scene, the evaluation of the irradiance is done over many frames and anywhere on the input images. A strong vignetting might cause some artifacts. In practice, none of our experiments suffered from this issue.

5.3. Realistic Retexturing

In Chapter 4, we have shown that we could compute fast and accurately the 2–D deformation of a surface. In an Augmented Reality application such as the one depicted by Figure 5.1, this is what is needed to modify in real-time the appearance of that surface. However, to achieve a convincing illusion, it is important not only to model geometric deformations but also lighting changes. To this end, we have developed a dynamic approach to estimating the amount of light that reaches individual image pixels by comparing their colors to those of the model image. This lets us either erase patterns from the original images and replace them by blank but correctly shaded areas, which we think of as *Diminished Reality*, or to replace them by virtual ones that convincingly blend-in because they are properly lighted.

The Lambertian Case The augmented surface is assumed Lambertian. It is easy to control the acquisition of its reference image. With no loss of generality, we can therefore assume that it has been acquired when the surface was both undeformed and lighted uniformly, which means that every surface point receives the same amount of light in the color band we are working with.

Under this assumption, let $\mathbf{m}_{\mathrm{p}}$ and $\mathbf{u}_{\mathrm{p}}$ be the pixel intensities showing the same surface point p in the reference and input image respectively, and let a_{p} be the corresponding surface albedo. By assuming a camera gain $g_c = 1$ and bias $b_c = 0$, we obtain from Equation 5.2:

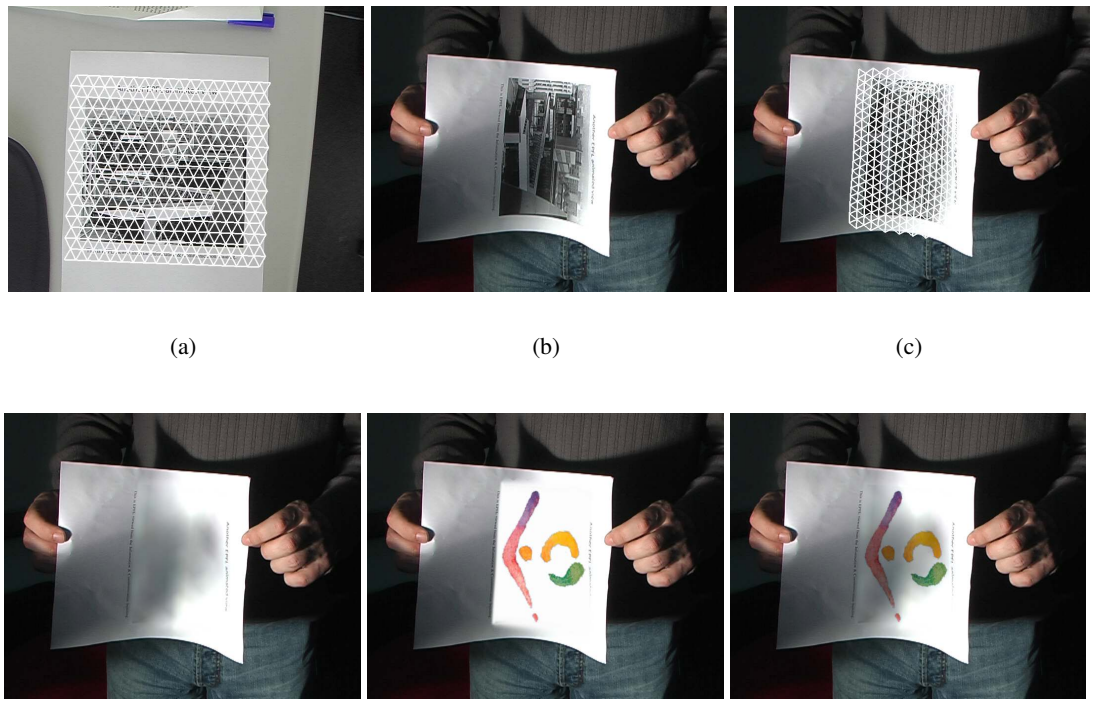
$$\mathbf{m}_{\mathrm{p}} = e_r a_{\mathrm{p}} \quad , \tag{5.3}$$

$$\mathbf{u}_{\mathrm{p}} = e_{i,p} a_{\mathrm{p}} \quad , \tag{5.4}$$

where $e_{i,p}$ is the total irradiance reaching surface point p, and e_r the total irradiance in the reference image assumed to be the same at all surface points. In general, the values of $\mathbf{m}_{\mathrm{p}}$ and $\mathbf{u}_{\mathrm{p}}$ are different due to changes in both normal orientations and lighting conditions. However, the geometric registration we have established between the two images tells us that they correspond to the same physical point, which we exploit as follows.

Let us consider a white surface area with albedo a_{w} at location w on the surface. If the target surface has no white part, it is always possible to put a white object next to it while taking the reference image. We can measure on the reference image the pixel intensity $\mathbf{m}_{\mathrm{w}}$ where this white location w is projected and write

$$\mathbf{m}_{\mathrm{w}} = e_r a_{\mathrm{w}} \quad , \tag{5.5}$$



(a) (b) (c)

(d) (e) (f)

Figure 5.1.: (a) The reference image of the target surface with the model mesh overlaid. (b) An input image. (c) The mesh is correctly deformed and registered to the input image. (d) The original pattern has been erased and replaced by a blank but correctly shaded image. (e) A virtual pattern replaces the original one. It is correctly deformed but not yet relighted. (f) The virtual pattern is deformed and relighted.

where e_r is the irradiance of Equation 5.3. Using this white normalization $\mathbf{m}_\mathrm{w}$, we can compute a new image, looking similar to the input one, except that the surface albedo is changed to a_w. In the input image, if there were no texture, the corresponding image intensity should be

$$\mathbf{s}_\mathrm{p} = e_{i,p} a_\mathrm{w} = a_\mathrm{w} e_r \frac{\mathbf{u}_\mathrm{p}}{\mathbf{m}_\mathrm{p}} = \mathbf{m}_\mathrm{w} \frac{\mathbf{u}_\mathrm{p}}{\mathbf{m}_\mathrm{p}} \quad . \tag{5.6}$$

Note that $\mathbf{s}_\mathrm{p}$ is expressed exclusively in terms of image intensities, which are readily available, as opposed to albedoes or surface normals that are not.

Replacing the intensities $\mathbf{u}_\mathrm{p}$ of all the pixels on the object surface by $\mathbf{s}_\mathrm{p}$ yields images such as the one of Figure 5.1(d) where the original texture has been replaced by a blank but correctly shaded surface. To draw a shaded new texture, as in Figure 5.1(f), we simply multiply texture values with their corresponding white $\mathbf{s}_\mathrm{p}$.

Note that, because we perform the computation locally, it remains valid no matter how many sources there are and what their specific characteristics may be. The only thing that has to be true is that the contribution of the individual light sources to the pixel intensity are all modulated by the same diffuse albedo and do not depend on the viewpoint.

In practice, we compute the lighting factor only at mesh vertices, averaging pixels values of both model and input images over an hexagonal area surrounding it. The resulting $\mathbf{s}_\mathrm{p}$ values are then interpolated over triangles by the GPU through a few OpenGL calls.

In some cases, $\mathbf{s}_\mathrm{p}$ is difficult to estimate reliably on large single-colored areas. In the example of Figure 5.2(a), recovering the $\mathbf{s}_\mathrm{p}$ blue component over the red area is hard because sensor inaccuracy on remaining blue light is amplified by a big factor. However, the visual impression given by Figure 5.2(b) is still that the original painting has been erased and replaced.

Specularities and Saturation The assumptions used to derive Equation 5.6 are clearly violated for specular materials. However, as illustrated by Figure 5.3, this does not have severe consequences even in the presence of strong specularities and the illusion remains convincing.

This is because, when there is a specularity, the image intensity increases and the $\frac{\mathbf{u}_\mathrm{p}}{\mathbf{m}_\mathrm{p}}$ ratio of Equation 5.6 becomes large. As a result, the $\mathbf{s}_\mathrm{p}$ intensity that is used to draw the synthetic patterns also increases, which is perceptually correct since it yields intensity maxima at specularities' locations. In other words, the absolute value of $\mathbf{s}_\mathrm{p}$ may not be correct but its magnitude relative to its neighbors remains consistent with the presence of a specularity. And since the human eye is much more sensitive to relative values than to absolute ones, this suffices.

In practice, specular peaks often saturate the camera sensor, thus making the estimation of $\mathbf{u}_\mathrm{p}$ unreliable. We detect such cases by simple thresholding and we handle saturation by setting $\mathbf{s}_\mathrm{p}$ to its maximal possible value. Since color computation is applied independently on the red, green and blue channels, one channel can saturate while the other do not. As a result, not only specular peaks but also saturated areas in the input image are correctly transcribed into the synthetic ones.

5.4. Radiance Map for 3–D Augmentation

In this section, we extend retexturing to make possible the realistic shading of virtual geometry. To do so, we need the 3–D detection of a flat pattern and the calibration of one or more cameras,

(a) (b)

Figure 5.2.: (a) original image. (b) The ISMAR logo replaces the shirt print. Recovering white is hard in this image since the model has large single-colored areas, making light evaluation difficult.

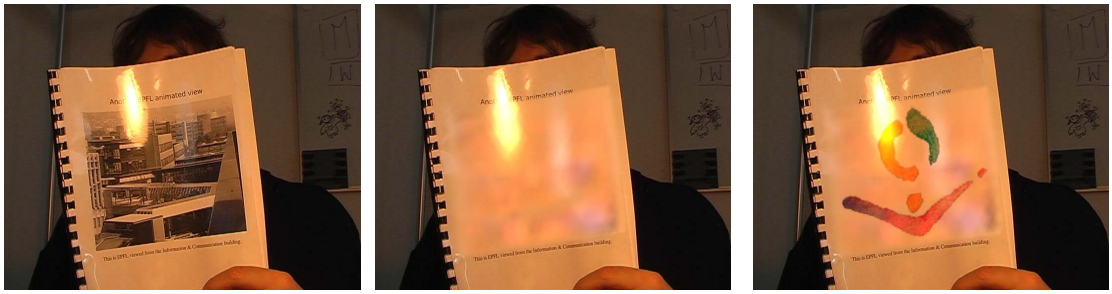


Figure 5.3.: Handling specularities. (a) Input image with strong specularities. The main one is produced by a lamp, while the two smaller ones can be attributed to light coming through window. To produce this result, the paper has been covered by a transparent plastic sheet. (b) The picture has been erased from the surface but the specularities still appear to be at the right places. (c) The ISMAR logo has been inserted.

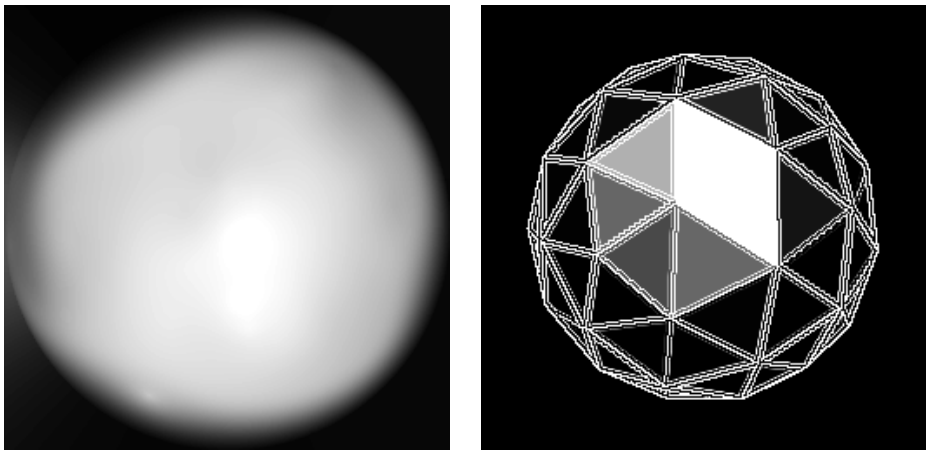
both detailed in Appendix B. We present two complementary approaches to 3–D shading. The first one involves solving a linear system of equations derived from the calibration sequence to express the illumination as a function of surface normals. Once the gains and biases have been estimated, this model can be incrementally updated to reflect lighting changes. The algorithm can therefore be embedded into a real-time application that handles non-constant lighting. However, it is not designed to synthesize either shadows or specular reflections. We use therefore a second approach based on deconvolution. It explicitly computes a light distribution that could have produced the observed pixel intensities. It is more computationally intensive and makes no provisions for time-varying lighting but allows added virtual objects to cast shadows and produce realistic specularities.

Assumptions Recall the illumination model presented in Section 5.2. It considers a potentially infinite number of sources that we assume to be both directional and outside the capture volume. Moreover, by printing the calibration pattern on matte paper, we ensure that it reflects light equally in all directions. As a result, the difference in intensity values between images taken at the same time by two different cameras are due to shutter speed or aperture, but not to camera pose. If we further assume that the same sources are visible from every point of the calibrated volume, the amount of light reflected by a point on the calibration pattern depends only on the surface normal $\mathbf{n}_t$ at time t. In order to simplify reasoning, we assume here a flat augmented pattern. Therefore, the observed normal $\mathbf{n}_t$ is constant over the surface and depends only on time, as the whole plane rotates.

To be robust to small localization errors, we do not consider individual points, but small patches π that average the local property around points on the calibration pattern. Applying these assumptions to Equation 5.1, the irradiance e at surface patch π and time t can therefore be simplified to:

$$e_{\pi,t} = \sum_{l=0}^{L} \max\left(\mathbf{n}_t \mathbf{d}_l, 0\right) \Omega_l \,, \tag{5.7}$$

where Ω_l represents the radiosity, or power, of source l and $\mathbf{d}_l$ its direction. Recall that the pixel



(a) (b)

Figure 5.4.: Illumination models. (a) An interpolated light map computed using the on-line technique of Section 5.4.1. (b) A dome of light sources whose individual powers are estimated using the technique of Section 5.4.2. In this picture, the intensity of a triangle represents the power of an individual light source.

intensity $I(c, \pi, t)$ of patch π in the image acquired by camera c at time t is

$$I(c, \pi, t) = g_c a_\pi e_t + b_c \,, \tag{5.8}$$

where a_π is the average surface albedo over π, g_c the camera gain, and b_c its bias.

Our goal is to compute the gain and bias of all cameras, together with a collection of irradiances e_t for all observed surface poses. In practice these quantities can only be known up to a scale factor. We use for a_π the mean intensity over π in an image acquired under uniform diffuse lighting such as the ones of Figure B.2.

5.4.1. On-line Lighting Calibration

Instead of explicitly evaluating the Ω radiances of Equation 5.7, we directly compute the $e_{\pi,t}$ irradiances as a function of the orientation of the $\mathbf{n}$ normals. More specifically, we simultaneously compute the gains, biases, and $e_{\pi,t}$ irradiance values for the normals we have observed. We then interpolate this set to estimate the unobserved values.

This yields a *light map* $M(\mathbf{n})$ such as the one depicted by Figure 5.4(a). When synthesizing an augmented image $\mathbf{s}$ with a virtual surface whose normal is $\mathbf{n}_\mathrm{v}$, the augmented pixel value is set to $\mathbf{s}_\mathrm{v} = g_c a_\mathrm{v} M_t(\mathbf{n}_\mathrm{v}) + b_c$, where a_v is the albedo of the virtual surface at point v. This lightmap rendering is easily done by the GPU and requires only a short OpenGL Shading Language (GLSL) program.

5.4.1.1. Linear Estimation of the Light Map

The geometric calibration process provides many surface normals $\mathbf{n}_t$ and pixel values $I(c, \pi, t)$. To solve for the unknown gains, biases, and radiances, we linearize the problem by replacing

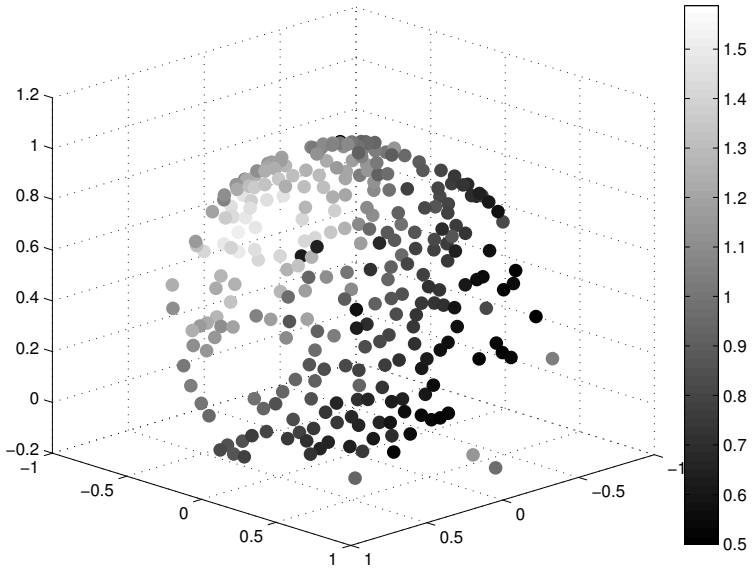


Figure 5.5.: The irradiance factors e_t obtained by solving the linear system of Equation 5.9. Interpolating these values yield a light map similar to the one of Figure 5.4(a).

some variables in Equation 5.8. Let

$$\begin{aligned} g'_c &= \frac{1}{g_c} \ , \\ b'_c &= \frac{b_c}{g_c} \ . \end{aligned}$$

For each patch in each detected frame, Equation 5.8 can be rewritten as

$$-I(c,\pi,t)g'_c + a_\pi e_t + b'_c = [-I(c,\pi,t)\ 1\ a_\pi] \begin{bmatrix} g'_c \\ b'_c \\ e_t \end{bmatrix} = 0 \ . \tag{5.9}$$

Putting all these equations together yields a large but sparse linear system and we find our solution as the eigenvector associated to the smallest eigenvalue. In case of color images, one system is solved for each color band. The scale of the result is arbitrary, and we choose to set the gain of the first camera to 1. Typically 50 to 300 irradiances solved simultaneously is enough to provide gain and bias relating the multiple cameras. Figure 5.5 depicts the solution of such a system. Figures 5.7 and 5.8 show the same teapot rendered with different camera biases. When comparing the intensity on the table, the difference in camera aperture is clearly visible and well reproduced on the virtual teapot.

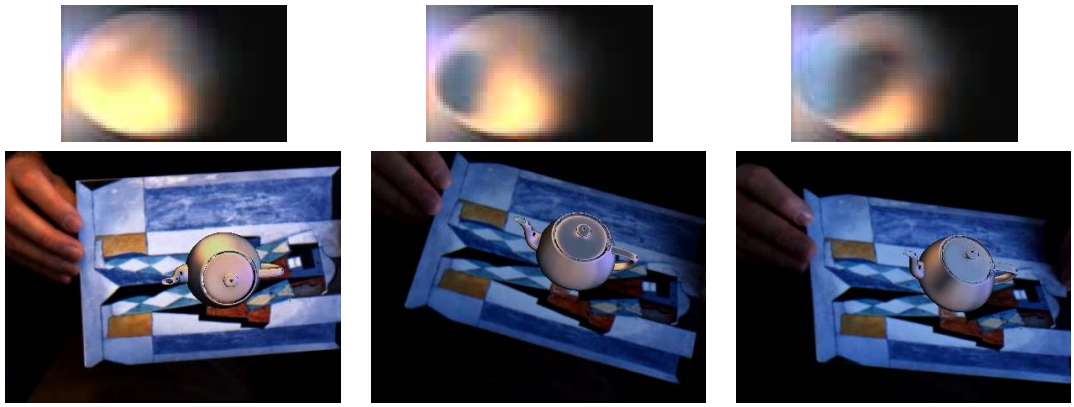


Figure 5.6.: Rendering a virtual teapot under real and dynamic lighting. The top row shows the updated light map. On the left image, a lamp lights up the card and the virtual teapot. The lamp is then switched off, and the lightmap progressively reflects the new illumination conditions.

5.4.1.2. Incrementally Updating the Light Map

The above computation assumes that the lighting does not change during calibration. However, once the gains and biases are known, computing a new irradiance e_{t+1} from a new observed pixel $\mathbf{u}_{\pi,t+1}$ is trivial. Thus, if the illumination changes, new frames can update the light map at the same time it shades a virtual scene. Each frame can sample only one normal at a time. Thus, if the light changes suddenly, it is not possible to update the whole irradiance map at once. Instead, we locally update the irradiance around the measured normal and keep the old values for other normals.

Let $M_t(\mathbf{n})$ be the light map at time t. To update its value for a surface of normal $\mathbf{n}$ with the recently computed sample e_{t+1} corresponding to the observed real surface orientation $\mathbf{n}_{t+1}$, we write

$$\forall \mathbf{n} : M_{t+1}(\mathbf{n}) = (1 - f(\mathbf{n}))\, M_t(\mathbf{n}) + f(\mathbf{n}) e_{t+1} \quad ,$$

where

$$f(\mathbf{n}) = \begin{cases} \exp\left(-\cos^{-1}\left(\mathbf{n} \cdot \mathbf{n}_{t+1}\right) \frac{1}{2\sigma^2}\right) & \text{if } \cos\left(\mathbf{n} \cdot \mathbf{n}_{t+1}\right) > 0 \\ 0 & \text{otherwise} \end{cases}$$

and σ^2 a constant blurring factor. In practice, our system initializes the light map using the solution of the linear system of equations and then updates it for each new frame. Figure 5.6 illustrates such a dynamic light map.

5.4.2. Offline Estimation of Light Distribution for Rendering Specular Effects and Casting Shadows

While the previous method yields satisfying results for fast on-line pre-visualization, its accuracy suffers from two numerical problems. First, it does not minimize a physical error since

the problem is intrinsically non-linear. Second, it solves simultaneously for camera-related parameters, the gains and the biases, and lighting-related parameters, the irradiances. As is the case for geometric camera calibration, performing non-linear optimization and dissociating the estimation of the internal parameters, here the gains and biases, from that of the external ones, here the irradiances, yields more reliable results.

In [83], Lagger et al. presented an offline approach to doing so. It is computationally more expensive than the on-line technique of Section 5.4.1 but produces a more sophisticated illumination model that allows for specular highlights, cast shadows, and changing the materials of the virtual objects. The lighting environment is modeled as a regularly sampled dome of lights of varying power. The process begins by estimating the gain and bias of each camera in a way that is independent from lighting effects and normalizing the pixel intensities. It then applies a regularized deconvolution algorithm on the observed pixel intensities within the calibration object to assign to each individual light the power that best explains the observations. It yields a lighting sphere with smooth but well-defined clusters, such as the one of Figure 5.4(b). The knowledge of the source positions allows for realistically shaded virtual objects. Realistic cast shadows and specularities can be computed using a Phong shading model. As shown in the next section, the conjunction of tracking and lighting distribution estimation allows us to render convincing augmented images.

5.4.3. Results

We used a three camera set-up to produce the augmented images of Figure 5.7. In the first row, we use only the geometric calibration parameters to draw the virtual teapot at the right place and use a randomly selected point light source to relight it. Even though the teapot is correctly registered, the result is unconvincing because the shading patterns do not match those of the other real objects present in the scene. In the middle row, we use the output of the on-line photometric calibration procedure of Section 5.4.1 to relight the object. The result is much improved but highlights and shadows are still missing. As shown in the bottom row of the figure, using the output of the more sophisticated offline calibration procedure of Section 5.4.2 solves both problems. Both highlights and shadows now appear at the right places, thus significantly increasing the realism.

Figure 5.8 showcases the flexibility that our multi-camera system provides. In two of the three images, the calibration pattern is not visible in the view we are trying to augment and a monocular approach that relies on detecting it would fail. However, because it is seen by another camera and because the relative positions of the camera with respect to each other have been computed, we can nevertheless draw it at the right location, as evidenced by the fact that the real box occludes it correctly.

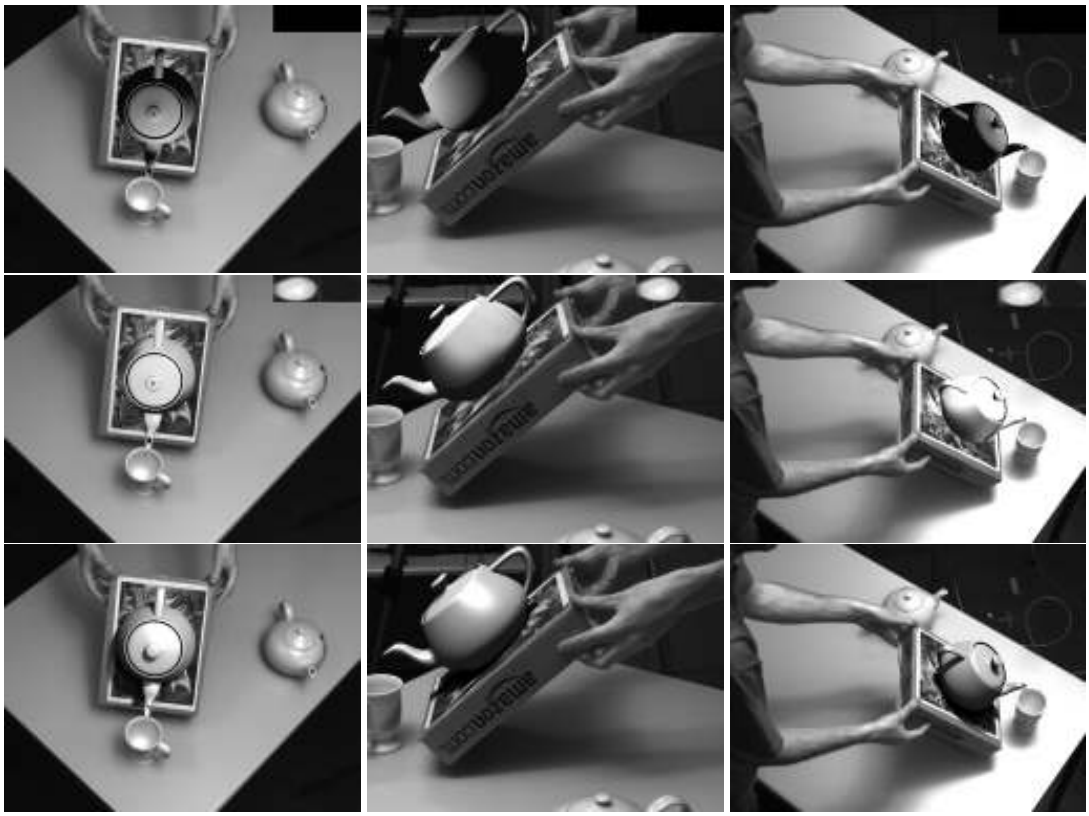


Figure 5.7.: Adding a virtual teapot. **First row**: We use only the geometric calibration data and relight the teapot using a random point light source. The shading patterns do not match those of the real objects. **Second row**: We use the on-line model of Section 5.4.1 to relight the teapot. The result is better but the highlights and shadows are still missing. **Third row**: Using the more sophisticated offline technique of Section 5.4.2 produces realistic highlights and shadows.

Figure 5.8.: Even when the calibration pattern is invisible in the view we are augmenting, the real object is accurately registered and correctly occludes the virtual teapot. This is possible because the calibration pattern is seen by another camera whose position and orientation are known.

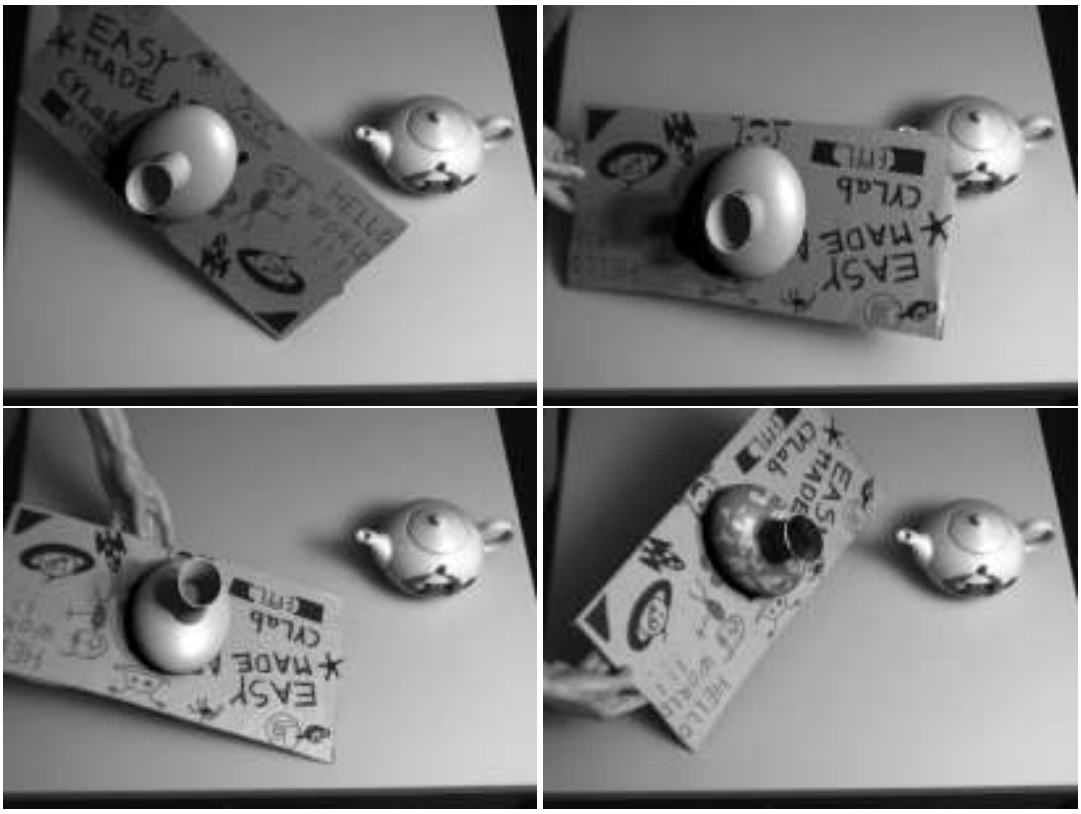


Figure 5.9.: Calibration and augmentation of a piece of cardboard. Note that the virtual vase casts shadows on the real object in all frames and that a specularity moves realistically on its surface. In the bottom right image, we used a texture map to change the albedo of the vase.

6. Handling Occlusions

In augmented reality, a real object, such as a hand, could occlude, be occluded by, or intersect a virtual augmentation. Dealing with all the three cases requires estimating the depth at which the occluder is seen. Several approaches have been proposed to address this issue, few of them meeting the constraints of our framework that forbids dedicated depth sensors. Recovering depth from monocular views is not a simple task. We will not address it here. We rather address a simpler problem: The case where a real object occludes a virtual one without intersecting it. As we will see in this chapter, this restriction allows us to segment occlusion from a monocular view, yielding correct retexturing. In uncommon pathological cases of 3–D augmentation, it may introduce some artifacts. Overall, it is a perfectly acceptable price to pay to reach our weak hardware requirement goal.

Our framework gives us a basis to address the occlusion issue, because it assumes that the geometry and appearance of the object to augment is known. Chapter 4 and appendix B.8 cover registration techniques with an observed picture. It is then possible to render a synthetic image of the expected object appearance, in the exact pose showed by the input image. Of course, the synthetic and actual images differ. The main causes are illumination changes and occlusions.

At that point, our initial depth estimation problem can be reduced to a segmentation problem. Segmenting the areas where the differences are due to occlusion delimits the pixels that must display reality rather than virtual objects. The task is simplified, but still far from trivial, because occluding objects often cast shadows on unoccluded areas, changing their appearance. The ambiguity between illumination effects and occlusions easily confuses naive algorithms.

The difficulty can be seen as a chicken-and-egg problem. To compute illumination, it is important to exclude occluded pixels. Otherwise, the illumination map gets corrupted. However, knowing the illumination is very useful to determine the visibility of pixels, because direct comparison of intensities become possible. We propose to address the issue with an Expectation-Maximization (E-M) algorithm that estimates the segmentation in turns with illumination and occluding object colors. It also relies on normalized cross-correlation to accelerate the algorithm convergence and to increase its robustness by preventing it from getting stuck in local maxima.

Occlusion detection by segmentation is closely related to background subtraction, a popular video surveillance technique that segments foreground objects from images taken by a static camera. We can therefore inspire from the abundant literature covering background subtraction. It also turns out that our method, applied to background subtraction, performs very well in the presence of sudden and drastic illumination changes.

6.1. Related Work

Registering existing 3–D models of occluding objects is one option explored in [16]. Accurate mutual occlusion between natural and artificial objects can be achieved using dense depth computation from a stereo image pair [96, 118]. A combination of semi-interactive outlining of occluding objects and structure from motion allows occluded AR and *diminished* reality [60].

The problem of occlusion in augmented reality can be considered as a background subtraction problem. By back-warping the image to augment into a reference frame, one can locate visible areas of the model to augment. This is an approach described in [42]. It presents an AR system that includes interactive pointing. A pointer, such as a finger, occludes the augmented marked plane. A stereo pair computes a depth map, and a background subtraction technique segments the finger from the plane. The authors mention that *shadows will be represented in the difference mask but are not part of the pointer itself.* They improve segmentation by merging depth information from the stereo pair. In contrast, we address a very similar segmentation problem without requiring a stereo pair. Their work proves the usefulness of our tools for augmented reality applications.

Many background subtraction techniques faced problems with illumination changes and shadows. A popular solution consists in updating on-line a statistical background model. A pixel from a new image is then classified as background if it fits the model. Wren et al. [119] represent the color of each pixel by a three-dimensional Gaussian, learned from color observation of consecutive frames. Since a single Gaussian is a poor approximation of the true probability density function, GMMs were proposed instead [37, 103]. These approaches have proved to be effective at handling gradual illumination changes and repetitive dynamic backgrounds. Many improvements have been published since, such as a recent method that dynamically selects the appropriate number of components for each pixel [124].

Introducing a GMM is not the only way to model a dynamic background. Elgammal et al. proposed to model both background and foreground pixel intensities by a nonparametric kernel density estimation [29]. In [100], Sheikh and Shah proposed to model the full background with a single distribution, instead of one distribution per pixel, and to include location into the model.

Shadows cast by moving objects cause illumination changes that follow them, thereby hindering the integration of shadowed pixels into the background model. This problem can be alleviated by explicitly detecting the shadows [85]. Most of them consider them as binary [85], with the notable exception of [102] that also considers penumbra by using the ratio between two images of a planar background. Our approach also relies on image ratios, but treats shadows as a particular *illumination effect*, a wider class that also include the possibility of switching lights on.

Another way to handle illumination changes is by using illumination invariant features, such as edges. Edge information alone is not sufficient, because some part of the background might be uniform. Thus, Jabri et al. presented an approach to detect people fusing color and edge information [53]. More recently, Heikkilä and Pietikäinen modeled the background using histograms of local binary patterns [49]. The bilayer segmentation of live video presented in [22] fuses color and motion clues in a probabilistic framework. In particular, they observe in a labeled training set the relation between the image features and their target segmentation. This

solution is not acceptable in our case, since shadows are mostly produced by occluding objects themselves, making difficult an offline learning of illumination changes.

6.2. Method

Our method can serve in two different contexts. For augmented reality applications, where an object is moving in the camera field and occlusions have to be segmented for realistic augmentation. For background subtraction, where both the scene and the camera are static.

Let us assume that we are given an unoccluded *model image* of an object or a background scene. Our goal is to segment the pixels of an *input image* in two parts, those that belong to the same object in both images and those that are occluded. If we are dealing with a moving object, we first need to register the input image, as covered in Chapter 4, and create an image that can be compared to the model image pixel wise. If we are dealing with a static scene and camera, that is, if we are performing standard background subtraction, registration is not necessary. It is the only difference between both contexts, and the rest of the method is common. In both cases, the intensity and color of individual pixels are affected mostly by illumination changes and the presence of occluding objects.

Changes due to illumination effects are highly correlated across large portions of the image and can therefore be represented by a low dimensional model that accounts for variations across the whole image. In this work, we achieve this by representing the ratio of intensities between the stored *background image* and an input image in all three channels as a Gaussian Mixture Model (GMM) that has very few components—2 in all the experiments shown in this chapter. This is in stark contrast with more traditional background subtraction methods [119, 37, 103, 124] that introduce a model for each pixel and do not explicitly account for the fact that inter-pixel variations are correlated.

Following standard practice [89], we model the pixel colors of occluding objects, such as a hand holding the augmented object, as a mixture of Gaussian and uniform distributions.

To fuse these clues, we model the whole image—background, foreground and shadows—with a single mixture of distributions. In our model, each pixel is drawn from one of five distributions: Two Gaussian kernels account for illumination effects, and two more Gaussians, completed by a uniform distribution, represent the foreground. A hidden variable assigns pixels to one of the five distributions (E-step) and then optimizes the distributions parameters (M-step).

Since illumination changes preserve texture whereas occluding objects radically change it, the correlation between image patches in the model and input images provides a hint as to whether pixels are occluded or not in the latter, especially where there is enough texture.

In order to lower the computational burden, we assume pixel independence. Since this abusive assumption entails the loss of the relation between a pixel and its neighbors, it makes it impossible to model texture. However, to circumvent this issue, we characterize each pixel of the input image by a five dimensional feature vector: The usual red, green, and blue values plus the normalized cross-correlation and texturedness values. Feature vectors are then assumed independent, allowing an efficient maximization of a global image likelihood, by optimizing the parameters of our mixture. In the remainder of this section, we introduce in more details the different components of our model.

6.2.1. Illumination Likelihood Model

First, we consider the visible model, which is responsible for all pixels that have a counterpart in the model image $\mathbf{m}$. If a pixel u_i of the input image $\mathbf{u}$ shows the occlusion free target object, the luminance measured by the camera depends on the light reaching the surface (the irradiance e_i) and on its albedo. Irradiance e_i is function of visible light sources and of the surface normal. Under the Lambertian assumption, we have seen in Section 5.2 (Equation 5.3) that the pixel value u_i is: $u_i = e_i a_i$, where a_i is the albedo of the target object at the location pointed by u_i. Similarly, we can write: $m_i = e_m a_i$,with e_m assumed constant over the surface. This assumption is correct if the model image $\mathbf{m}$ has been taken under uniform illumination, or if a textured model free of illumination effects is available. Combining the above equations yields:

$$l_i = \frac{u_i}{m_i} = \frac{e_i}{e_m} ,$$

which does not depend on the surface albedo. It depends on the surface orientation and on the illumination environment. In the specific case of a planar surface lit by distant light sources and without cast shadows, this ratio can be expected to be constant for all i [102]. In the case of a three channel color camera, we can write the function l_i that computes a color illumination ratio for each color band:

$$l_i = \left[\begin{array}{ccc} \frac{u_{i,r}}{m_{i,r}} & \frac{u_{i,g}}{m_{i,g}} & \frac{u_{i,b}}{m_{i,b}} \end{array} \right]^T ,$$

where the additional indices r, g, b denotes the red, green and blue channel of pixel u_i, respectively.

In our illumination model we suppose that the whole surface can be described by K different illumination ratios, that correspond to areas in u_i with different orientations and/or possible cast shadows. Each area is modeled by a Gaussian distribution around the illumination ratio μ_k and with full covariance Σ_k. Furthermore we introduce a set of binary latent variables $x_{i,k}$ that take the value 1 if and only if pixel i belongs to Gaussian k and 0 otherwise. Then, the probability of the ratio l_i is given by:

$$p(l_i \,|\, x_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{k=1}^{K} \pi_k^{x_{i,k}} \mathcal{N}\left(l_i; \mu_k, \Sigma_k\right)^{x_{i,k}} , \tag{6.1}$$

where $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ denote all parameters of the K Gaussians. π_k weights the relative importance of the different mixture components. Even though the ratios l_i are not directly observed, this model has much in common with a generative model for illumination ratios.

So far we described the visible model. The occluded model is responsible for all pixels that do not correspond to the model image $\mathbf{m}$. These pixels are assumed to be generated by sampling the occluding color distribution, which we model as a mixture of $\bar{K}$ Gaussians and a uniform distribution. By this choice, we implicitly assume that the occluding object is composed of $\bar{K}$ colors μ_k, handled by the normal distributions $\mathcal{N}(u_i; \mu_k, \Sigma_k)$, and some suspicious pixels that occur with uniform probability $1/256^3$. Again, as in the visible model, the same latent variables are used to select a specific Gaussian or the uniform distribution. The probability of observing a

pixel value u_i given the state of the latent variable x_i and the parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ is given by:

$$p(u_i \,|\, x_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \left(\frac{\pi_{K+\bar{K}+1}}{256^3}\right)^{x_{i,K+\bar{K}+1}} \prod_{k=K+1}^{K+\bar{K}} \pi_k^{x_{i,k}} \mathcal{N}\left(u_i; \mu_k, \Sigma_k\right)^{x_{i,k}} \ . \tag{6.2}$$

The overall model consists of the visible (Equation 6.1) and the occluded (Equation 6.2) models. Our latent variables x_i select the one distribution among the total $K+\bar{K}+1$ components which is active for pixel i, i.e.

$$\forall i, \ \sum_{k=1}^{K+\bar{K}+1} x_{i,k} = 1 \, .$$

Consider Figures 6.1(a) and 6.1(b) for example: The visible pixels could be explained by $K = 2$ illumination ratios, one for the cast shadow and one for all other background pixels. The occluding hand could be modeled by the skin color and the black color of the shirt ($\bar{K} = 2$). The example in Figure 6.1 shows clearly that the importance of the latent variable components is not equal. In practice, there is often one Gaussian which models a global illumination change, *i.e.* most pixels are assigned to this model by the latent variable component $x_{i,k}$. To account for the possibly changing importance, we have introduced π_k that globally weight the contribution of all Gaussian mixtures $k = 1 \ldots K + \bar{K}$ and the uniform distribution $k = \bar{K} + 1$.

A formal expression of our model requires combining the visibility pdf of Equation 6.1 and the occlusion pdf of Equation 6.2. However, one is defined over illumination, whereas the other over pixel color, making direct probabilities incompatible. We therefore express the visible model as a function of pixel color instead of illumination:

$$p(u_i \,|\, x_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mid J_i \mid p(l_i \,|\, x_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \ , \tag{6.3}$$

where $\mid J_i \mid$ is the determinant of the Jacobian of function[1] $l_i(u_i)$. Multiplying this equation with Equation 6.2 composes the complete color pdf.

Some formulations define an appropriate prior model on the latent variables $\boldsymbol{x}$. Such a prior model would incorporate the prior belief that the model selection $\boldsymbol{x}$ shows spatial [89] and spatio-temporal [22] correlations. These priors on the latent variable $\boldsymbol{x}$ have shown to improve the performance of many vision algorithms [36]. However, they increase the complexity and slow down the computation substantially. To circumvent this, we propose in the next section a spatial likelihood model, which can be seen as a model to capture the spatial correlation nature of pixels and which gives real-time performance.

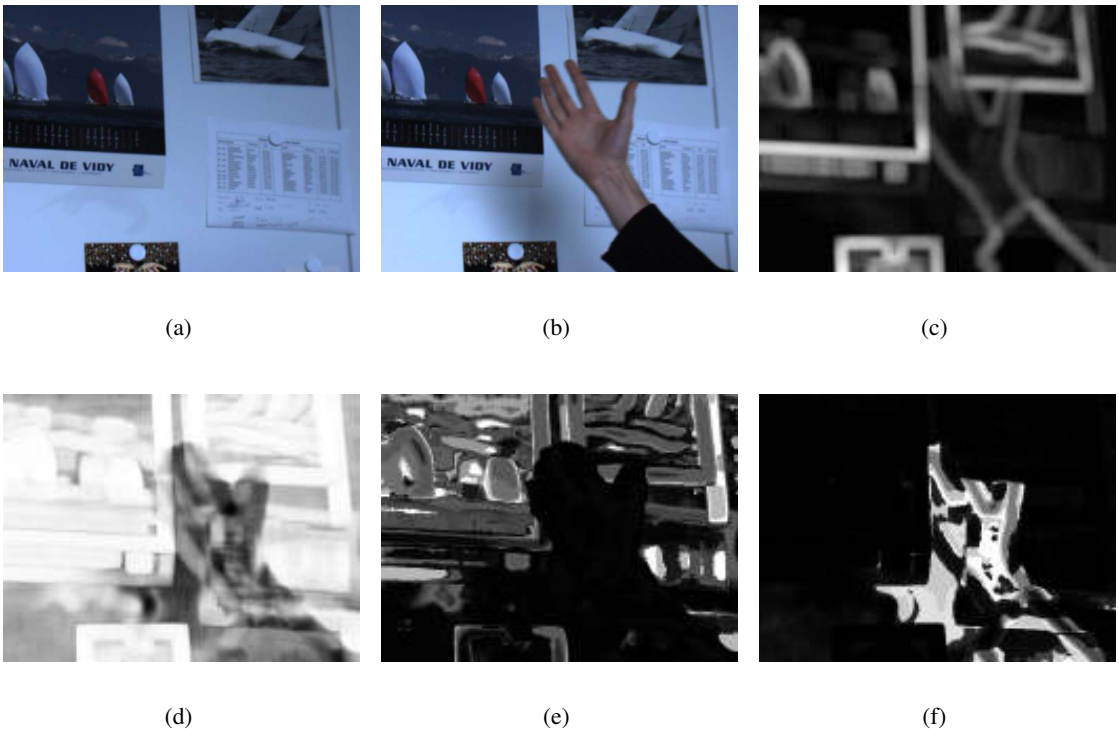
6.2.2. Spatial Likelihood Model

In this section, we present an image feature and a way to learn off-line its relationship with our target segmentation. Consider an extended image patch w_i around pixel i for which we ex-

[1] To establish Eq. 6.3, let $p(u)$ be a shortcut for the pdf $p(u_i \mid x_i, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $q(l(u))$ for $p(l_i \mid x_i, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. The probability of observing a pixel of color u within a color domain D is:

$$P(u \in D) = \int_D p(u) du = \int_{l(D)} q(x) dx = \int_D q\left(l(u)\right) \mid J \mid du \ ,$$

where $\mid J \mid$ denotes the determinant of the Jacobian of $l(u)$. It follows that $q(l(u)) \mid J \mid = p(u)$.



(a) (b) (c)

(d) (e) (f)

Figure 6.1.: Elements of the approach. (a) Background image $\mathbf{m}$. (b) Input image $\mathbf{u}$. (c) Textureness image $\boldsymbol{f}^2$. (d) Correlation image $\boldsymbol{f}^1$. (e) Probability of observing $\boldsymbol{f}$ on the background, according to the histogram $h(f_i \mid v_i)$ (f) Probability of observing $\boldsymbol{f}$ on the foreground, according to the histogram $\bar{h}(f_i \mid \bar{v}_i)$.

tract a low dimensional feature vector $f_i = [f_i^1, f_i^2]$. The basic idea behind our spatial likelihood model is to capture texture while keeping a pixel independence assumption. To achieve real-time performance we use two features that can be computed very fast. We model their distribution independently for the background and for the foreground, by histograms of the discretized feature values. We use the normalized cross-correlation (NCC) between the input and model images as one feature and a measure of the amount of texture as the other feature. f_i^1 is given by:

$$f_i^1 = \frac{\sum_{j \in w_i} (u_j - \bar{u}_i)(m_j - \bar{m}_i)}{\sqrt{\sum_{j \in w_i} (u_j - \bar{u}_i)^2 \sum_{j \in w_i} (m_j - \bar{m}_i)^2}},$$

where w_i denotes a window around pixel i, and $\bar{u}_i = \frac{1}{|w_i|} \sum_{j \in w_i} u_j$ is the average over w_i. The correlation is meaningful only in windows containing texture. Thus, the texturedness of window i is quantified by:

$$f_i^2 = \sqrt{\sum_{j \in w_i} (u_j - \bar{u}_i)^2} + \sqrt{\sum_{j \in w_i} (m_j - \bar{m}_i)^2}.$$

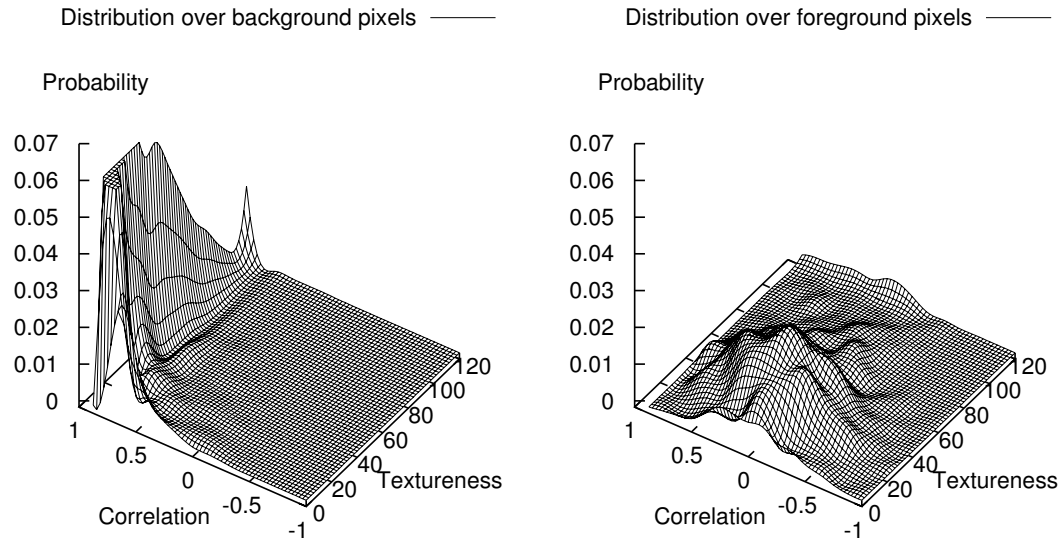


Figure 6.2.: Joint correlation and texturedness distributions over background and foreground pixels.

Let v_i represent the event that pixel i is visible and $\bar{v}_i$ that it is occluded. We denote the visible and occluded distributions by $h(f_i \mid v_i)$ and $\bar{h}(f_i \mid \bar{v}_i)$, respectively. They are trained from a set of manually segmented image pairs. Since joint correlation and amount of texture is modeled, the histograms remain valid for new illumination conditions and for new scenes. Therefore, the training is done only once, off-line. Once normalized, these histograms model the probability of observing a feature f_i on visible areas or on occluded areas. Figure 6.2 depicts both distributions. One can see that they are dissociate, especially in highly textured areas.

Figure 6.1 shows a pair of model and input images, the corresponding texture and correlation images f_i^2 and f_i^1, and the results of applying the histograms to $\boldsymbol{f}$. It is obvious that the correlation measure is only meaningful in textured areas. In uniform areas, because NCC is invariant to illumination, it can not make the difference between the target object exposed to some uniform illumination or a uniform occluding object.

Both histograms are learnt in the two visible and occluded cases which are related to the latent variable x_i designing one of the distributions of our model. Therefore, h can be linked to all visible distributions corresponding to $\{x_{i,1}, ..., x_{i,K}\}$ and $\bar{h}$ to all occluded ones, corresponding to $\{x_{i,K+1}, ..., x_{i,K+\bar{K}+1}\}$.

6.2.3. Maximum likelihood estimation

Having defined the illumination and the spatial likelihood model, we are now in the position to describe the Maximum Likelihood (ML) estimation of the combined model. Let $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}\}$

denote the vector of all unknowns. The ML estimate $\tilde{\boldsymbol{\theta}}$ is given by:

$$\tilde{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \{ \log \sum_{\boldsymbol{x}} p(\boldsymbol{u}, \boldsymbol{f}, \boldsymbol{x} \,|\, \boldsymbol{\theta}) \} \tag{6.4}$$

where $p(\boldsymbol{u}, \boldsymbol{f}, \boldsymbol{x} \mid \boldsymbol{\theta}) = p(\boldsymbol{u}, \boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{f}, \boldsymbol{x} \mid \boldsymbol{\theta})$ represents the combined pdf of the illumination and the spatial likelihood models given by the product of Equations. 6.3, 6.2 and the histogram distributions $h(f_i \,|\, v_i)$, $\bar{h}(f_i \,|\, \bar{v})$. Since the histogram distributions are computed over an image patch, the pixel contributions are not independent. However, in order to reach the real-time constraints, we assume the factorization over all pixels i in Equation 6.4 to be approximately true. We see this problem as a trade-off between *(i)* a prior model on $\boldsymbol{x}$, that models spatial interactions [22, 36] with a higher computational complexity and *(ii)* a more simple, real time model for which the independence assumption is violated, in the hope that the spatially dependent feature description $\boldsymbol{f}$ accounts for pixel dependence.

The pixel independence assumption simplifies the ML estimate to:

$$\tilde{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \{ \log \prod_{i} \sum_{x_i} p(u_i, l_i, f_i, x_i \,|\, \boldsymbol{\theta}) \} \tag{6.5}$$

The expectation-maximization (E-M) algorithm can maximize Equation 6.5. It alternates the computation between an expectation step (E-step), and a maximization step (M-step).

E-Step: On the $(t+1)^{th}$ iteration the conditional expectation $\boldsymbol{b}^{t+1}$ of the log-likelihood *w.r.t.* the posterior $p(\boldsymbol{x} \mid \mathbf{u}, \boldsymbol{\theta})$ is computed in the E-step. By construction, *i.e.* by the pixel independence, this leads to a closed-form solution for the latent variable expectations b_i, which are often called beliefs. Note, that in other formulations, where the spatial correlation is modeled explicitly, the E-step requires iterative approximations like mean field [36]. The update equations for the expected values $b_{i,k}$ of $x_{i,k}$ are given by:

$$b^{t+1}_{i,k=1\ldots K} = \frac{1}{N} \pi_k \mid J_i \mid \mathcal{N}(l_i; \mu^t_k, \Sigma^t_k) h(f_i \,|\, v_i) \tag{6.6}$$

$$b^{t+1}_{i,k=K+1\ldots \bar{K}} = \frac{1}{N} \pi_k \mathcal{N}(u_i; \mu^t_k, \Sigma^t_k) \bar{h}(f_i \,|\, \bar{v}_i) \tag{6.7}$$

$$b^{t+1}_{i,\bar{K}+1} = \frac{1}{N} \pi_{K+\bar{K}+1} \frac{1}{256^3} \bar{h}(f_i \,|\, \bar{v}_i) \;,$$

where $N = \sum_k b^{t+1}_{i,k}$ normalizes the sum of the beliefs $b^{t+1}_{i,k}$ to one. The first line in Equation 6.6 corresponds to the beliefs that the k^{th} normal distribution of the illumination visible model is active for pixel i. Similarly, the other two lines (Equation 6.7) correspond to the beliefs for the occluded model.

M-Step: Given the beliefs $b^{t+1}_{i,k}$, the M-step maximizes the log-likelihood by replacing the binary latent variables $x_{i,k}$ by their expected value $b^{t+1}_{i,k}$.

$$\mu^{t+1}_k = \begin{cases} \frac{1}{N_k} \sum_{i=1}^{N} b^{t+1}_{i,k} l_i & \text{if } k \leq K \\ \frac{1}{N_k} \sum_{i=1}^{N} b^{t+1}_{i,k} u_i & \text{otherwise} \end{cases} , \tag{6.8}$$

where $N_k = \sum_{i=1}^{N} b_{ik}^{t+1}$. Similarly, we obtain:

$$\Sigma_k^{t+1} = \begin{cases} \frac{1}{N_k} \sum_{i=1}^{N} b_{i,k}^{t+1} (l_i - \mu_k)(l_i - \mu_k)^{\mathrm{T}} & \text{if } k \leq K \\ \frac{1}{N_k} \sum_{i=1}^{N} b_{i,k}^{t+1} (u_i - \mu_k)(u_i - \mu_k)^{\mathrm{T}} & \text{otherwise} \end{cases} \tag{6.9}$$

$$\pi_k^{t+1} = \frac{N_k}{\sum_k N_k} \tag{6.10}$$

Alternating E and M steps ensure convergence to a local maximum. After convergence, we can compute the segmentation by summing the beliefs corresponding to the visible and occluded models. The probability of a pixel being described by the background model is therefore given by:

$$p(v_i \mid \tilde{\boldsymbol{\theta}}, \mathbf{u}) = \sum_{k=1}^{K} b_{i,k} \,. \tag{6.11}$$

In the next section, we discuss implementation and performance issues.

6.2.4. Implementation details

Our algorithm can be used in two different manners. First, it can run on-line, with a single E-M iteration at each frame, which is fast to compute. On very abrupt illumination changes, convergence is reached after a few frames (rarely more than 6). Second, the algorithm can run offline, with only two images as input instead of a video history. In this case, several iterations, typically 5 to 10, are necessary before convergence.

Local NCC can be computed efficiently with integral images, with a complexity linear with respect to the number of pixels and constant with respect to the window size. Thus, the complexity of the complete algorithm is also linear with the number of pixels, and the full process of acquiring, segmenting, and displaying images is achieved at a rate of about 2.3×10^6 pixels per second, using a single core of a 2.0GHz CPU. This is about 18 FPS for half PAL (360x288), 12 FPS for 512x384, and 5-6 FPS for 720x576 images.

Correlation and texturedness images, as presented in Section 6.2.2, are computed from single channel images. We use the green channel only, because it is more represented on a Bayer pattern. The correlation window is a square of 25×25 pixels, cropped at image borders.

For most experiments presented in this work, $K = 2$ and $\bar{K} = 2$. The histograms h and $\bar{h}$ have been computed only once, from 9 pairs of images (about 2×10^6 training pixels). Training images do not contain any pattern or background used in test experiments.

The function l_i as presented in the previous section is sensitive to limited dynamic range and to limited precision in low intensity values. Both following functions assume the same role with more robustness and give good result:

$$l_i^a(u_i) = \left[\arctan\left(\frac{u_{i,r}}{m_{i,r}}\right) \arctan\left(\frac{u_{i,g}}{m_{i,g}}\right) \arctan\left(\frac{u_{i,b}}{m_{i,b}}\right) \right]^T$$

$$l_i^c(u_i) = \left[\frac{u_{i,r} + c}{m_{i,r} + c} \frac{u_{i,g} + c}{m_{i,g} + c} \frac{u_{i,b} + c}{m_{i,b} + c} \right]^T$$

where c is an arbitrary positive constant. In our experiments, we use $c = 64$.

6.2.5. Optional Graph-Cut for spatial coherence

The expectation-maximization algorithm presented above assumes pixel independence. Thanks to cross-correlation that captures local texture, this abusive assumption does not degrade results. However, produced segmentation are sometimes noisy. In that case, the segmentation problem taking spacial consistency into account can be formulated as a weighted graph minimum cut problem, for which efficient and globally optimal algorithms exist.

In theory, integration of a graph-cut algorithm in E-M is not directly possible, since the convergence properties of E-M rely on computing the expectation of latent variables, while graph-cut provide a binary segmentation. In practice, it is still possible to use it within an iterated algorithm [89] or as a post-processing step following E-M, which is the approach we choose here.

The graph is formulated as follow. The source node is linked to every pixel i, with a weight of $-\log\, p\left(v_i \mid \tilde{\boldsymbol{\theta}}, \mathbf{u}\right)$, the negative log-likelihood of pixel i to be visible (Equation 6.11). The sink node connects pixels with a weight of $-\log\, p\left(\bar{v}_i \mid \tilde{\boldsymbol{\theta}}, \mathbf{u}\right)$ and accounts for occluded pixels. Each pixel is connected to its neighbors with a weight defined by the function $\kappa_{i,j}$. In our case, we chose a constant weight for a 4-neighborhood[2]:

$$\kappa_{i,j} = \begin{cases} 1 & \text{if } i,j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases}$$

The minimum cut $\hat{\mathcal{P}}$ minimizes the following function:

$$\min_{\mathcal{P}} -\sum_{i\in\mathcal{P}} \log p\left(v_i \mid \tilde{\boldsymbol{\theta}}, \mathbf{u}\right) - \sum_{i\notin\mathcal{P}} \log\, p\left(\bar{v}_i \mid \tilde{\boldsymbol{\theta}}, \mathbf{u}\right) + \sum_{i\in\mathcal{P}}\sum_{j\notin\mathcal{P}} \kappa_{i,j}\ .$$

It equivalently maximizes:

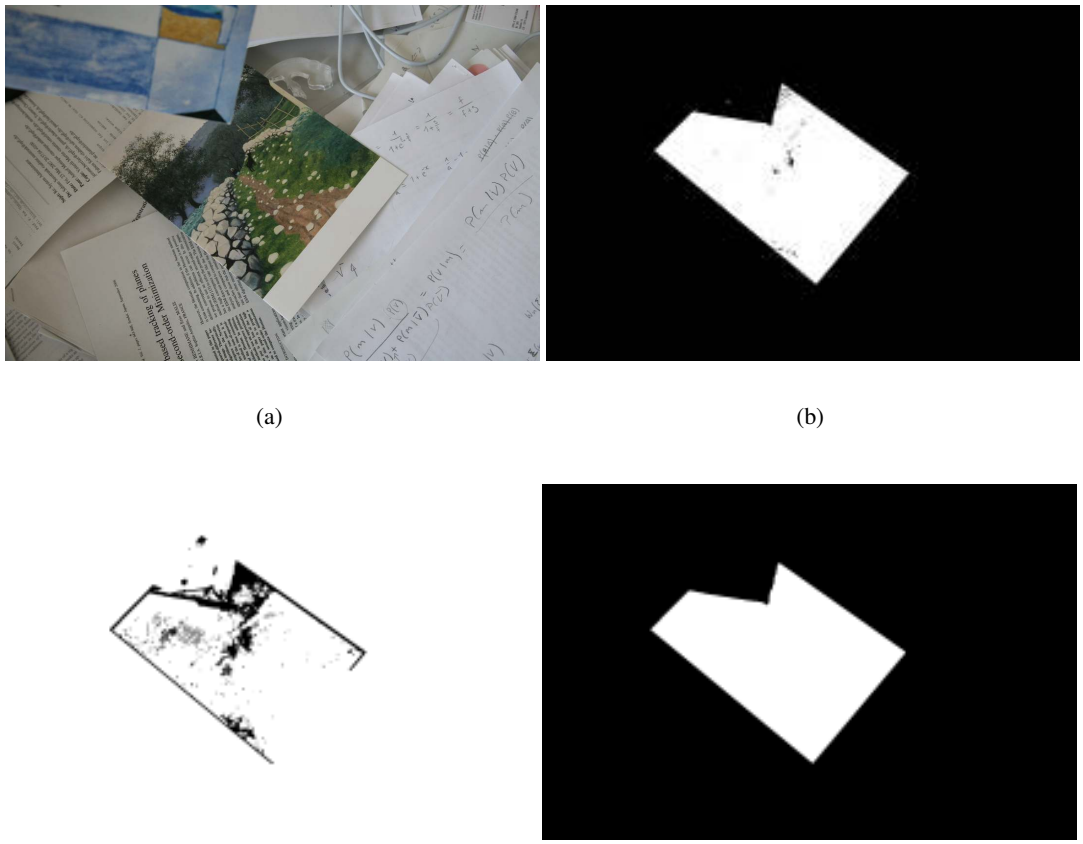
$$\max_{\mathcal{P}} \prod_{i\in\mathcal{P}} p\left(v_i \mid \tilde{\boldsymbol{\theta}}, \mathbf{u}\right) \prod_{i\notin\mathcal{P}} p\left(\bar{v}_i \mid \tilde{\boldsymbol{\theta}}, \mathbf{u}\right) \prod_{i\in\mathcal{P}}\prod_{j\notin\mathcal{P}} e^{-\kappa_{i,j}}\ .$$

This minimum cut problem can be efficiently solved by considering the dual maximum flow problem [15, 59, 89].

Even if the max-flow algorithm we use is efficient, segmenting a full image implies quite a large graph and takes too much time for real-time applications. To improve speed, we actually use an approximation that consists in cutting several separate small graphs rather than a single large one. Computation is then concentrated on ambiguous pixels and does not loose time questioning obvious ones. Figure 6.3 depicts such a partition of ambiguous areas.

Our algorithm works as follow. In a first pass, pixels are labeled as either definitely visible, definitely occluded, or unknown. Labeling is done by thresholding the probability of Equation 6.11. Above $1-e$, it is visible. Under e, it is occluded. We take $e = 0.001$. A second pass tags the neighbors of unknown pixels as unknown. A third pass assigns connected pixels

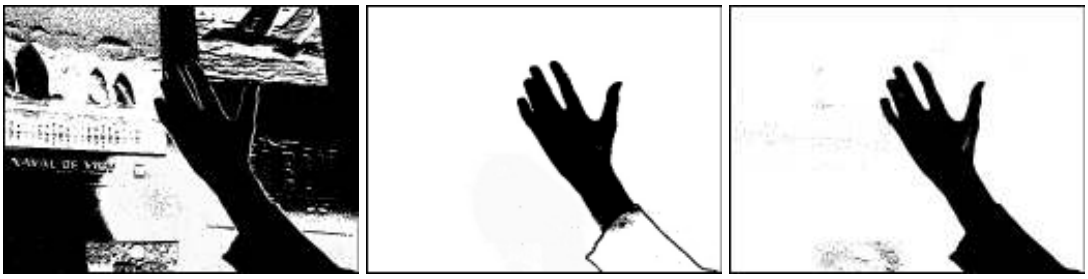
[2] Another option is to compute a weight based on the gradient of the input image, to model the idea that a change in pixel values is often observable at occlusion boundaries [15].



(a) (b)

(c) (d)

Figure 6.3.: Smoothing the visibility map using Graph Cut. (a) Input image. (b) Visibility probability of Equation 6.11, locally noisy. (c) Partition of the noisy areas. Each partition is displayed with a different gray shade and composes a dissociate graph. (d) The result of computing the minimum cut in each graph.



(a) (b) (c)

Figure 6.4.: Segmenting the hand of Figure 6.1. (a) Result of [124] when the background model adjusts too slowly to handle a quick illumination change. (b) When the background model adjusts faster. (c) Our result.

to a graph using a flood fill algorithm. The maximum flow is computed, resulting in a minimum cut and finally a segmentation that updates pixel label to either definitely occluded or definitely visible.

The output of the algorithm is a smooth, spatially coherent, binary segmentation.

6.3. Results

The first step in our occlusion detection method consists in *unwarping* the input image. It represents the appearance of the object in the conditions in which the input image has been taken and allows pixel to pixel comparison with the model image. We tested our algorithm with three different geometric transformations. The simplest unwarping is the identity transform: The unwarped image is the input image. In that case, the model image must be taken under the same pose, and the problem is called background subtraction and is not directly useful for augmented reality. The second type of geometric unwarping applies to planar objects. We have seen in Chapter 4 how to obtain the homography registering the model image of a planar object with a different view of the same object. Inverting an homography is as easy as computing a 3 by 3 matrix inverse. In that case, composing the unwarped image is done by applying the appropriate homography to the input image. The third type of tested geometric transform is the 2–D hexagonal mesh of Section 4.2.1, it is able to model many deforming surfaces and results are presented in the next chapter.

Comparison with background subtraction techniques Our approach to occlusion segmentation is closely related to background subtraction, a technique aiming at segmenting moving object in video sequences shot by a fixed camera. We show here that our method can directly be applied for video surveillance, and that our way of handling illumination changes is particularly robust compared to standard background subtraction practices.

We begin by the sequence of Figure 6.4 in which an arm is waved in front of a cluttered wall.

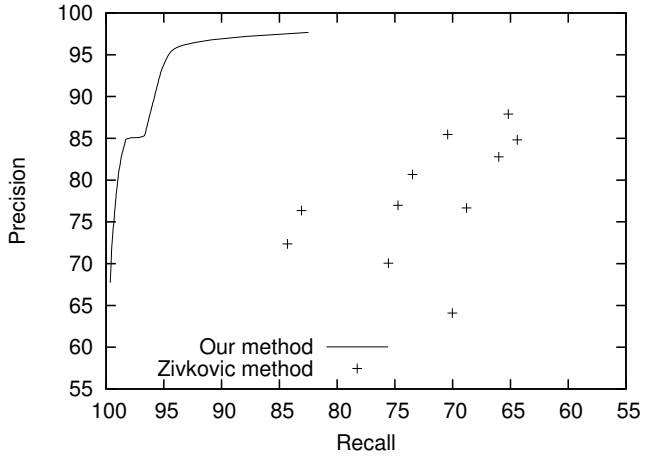


Figure 6.5.: ROC curve for our method obtained by varying a threshold on the probability of Equation 6.11. The crosses represent results obtained by [124] for different choices of learning rate and decision threshold, two of which are visible on Figure 6.4(a) and 6.4(b).

The arm casts a shadow, which affects the scene's radiosity and causes the camera to automatically adapt its luminosity settings. With default parameters, the algorithm of [124] reacts to this by slowly adapting its background model. However, this adaptation cannot cope with the rapidly moving shadow and produces the poor result of Figure 6.5(a). This can be prevented by increasing the rate at which the background adapts, but, as shown in Figure 6.5(b), it results in the sleeve being lost. By contrast, by explicitly reevaluating the illumination parameters at every frame, our algorithm copes much better with this situation, as shown in Figure 6.5(c). To compare these two methods independently of specific parameter choices, we computed the ROC curve of Figure 6.5(d). We take precision to be the number of pixels correctly tagged as foreground divided by the total number of pixels marked as foreground and recall to be the number of pixels tagged as foreground divided by the number of foreground pixels in the ground truth. The curve is obtained by binarizing using different thresholds for the probability of Equation 6.11. We also represent different runs of [124] by crosses corresponding to different choices of its learning rate and the decision threshold. As expected, our method exhibits much better robustness towards illumination effects.

Figure 6.6 depicts a sequence with even more drastic illumination changes that occur when the subject turns on one light after the other. The GMM based-method [124] immediately reacts by classifying most of the image as foreground. By contrast, our algorithm correctly compares the new images with the background image, taken to be the average of the first 25 frames of the sequence.

Figure 6.7 shows the *light switch* benchmark of [109]. We again built the background representation by averaging 25 consecutive frames showing the room with the light switched off. We obtain good results when comparing it to an image where the light is turned on even though, unlike the other algorithms [124, 49], we use a single frame instead of looking at the whole video.

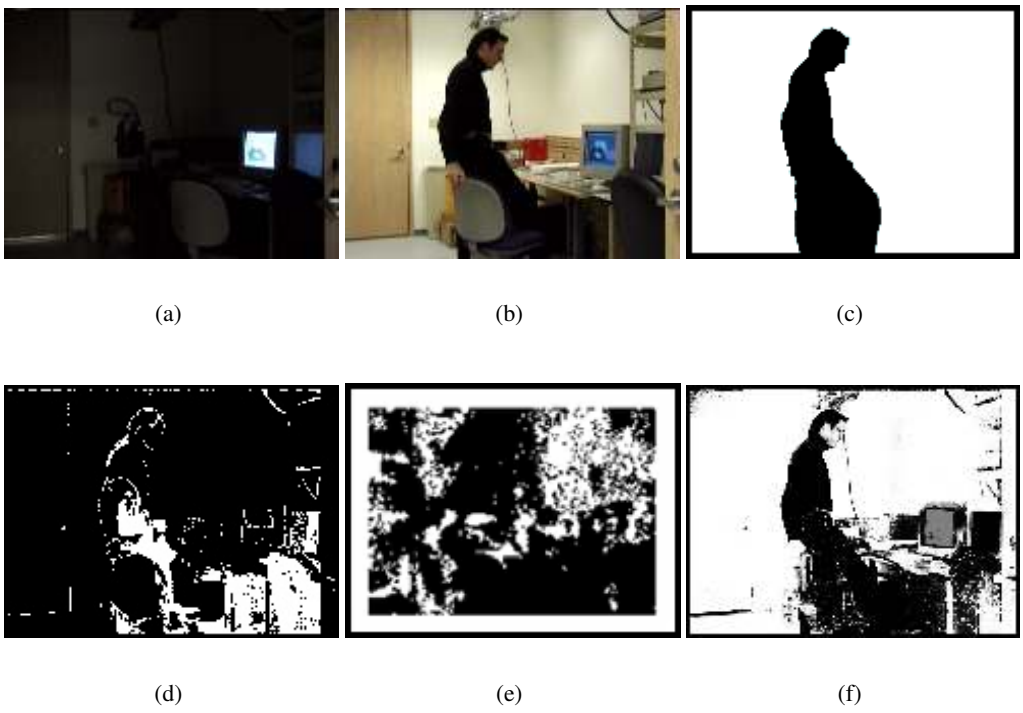
Figure 6.6.: Top row: Three very different input images and a model image of the same scene. The changes are caused by lights being turned on one after the other and the person moving about. Bottom row: Our algorithm successfully segments out the person in all three input images. The rightmost image depicts the completely wrong output of a state-of-the-art approach [124] applied on the third image.

The foreground recall of 82% that appears in [49] entails a precision of only 25%, whereas our method achieves 49% for the same recall. With default parameters, the algorithm of [124] cannot handle this abrupt light change and yields a precision of 13% for a recall of 70%.

Finally, as shown in Figure 6.8, we ran our algorithm on one of the PETS 2006 video sequences that features an abandoned luggage to demonstrate that our technique is indeed appropriate for surveillance applications because it does not lose objects by unduly merging them in the background.

Planar Objects Our method works well with planar objects. The geometric transform they undergo is well modeled by the recovered homography, even if many features are occluded: The few visible ones usually bring enough information to recover the pose. Since the orientation of a planar object is by definition homogeneous, illumination effects are expected to be generally simple. In practice, those caused by the occluding elements themselves are complex. In the case of fingers holding a card, as depicted by Figure 6.9, the user's hands cast shadows whose intensity can be important. Therefore, modeling illumination with a mixture of two Gaussians, as presented in this chapter, is adapted: The first Gaussian accounts for the global illumination over the card, and the second for the shadow under the finger. The softer the shadow, the more elongated the Gaussian in the intensity axis. For proper augmented reality, the visibility map, computed in model image space, has to be *warped* again. It can be used as an alpha channel that masks occluded virtual elements.

Non-rigid Surfaces Among the considered geometric cases, the non-rigid one is the most general and the most difficult for several reasons. The many degrees of freedom of a non-rigid model make difficult to compensate the lack of information due to occlusions, as opposed to the



(a) (b) (c)

(d) (e) (f)

Figure 6.7.: Segmenting the *light switch* test images from [109]. (a) Background model. (b) Test image. (c) Manually segmented ground truth. (d) The output of Zivkovic's method [124]. (e) Result published in [49], using an approach based on local binary patterns. (f) Our result, obtained solely by comparing (a) and (b). Unlike the other two methods, we used no additional video frames.

rigid planar case. Illumination effects are also more complex, since the surface orientation is heterogeneous. In that case, our assumptions of two Gaussians can be seen as modeling parts of the surface that are directly illuminated, while the other Gaussian accounts for indirect light on shaded areas. Shadows are either self-cast or cast by an occluding object.

In our experiments, unwarping was simply done by sampling the input image with coordinates transformed with the function $T_\theta(\mathrm{p})$ of Equation 4.1, a task conveniently executed by the GPU. Figure 6.10(d) shows such an unwarped image. The quality of the unwarped image is directly linked with the registration accuracy which might decrease in the presence of strong occlusions. The abundance of texture painted on our test T-shirt allows us to circumvent this issue. The little registration jitter is passed on the unwarped image and further on the normalized cross-correlation step of occlusion detection. In this case, the graph cut smoothing technique of Section 6.2.5 greatly improves the final visual quality.

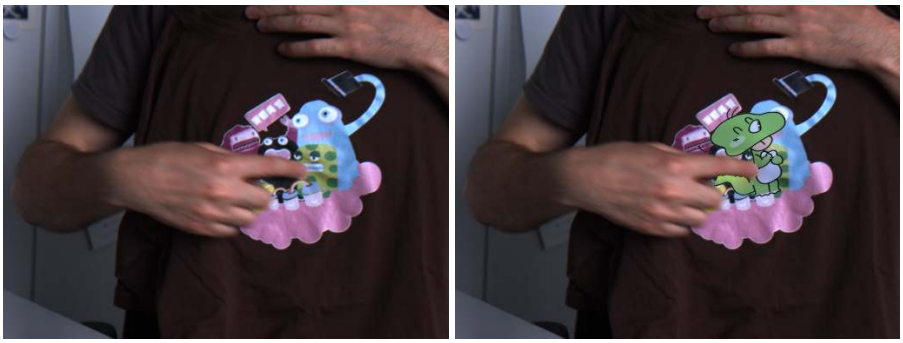
(a) (b)

(c) (d)

Figure 6.8.: PETS 2006 Dataset. (a) Initial frame of the video, used as background model. (b) Frame number 2800. (c) The background subtraction of [124]: The abandoned bag in the middle of the scene has mistakenly been integrated into the background. (d) Our method correctly segments the bag, the person who left after sitting on the bottom left corner, and the chair that has been removed on the right.

(a) (b) (c)

Figure 6.9.: Occlusion segmentation on a moving object. (a) Input frame in which the card is tracked. (b): Visibility mask produced by our method. (d) We use it as an alpha channel to convincingly draw the virtual text and account for the occluding hand.



(a) (b)

(c) (d) (e)

Figure 6.10.: Augmenting an occluded T-shirt. (a) Input frame. (b) Augmented result. (c) Reference image. (d) Unwarped image. (e) Visibility map computed from (c) and (d).

7. Putting it All Together

This Chapter presents AR results obtained by putting together the presented algorithms, thus verifying their suitability for real AR applications that require robustness, ease of deployment, computation speed, accuracy, and visual quality.

7.1. Artistic Augmented Reality

To check that our approach to AR actually corresponds to the constraints of a real application, we collaborated with an artist to create two AR applications. Camille Scherrer, a student at the university of art and design, Lausanne (ECAL), imagined, designed, and manufactured two books and animations to augment their pages using our method.

Several points facilitated our collaboration. First, since our approach only relies on natural texture, the artist could design the book without having to integrate technical elements. Second, the choice of 2–D AR instead of 3–D facilitated the integration of virtual elements with the real pages: The artist was able to paint augmented creatures directly over a reference frame, immediately seeing the mixed result. At runtime, our system could then easily register the reference frame and warp the virtual creatures to the correct location and perspective. Third, the automation of our approach allowed the artist to insert new pages in the system without requiring any technical support since adding a new pattern does not require any parameter tuning. In this manner, the artist has been able to produce two artworks, *The Haunted Book* and *Le monde des montagnes*.

The Haunted Book The Haunted Book, depicted by Figure 7.1, is inspired by old poetry books. The integration of very recent technologies into this old and dusty universe makes it unusual and particularly interesting. The Haunted Book is based on a poem written by Thomas Hood, The Haunted House. As the narrator in the poem walks through the haunted house, the reader also walks through the book and discovers hidden creatures.

The artist's interpretations of these hidden creatures appear as a skeleton's arm grabbing out of a letterbox, flying fish jumping out of an old cupboard, and loads of ugly insects running down a sofa. The animated engravings create a subtle and adapted way to enhance the illustrations by staying in the universe of the poem. The AR technology used for the project yields this subtlety by staying discreet.

Le monde des montagnes *Le monde des montagnes*, the world of mountains, is a very particular book. When one observes it through the eye of a camera, a whole invisible universe reveals itself beyond the printed pages. Between remembering and strange stories, the reader discovers, page after page, an animated world that mixes with reality. The artwork is based on

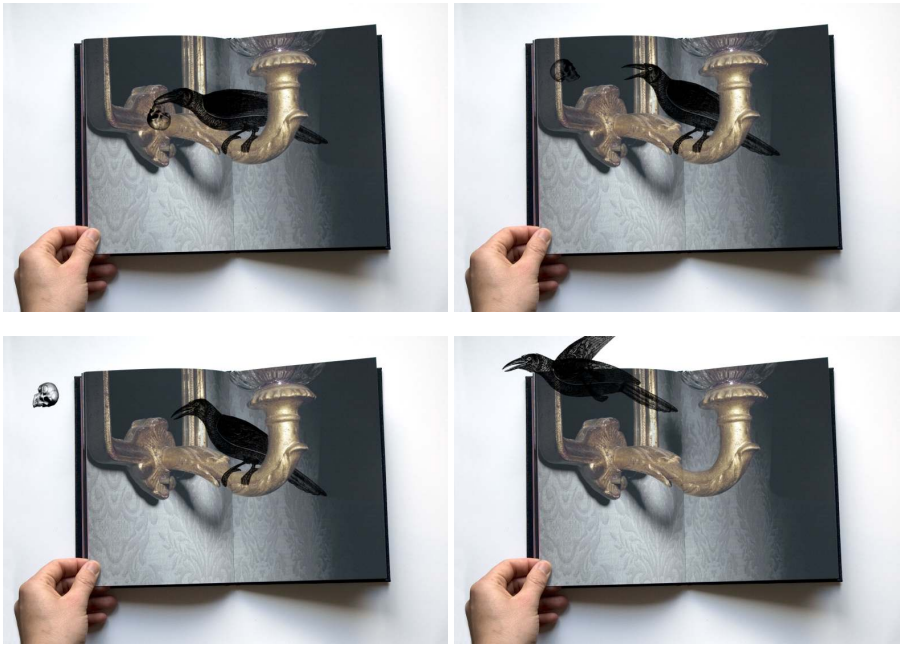


Figure 7.1.: An example of animation from the Haunted Book.

Figure 7.2.: The AR setup developed by Camille Scherrer for augmenting a magic book. The camera is hidden in the lamp, and augmented images are displayed on the laptop screen.

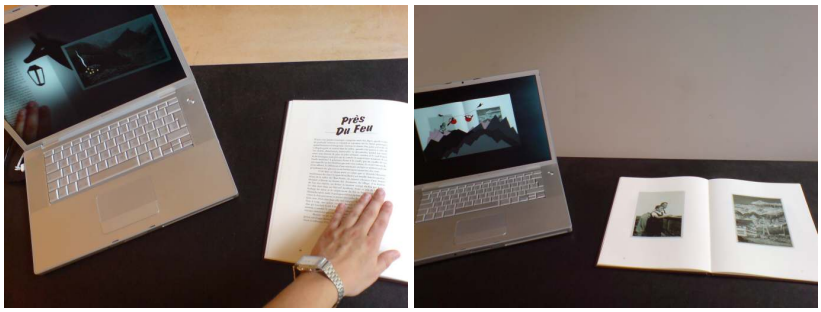


Figure 7.3.: Augmented pages of a magic book designed by Camille Scherrer.

two main elements. First, an illustrated book telling stories happening in the mountain. Second, a laptop displaying images shot by a camera, as depicted by Figure 7.2. The pages of the book appear augmented by animations that smoothly blend into the original book illustrations, as shown by Figure 7.3. The unobtrusiveness of AR in our approach is here further improved by hiding the camera in the lamp.

The Haunted Book and *Le monde des montagnes* demonstrate an innovative way of integrating virtual creatures in the real pages of a book. The augmented books create a desire of searching for moving pictures through the pages. AR makes paper and computer screen meet, extracting a part of imaginary out of a reality our eyes can see and our hands can concretely feel.

The resulting atmosphere convinced several art professionals, namely Alain Bellet (Head of the Media and Interaction Design Unit at écal), Angelo Benedetto (Head of the Visual Communication Unit at écal), Michael Zai (founding member of etoy and professor at the Media and Interaction Design Unit at écal), and Gael Hugo. The success of our collaboration and the quality of result prove that our approach has contributed to increasing the maturity and usability of augmented reality techniques [95].

7.2. 3–D Non-rigid Augmented Reality

To verify the capability of our approach to provide 3–D non-rigid AR, we designed an experimental application. The user holds and deforms in its hand a textured sheet of paper. A fixed camera oriented towards the user films the scene. A computer screen displays the augmented stream. Four virtual pyramidal poles appear to be to be glued to the paper, and electric arcs connect their spikes. As the paper deforms, the poles' orientation changes, modifying the path of electric sparks. New users usually need less than a minute to understand this interaction. They can then play to give the electric arc the shapes they want, as depicted by Figure 7.4.

This experimental application relies on the registration method described in Section 4.3. It runs at about 7-10 frames per second. It shows that our system is robust to user actions, that it has a delay short enough and a frame rate fast enough for fluent interaction, and that its accuracy is sufficient. Our system rises to the augmented reality challenge. Overall, users enjoyed the application. Its reactivity and accuracy produce a convincing augmented reality effect.

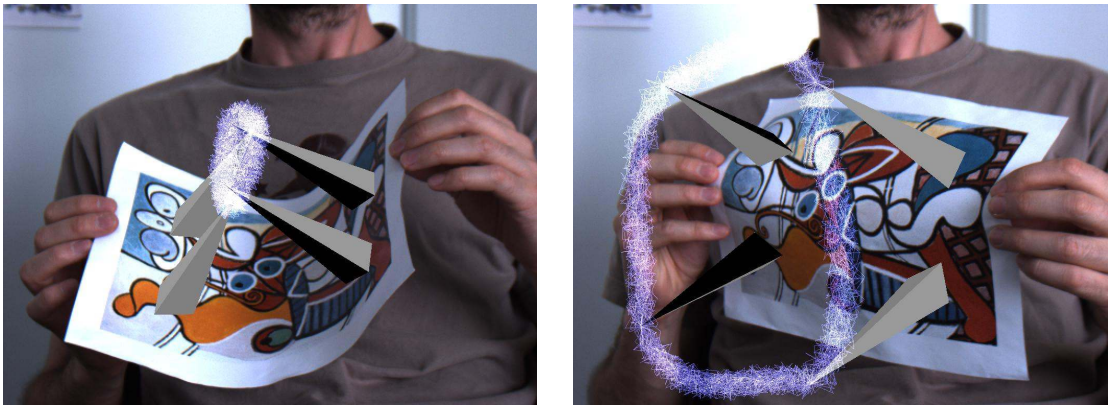


Figure 7.4.: A toy application to illustrate our method's suitability for 3–D non-rigid augmented reality purposes.

Figure 7.5.: A deformed sheet of paper augmented with a virtual tree and a virtual monster. The author's finger clearly appears *over* the augmented tree. The shadow due to paper bending is also visible on the tree. Left: Input frame. Right: Augmented result.

7.3. Retexturing Non-rigid Surfaces

We present here results that achieve realistically shaded and occluded non-rigid retexturing. Our system superimposes a partially transparent virtual layer over a deforming T-shirt or sheet of paper. The augmentation appears to be glued to the surface. Its texture is modified and we call that process retexturing. Achieved in real-time and interactively, it is a flexible form of AR. Our approaches to non-rigid registration and realistic shading allows for convincing replacement of the real texture with a logo, a blank page, or a virtual creature. It combines the 2–D non-rigid registration of Section 4.2, the illumination handling for retexturing of Section 5.3 and the occlusion handling of Chapter 6. Illumination effects on the real surface are reproduced on the augmented one. Occlusions are also handled, and fingers or other real object can occlude virtual ones. Figure 7.5 depicts a frame of such an augmented video stream.

Our system runs at a rate of about 6-8 frames per second, for 512×384 input images. Ta-

Task	Rel. Weight
Feature detection	6.5%
Feature matching	23.7%
Registration	52.9%
E-M for occlusion segmentation	13.2%
Graph Cut	3.7%

Table 7.1.: The tasks carried out by our system and their relative computation time.

ble 7.1 summarizes the relative computation weight of the different tasks. The time profile depends on many parameters such as the input video resolution or the number of detected feature points. Moreover, several parts could be accelerated by using the GPU. Therefore, Table 7.1 only provide a rough idea of the relative importance of each stage. Unsurprisingly, the heaviest task is non-rigid registration, more precisely the successive optimizations of Equation 4.2 with different radii of confidence. They take about 53% of executed instructions. The second task is wide-baseline feature matching, with about 24%.

Our prototype proves the feasibility of augmented reality for non-rigid surfaces. The result is fast enough for interactive use. Illumination and occlusion effects provide a realistic visual quality.

8. Conclusion

We presented a framework for augmenting images of non-rigid surfaces acquired by a single standard camera. We developed new algorithms for geometric registration, illumination estimation, and occlusion segmentation. We emphasized all along accessibility both for end-users and for application designers. We ensured the consistency of our framework by making all its components comply to a common set of constraints:

- It can all handle both rigid and non-rigid surfaces;
- It does not require any engineering of the scene;
- It runs in real-time;
- It requires a single camera but can take advantages of additional ones, if available;
- It delivers high visual quality;
- It is easy to use, both for end-users and for application designers.

We demonstrated it using a wide range of deforming objects such as sheets of paper, rubber balloon, sails and T-shirts. Our key contribution has therefore been to make AR possible in such a challenging context and we conducted many experiments to support this claim.

Observed automated initialization, tracking accuracy, and computation speed prove that our framework is a solid basis for convincing AR applications. One of them is a toy application in which the user plays with an electric arc deforming with poles virtually hammered into a real sheet of paper. Novice users very quickly understand the interaction principle and enjoy it, thus validating our approach. The suitability of our framework for AR application has also been demonstrated by augmenting target objects of size ranging from a credit card to a board of about 100 by 80 cm.

To ensure that our framework is easy to use and to deploy, we distributed a software package, BazAR [65]. A number of users successfully used it to good effect. Among them are the French company Total Immersion, the Norwegian one ARmusement, and the artist Camille Scherrer. Her two artworks showcase the high visual quality that can be achieved using our software. Furthermore, it is easy to deploy because it handles one or more camera, it only makes weak assumptions on the environment, and it runs on standard hardware.

Even in the presence of complex illumination conditions, our framework is able to render virtual objects shaded with real light. We tested this ability by augmenting a card with a virtual teapot that reflects the change in illumination color occurring when the user switches a lamp on. Our approach to retexturing non-rigid surfaces also handles complex illumination effects, such as saturation, cast shadows and specularities. To verify this behavior, we successfully

and realistically augmented a deforming T-shirt partially illuminated with a projector. We also augmented a sheet of paper in a plastic cover causing specularities. The reflections of a lamp and a window are visible on the augmented image, proving the flexibility of our framework in that domain.

Poor handling of virtual objects occluded by real ones might cause augmentation to be uncomfortable and unrealistic. Therefore, our framework integrates a new method for occlusion segmentation and we successfully retextured a moving planar card and a deforming T-shirt occluded by fingers and other objects.

Dealing with multiple cameras in AR requires their relative registration, a process that can be tedious. However, our new approach to geometric and photometric camera calibration handles it gracefully, requiring minimal interaction from the user. We were able to calibrate five cameras accurately enough for AR purposes, or two webcams plugged on the same laptop. Once calibration is done, our framework can augment every view, using any camera that can see the target object. If the object is seen by several cameras, the accuracy is improved.

Overall, our experiments well demonstrate the capacities of our framework. They prove the possibility of achieving convincing AR on deforming surfaces only with a monocular camera.

8.1. Impact of the Thesis

Beside our scientific publications, our work had an impact for a number of users, through the several pieces of software we distributed.

Deform3D Deform3D is an image analysis software that measures the deformations of non-rigid objects. Starting from a reference shape and at least one picture of the deformed object, Deform3D recovers both its deformation and the camera pose. It has been used by Voiles Phi SA, a Swiss sailmaking company, to improve the design of Alinghi's spinnakers. It integrates Mathieu Salzmann's deformation models [90, 92]. The graphical user interface, visible on Figure 8.1, has been developed by André Mazzoni and Konstantin Starchev.

Alinghi Demonstration We developed an interactive demonstration of non-rigid sail measurement. It is composed of a piece of sail on a mobile support, a webcam, and two screens. Users can deform the piece of sail and immediately see on one screen that the 3–D representation follows. The other screen shows the image acquired by the webcam, augmented with the surface mesh. As depicted by Figure 8.2, the demonstration was open to the public in Valencia, Spain, during about 18 months. It was also exposed in the Olympic Museum, Lausanne, from September 20, 2007 to January 6, 2008.

BazAR BazAR is a software package we distributed under the GNU General Public License (GPL) in October 2006 [65]. It contains the planar object detector and the geometric camera calibration both described in Chapter 4. It also includes photometric calibration and is able to augment 3–D objects, as explained in Chapter 5. It has been licensed to several companies, and we received feedback by e-mail from France, Norway, Italy, United Kingdom, Spain, Denmark,

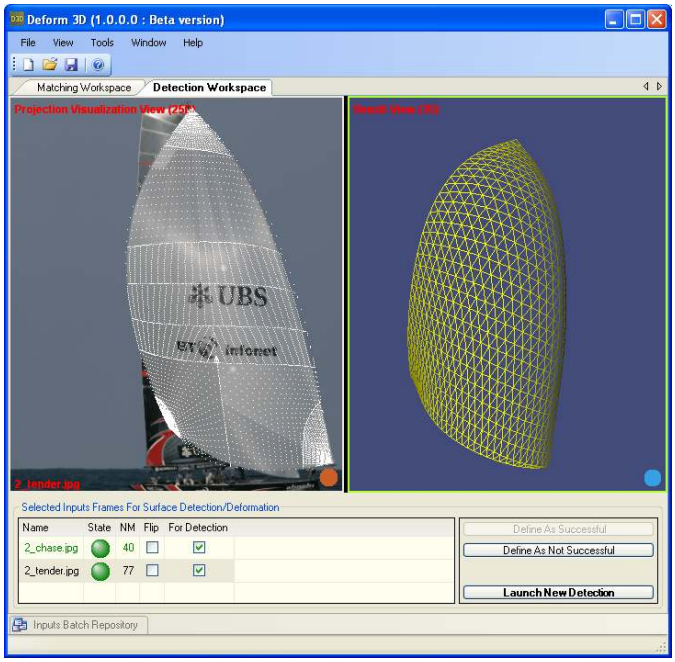


Figure 8.1.: A screenshot of Deform3D, our sail deformation measurement software.

Figure 8.2.: Our real-time demonstration in Alinghi base, Valencia, Spain. Visitors can deform a piece of sail and see both the image filmed by a webcam and the 3–D deformed shape.

Sweden, Czech Republic, Germany, Singapore, Israel, Lebanon, Poland, United States, China, Netherlands, New Zealand, and Canada.

8.2. Limitations and Future Work

When a surface deforms, its shading changes. Bumps and folds form shaded and highlighted areas whose analysis could be used to improve registration accuracy. Because our method only considers texture, it heavily depends on it. Such an improvement can potentially reduce this dependency.

The presented approach is mostly based on feature points. Therefore, texture such as straight lines remains unexploited. Techniques such as template matching or texture edge tracking could also improve accuracy and decrease texture dependency.

In our method, self-occlusions are ignored. Integrating them in the process could allow better accuracy. It would also make the method compatible with non-convex objects having large self-occluded areas.

A limitation of our system is the need for a manually created model of the target object. Even if this often reduces to taking a single picture, this process could be extended to automatic surface recognition and modeling from video. One could imagine a video editing software that automatically detects surfaces visible in a sequence. It would then let the user select a surface in one frame and edit its appearance. The software could then automatically apply the modification on all frames showing the same surface, handling gracefully deformation, illumination effects and occlusions.

Our camera geometric and photometric calibration is limited to a pinhole camera model. A possible future work could extend it to a thin lens camera model. It would imply estimating vignetting, radial distortion, focal distance, camera aperture, and point spread function. Such an extended model would allow realistic integration of virtual objects on images taken by cheap and wide angle cameras with important distortion. Rendering out-of-focus virtual objects would also become possible.

A further improvement would be to change our way of modeling illumination. Until now, we do not make any assumption on the illuminant color, letting it live in a 3–D space. However, it has been demonstrated that assuming a Plankian color is a reasonable assumption, reducing the uncertainty of the illuminant to its scalar temperature [33]. Such an assumption requires camera response calibration. It would greatly help for separating texture and light patterns. The occlusion segmentation of Chapter 6 would also be simplified, since the 3–D Gaussian mixture model could be reduced to a single dimension.

Our approach to illumination does not take shadows of virtual objects over real ones into account. The system could be extended to estimate the geometry of the environment and the direction of incoming light in order to cast virtual shadows on real surfaces.

A. Wide-baseline Keypoint Matching

The goal of keypoint matching is to establish correspondences between two images of the same object. Instead of matching every pixel, which would be both costly and difficult in uniform areas, an efficient option is to first try to detect points that are easier to locate, called *keypoints*. We do not address here the issue of keypoint extraction, since we use a rather standard approach. However, because we intensively rely on it, we present a description of an approach to point matching under large viewpoint and illumination changes.

A.1. Keypoint Matching as a Classification Problem

Most of traditional point matching methods rely either on using ad hoc local descriptors or on estimating local affine deformations [110, 6, 76, 94, 69]. By contrast, we treat wide-baseline matching of keypoints as a classification problem, in which each class corresponds to the set of all possible views of such a point.

During training, given at least one image of the target object, we synthesize a large number of views of individual keypoints. If the object can be assumed to be locally planar, this is done by simply warping image patches around the points under affine or homographic deformations. Otherwise, given the 3–D model, standard Computer Graphics texture-mapping techniques can be used. This second approach is more complex but relaxes the planarity assumptions. At run-time, it is then possible to use powerful and fast classification techniques to decide to which view set, if any, an observed keypoint belongs, which is as effective and much faster than the usual way of computing local descriptors and comparing their responses [64].

Figures A.1 and A.2 depict the construction of the set of image patches that represent the class corresponding to one point. A single object can have many keypoints, each of which, in turn, has many possible appearances. If some classification method could tell, given a new patch, to which point it belongs, it would provide a strong link between the image and the model. Several of these correspondences allows registration, as explained in Chapter 4. In our work, we used two methods: Randomized trees [63] and Ferns [80]. We only give here a description of ferns, because they outperforms decision trees.

A.2. Random Ferns[1]

To classify patches, we use the method first introduced in [80] that rely on non-hierarchical structures called ferns. Each one consists of a small set of binary tests and returns the probability that a patch belongs to any one of the classes that have been learned during training. These

[1]The author would like to thank Mustafa Özuysal and Vincent Lepetit for the description of ferns.

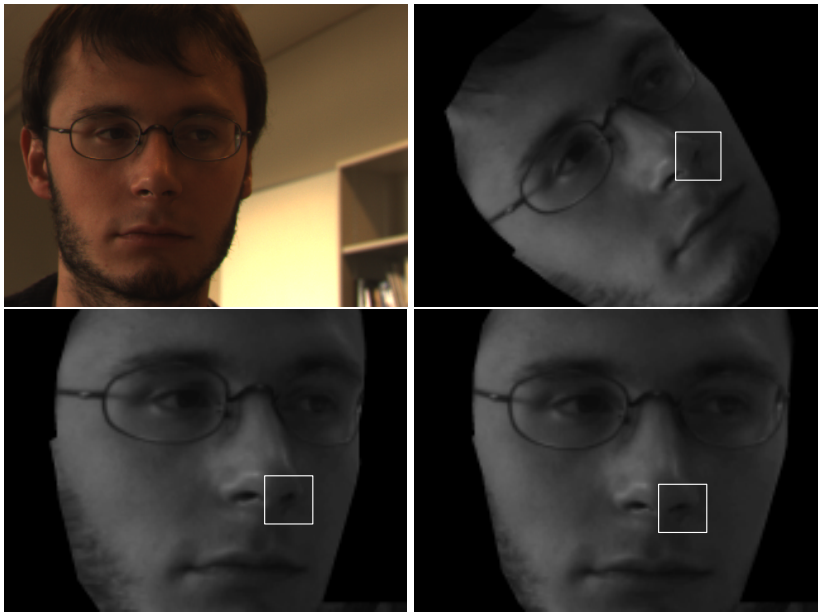


Figure A.1.: The construction of a viewset. Three synthetic views are generated from the original image on the left. The white square represents the patches extracted from these images to build a viewset of a keypoint on the nose.

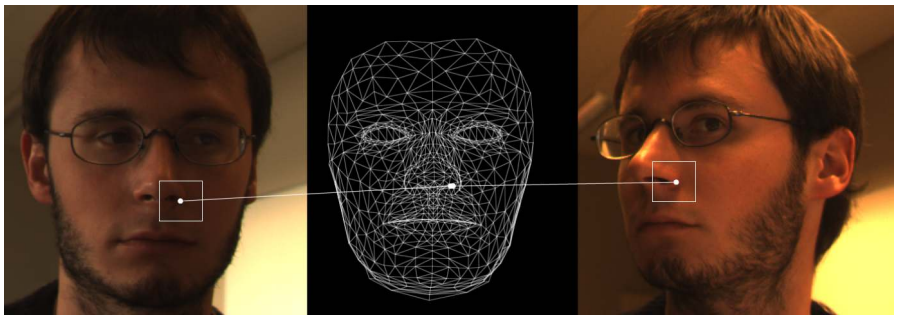


Figure A.2.: Using two different training images to build the viewset of the same keypoint.

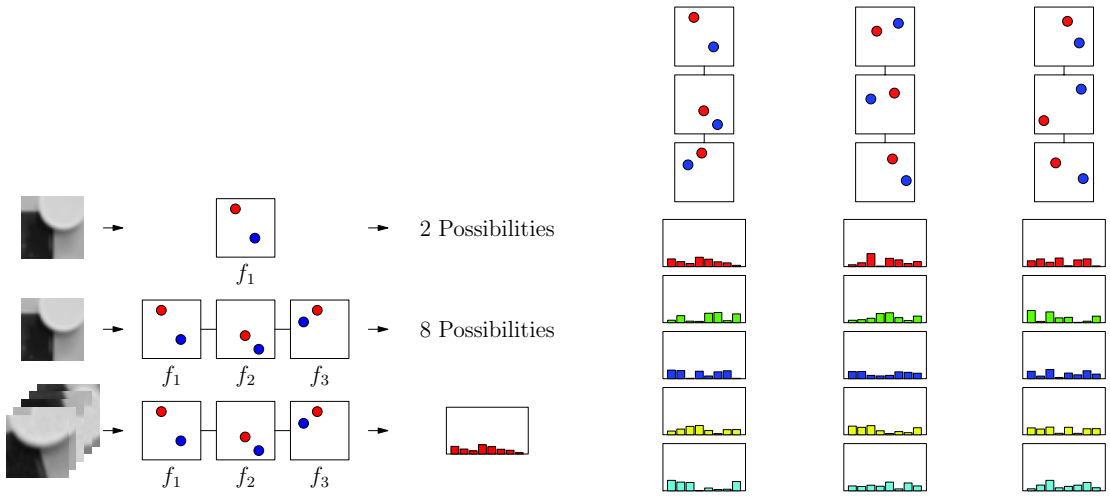


Figure A.3.: (a) For each patch a fern node outputs a binary number, and a fern with S nodes outputs a number between 0 and 2^{S-1}. For multiple patches of the same class, we can model the output of a fern with a multinomial distribution. (b) At the end of training, we have distributions over possible fern outputs for each class. Courtesy of Mustafa Özuysal [80].

responses are then combined in a Naive Bayesian way. We train the classifier by synthesizing many views of the keypoints extracted from a training image as they would appear under different perspective or scale.

Ferns are based on simple binary tests that compare the intensity of two pixels within the patch. The pixels are picked completely at random. A fern is a set of S grouped binary test. Since each test returns either 0 or 1, the fern assigns to an image patch a number between 0 and 2^{S-1}. During a training phase, each patch of a class is tested against several ferns, resulting in distributions that measures how likely this feature point is to obtain this number when tested with this fern (Figure A.3). Recognizing a new patch then amounts to test it against all ferns and to pooling their answers in a Naive Bayesian manner (Figure A.3).

More precisely, the set of all possible appearances of the image patch surrounding a keypoint is treated as a class. Therefore, given the patch surrounding a keypoint detected in an image, the task is to assign it to the most likely class. Let $c_i, i = 1, \ldots, H$ be the set of classes and let $f_j, j = 1, \ldots, N$ be the set of binary features that will be calculated over the patch we are trying to classify. Formally, we are looking for

$$\hat{c_i} = \arg\max_{c_i} P(C = c_i \mid f_1, f_2, \ldots, f_N) \,,$$

where C is a random variable that represents the class. Bayes' Formula yields

$$P(C = c_i \mid f_1, f_2, \ldots, f_N) = \frac{P(f_1, f_2, \ldots, f_N \mid C = c_i)P(C = c_i)}{P(f_1, f_2, \ldots, f_N)} .$$

Assuming a uniform prior $P(C)$, and since the denominator is simply a scaling factor that it is independent from the class, our problem reduces to finding

$$\hat{c_i} = \arg\max_{c_i} P(f_1, f_2, \ldots, f_N \mid C = c_i) . \tag{A.1}$$

The value of each binary feature f_j only depends on the intensities of two pixel locations $\mathbf{d}_{j,1}$ and $\mathbf{d}_{j,2}$ of the image patch. We therefore write

$$f_j = \begin{cases} 1 \text{ if } I(\mathbf{d}_{j,1}) < I(\mathbf{d}_{j,2}) \\ 0 \text{ otherwise} \end{cases} ,$$

where I represents the image patch. Since these features are very simple, many ($N \approx 300$) are required for accurate classification. Therefore a complete representation of the joint probability in Equation (A.1) is not feasible since it would require estimating and storing 2^N entries for each class. To make the problem tractable while accounting for the dependency between features, a good compromise is to partition them into M groups of size $S = \frac{N}{M}$. These groups are called *Ferns* and the joint probability for features in each Fern is computed. The conditional probability becomes

$$P(f_1, f_2, \ldots, f_N \mid C = c_i) = \prod_{k=1}^{M} P(F_k \mid C = c_i) , \tag{A.2}$$

where $F_k = \{f_{\sigma(k,1)}, f_{\sigma(k,2)}, \ldots, f_{\sigma(k,S)}\}, k = 1, \ldots, M$ represents the k^{th} fern and $\sigma(k, j)$ is a random permutation function with range $1, \ldots, N$.

For training, we assume that at least one model image of the object to be detected is available. Training starts by selecting a subset of the keypoints detected on this model image. This is done by deforming the image many times, applying the keypoint detector, and keeping track of the number of times the same keypoint is detected. The keypoints that are found most often are assumed to be the most stable and retained. These stable keypoints are assigned a unique class number. The training set for each class is formed by generating thousands of sample images with randomly picked affine deformations.

The training phase estimates the class conditional probabilities $P(F_m \mid C = c_i)$ for each Fern F_m and class c_i, as described in Equation A.2. For each Fern F_m, these terms are:

$$p_{k,c_i} = P(F_m = k \mid C = c_i) \ , \tag{A.3}$$

where the notation is simplified by considering F_m to be equal to k if the base 2 number formed by concatenating the binary features of F_m is equal to k. With this convention, Ferns can take $K = 2^S$ values and, for each one, we need to estimate the $p_{k,c_i}, k = 1, 2, \ldots, K$ under the constraint

$$\sum_{k=1}^{K} p_{k,c_i} = 1.$$

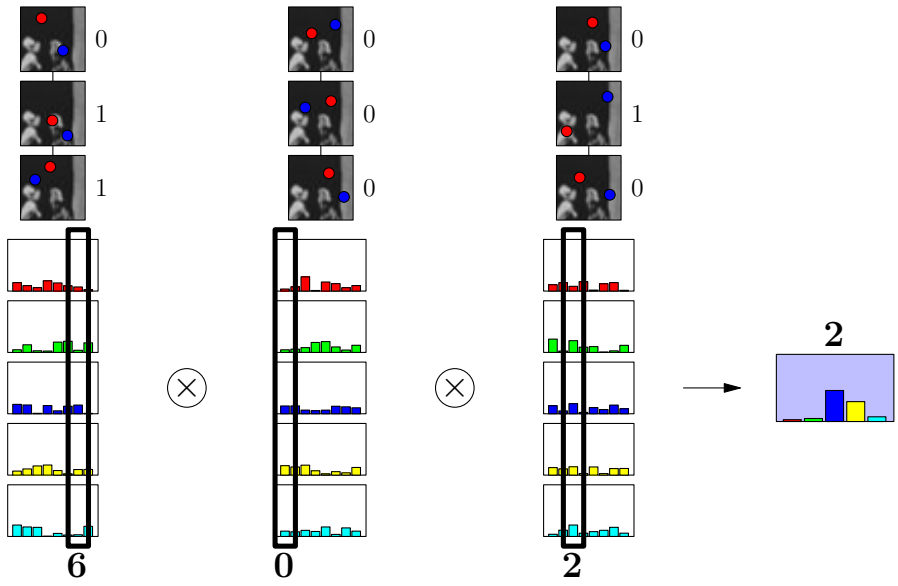


Figure A.4.: To recognize a new patch, the outputs selects rows of distributions for each fern, and these are then combined assuming independence between distributions. Courtesy of Mustafa Özuysal [80].

The simplest approach would be to assign the maximum likelihood estimate to these parameters from the training samples. For parameter p_{k,c_i} it is

$$p_{k,c_i} = \frac{N_{k,c_i}}{N_{c_i}},$$

where N_{k,c_i} is the number of training samples of class c_i that evaluates to Fern value k and N_{c_i} is the total number of samples for class c_i. These parameters can therefore be estimated for each Fern independently. Figure A.3 illustrates the training process.

In practice however, this simple scheme yields poor results because if no training sample for class c_i evaluates to k, which can easily happen when the number of samples is not infinitely large, both N_{k,c_i} and p_{k,c_i} will be zero. Since we multiply the p_{k,c_j} for all Ferns, it implies that, if the Fern evaluates to k, the corresponding patch can *never* be associated to class c_i, no matter the response of the other Ferns. This makes the Ferns far too selective because the fact that $p_{k,c_i} = 0$ may simply be an artifact of the necessarily limited size of the training set. To overcome this problem, p_{k,c_i} is taken to be

$$p_{k,c_i} = \frac{N_{k,c_i} + 1}{N_{c_i} + K}.$$

It amounts to introduce a uniform Dirichlet prior [12] over feature values. If a sample with a specific Fern value is not encountered during training, this scheme will still assign a non-zero value to the corresponding probability.

At runtime, recognition is achieved by computing Equation A.2 for each class and taking the one with the highest probability, as depicted by Figure A.4.

B. Camera Calibration

Previous appendix presented a wide baseline feature matching technique that, combined with RANSAC, can allow detection of a planar object, fixing the 8 DoF of an homography transformation. We explain here how to exploit this detection to calibrate the geometry of one ore more fixed camera, with potentially non-overlapping views. Calibrating cameras is essentially a solved problem. However, the framework of Section 2.4 forbids complex manual operations, or specialized hardware constraints. We present here a method suitable for any unaware user to use.

Before entering into technical details, we shall mention what is camera calibration, why it is useful for augmented reality, and why it takes an important space in this work.

Camera calibration is the process of recovering internal and external geometric properties of the camera. Internal parameters are the focal length, the principal point, the image skew and ratio. For augmented reality purposes, focal length is an important parameter: Rendering a virtual object with a very close virtual camera whose angle of view is large leads to quite a different result than a camera placed far away with a strong zoom. Therefore, 3–D augmented reality requires calibration. Traditional augmented reality software, such as ARToolkit [1], often assumes a universal and arbitrary calibration. By contrast, we propose a user friendly method that allows final users to calibrate their camera.

As detailed in Chapter 5, illuminating virtual objects with real light improves the visual experience of augmented reality. However, in the case of 3–D objects, this process requires a photometric calibration that in turn depends on geometric calibration. Because, augmented reality can be achieved without these visual quality considerations, they are not acceptable at any cost. Users are generally not willing to spend much efforts on calibration. This is why our approach is focused on automating every step rather than seeking for accuracy.

Our geometric calibration procedure includes several stages. Our system first computes homographies between the geometric plane of the target object and its image projections. It then retains the most reliable ones and the corresponding frames to estimate the intrinsic camera parameters and the relative pose of the calibration object with respect to the cameras in the corresponding frames. In turn, these poses are used to select a common referential and to compute the positions and orientations of the cameras with respect to each other. Finally, our system performs global non-linear minimization to refine these estimates. We review related work and outline the individual steps of this process below. Chapter 5 extends it to photometric calibration.

B.1. Related Work

Calibration algorithms that do not require an object known *a priori* are sometimes called *auto-calibration* algorithms. They usually involve a moving camera that is calibrated by simultane-

Figure B.1.: Four static cameras whose calibration is necessary for augmented reality.

ously recovering pose parameters and reconstructing the 3–D structure of the scene. However, they are not well adapted to computing the relative positions of multiple cameras. Furthermore, they lack robustness.

We therefore focus here on methods that rely on a calibration object or pattern because they are much more reliable. Commercially available systems rely on tri-dimensional calibration objects with retro-reflective markers, spheres, or disks on them, which are often complex to build and cumbersome. A notable exception are approaches such as [70] that only involve a moving wand with two markers on it. However, all methods that depend on retro-reflective markers require changing the shutter speed to reliably extract the markers from the images, which prevents simultaneous geometric and photometric calibration. Similarly, using light patterns emitted by a laser pointer as in [105] is a very practical approach to geometric calibration but requires modifying the cameras settings.

Our system falls into the category of recent approaches that rely on a planar target moved in front of the camera [104, 121]. This is attractive because such a target can be built by simply printing a pattern on a sheet of paper. These earlier techniques, however, were only designed to recover the internal parameters of a single camera. Here we are also interested in the positions and orientations of the cameras with respect to each other. The method closest to ours that we are aware of is presented in [111]. As ours, it provides both the intrinsic parameters and relative poses of a multi-camera system using a planar target. However the parameters are estimated via factorization of a matrix built from homographies between image pairs. This requires that *all* the positions of the planar target must be seen from *all* the cameras simultaneously. This is a major limitation that our method does not have.

(a) (b) (c)

Figure B.2.: Some examples of calibration patterns we used to test our system. These reference images serve to train a classifier, As discussed in Appendix A. It is then used to match them against input frames and provide homographies to the geometric calibration process. The images are acquired under diffuse illumination so that the normalized pixel intensities are proportional to the albedo estimates required by the illumination model of Section 5.2.

B.2. Selecting the best Homographies

As explained in the beginning of this chapter, our approach can compute the homography relating an input image and a reference image of a planar object, such as the ones depicted by Figure B.2. This happens when the user waves a calibration pattern in the field of view. Unlike checkerboard-based methods, our approach to estimating the homographies is sufficiently robust not to generate erroneous matches that would have to be removed by hand. What happens in practice is that when the pattern is either occluded or too slanted, it is simply not detected.

However, this is still not quite enough because some of these homographies are inherently ambiguous or singular from a calibration point of view. As shown in Figure B.3, these unreliable homographies come in two flavors. First, the ones that insufficiently distort the pattern are ambiguous and, depending on the noise, may yield wildly different pose estimates. Second, those that distort the pattern too much also produce unreliable pose estimates because the interest points become difficult to locate precisely enough.

To overcome this problem, we define a square in the plane attached to the pattern and measure the angle at each corner after warping it by the homography. If one of the angles is too large or too close to $\frac{\pi}{2}$, the homography is rejected. More formally, a homography $\mathbf{H}$ is rejected if at least one of the angles α of the warped square verifies

$$\begin{aligned}&\cos(\alpha) > \cos(\alpha_0) \text{ or}\\ &\cos(\alpha) < \cos(\pi - \alpha_0) \text{ or}\\ &\cos(\tfrac{\pi}{2} + \alpha_1) < \cos(\alpha) < \cos(\tfrac{\pi}{2} - \alpha_1).\end{aligned}$$

Good results have been achieved using $\alpha_0 = 0.01$ and $\alpha_1 = 0.005$. Remaining homographies are then sorted by number of matches and only the best ones are kept. In practice, we retain around fifty for each camera. As will be shown in Section B.7, this is enough to guarantee

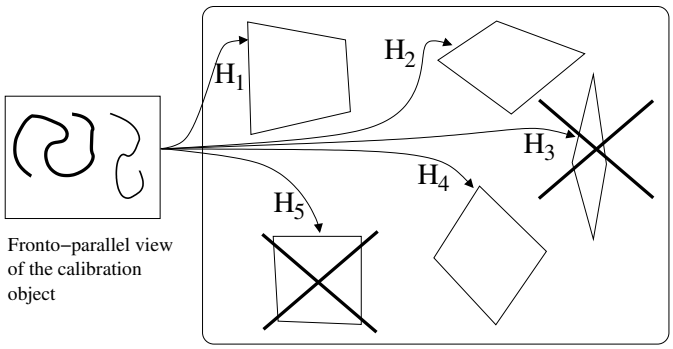


Figure B.3.: Dropping unreliable homographies. Homography $\mathbf{H}_3$ is ignored because, with such a slanted surface, keypoint detection becomes inaccurate and error-prone. $\mathbf{H}_5$ is also discarded because it yields a calibration singularity. The other homographies can be safely kept.

accuracy without imposing an unnecessary computational burden. From an interactive point of view, this selection is done on-line: The user keeps moving the pattern until the system has enough valid homographies.

B.3. Initial Estimation of the Internal Parameters

The internal parameters are first estimated for each camera individually, from the homographies $\mathbf{H}_{c \leftarrow p}$ relating camera c and pose p, using a method similar to the ones of [104, 121]. The computation is quite standard and is described in Section B.8, at the end of this appendix. This yields for each camera c a matrix of internal parameters

$$\mathbf{K}_c = \begin{bmatrix} \tau_c f_c & 0 & u_{0c} \\ 0 & f_c & v_{0c} \\ 0 & 0 & 1 \end{bmatrix}, \tag{B.1}$$

where f_c stands for the focal length, τ_c for the aspect ratio, and $(u_{0c}, v_{0c})^\top$ for the principal point. These internal parameters of all the cameras will then be refined together with the external ones during the non-linear global minimization of Section B.6.

B.4. Initial Estimation of the Poses

We recover the external parameters of each camera in a common referential in two steps. First, our algorithm computes external parameters in a coordinate system attached to the calibration pattern for each frame independently by making use of the internal parameters as estimated previously and the homography related to the frame. It then selects a common referential and computes camera poses in this referential by composing rotations and translations between pairs of frames.

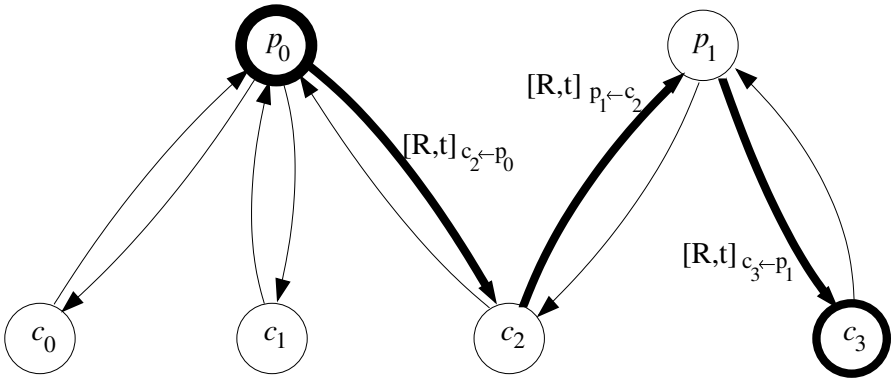


Figure B.4.: The graph corresponding to the network of Figure B.5. The common referential is defined by pose p_0, which is seen by three cameras while p_1 is seen by only two. The pose $[\mathbf{R}, \mathbf{t}]_{c_3}$ of camera c_3 in this referential can be recovered by composing poses with respect to individual cameras. In this case, $[\mathbf{R}, \mathbf{t}]_{c_3} = [\mathbf{R}, \mathbf{t}]_{c_3 \leftarrow p_1} [\mathbf{R}, \mathbf{t}]_{p_1 \leftarrow c_2} [\mathbf{R}, \mathbf{t}]_{c_2 \leftarrow p_0}$.

Displacements between the Calibration Object and the Individual Cameras Given a homography and the intrinsic parameters, we estimate the displacement of the calibration object with respect to the camera as described in the appendix. This step gives us the relative rotation $\mathbf{R}$ and translation $\mathbf{t}$ of the calibration object with respect to the camera. Together, they represent the rigid *displacement* corresponding to pose p as seen from camera c. We write this displacement as the 4×4 matrix

$$[\mathbf{R}, \mathbf{t}]_{c \leftarrow p} = \begin{bmatrix} \mathbf{R}_{c \leftarrow p} & \mathbf{t}_{c \leftarrow p} \\ 0 & 1 \end{bmatrix} . \tag{B.2}$$

The reverse displacement is computed by inverting the matrix which we denote as $[\mathbf{R}, \mathbf{t}]_{p \leftarrow c}$.

B.5. Handling Non-Overlapping Cameras

In practice, the calibration object may never be seen by all cameras simultaneously. Our system therefore selects as a common referential the one attached to the pose of the calibration object seen by the largest number of cameras. It then expresses all the external camera parameters in this referential by composing the displacements of Equation B.2 and their inverses.

Figure B.5 illustrates this behavior. In this case, p_0 provides the common referential because it is seen by three cameras whereas p_1 is seen by only two. Even though camera c_3 does not see p_0, its external parameters in this referential can be estimated by composing the pose of p_0 with respect to camera c_2 with the displacement between cameras c_2 and c_3, which can itself be estimated from the poses of p_1 with respect to these two cameras.

Recovering such chains amounts to compute paths between nodes of a connected graph, which is well understood from an algorithmic viewpoint [45]. More specifically, we define a graph whose nodes correspond to the cameras and the poses of the calibration object, such as the one

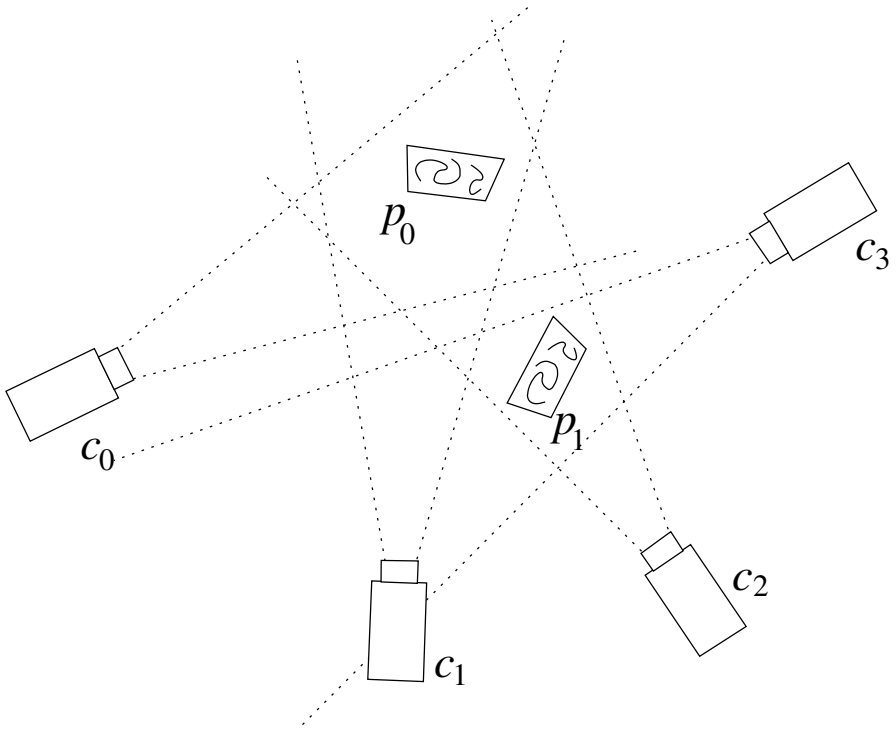


Figure B.5.: A camera network. In this example, the field of view of camera c_3 does not overlap with those of cameras c_0 and c_1. Nevertheless, a full registration is still possible. The displacement between cameras c_2 and c_3 can be estimated when the pattern is in pose p_1, while the relative positions and orientations of cameras c_0, c_1, and c_2 can be estimated using pose p_0. By chaining these estimates, we can express the external camera parameters for all cameras in a common referential.

depicted by Figure B.4. An edge links a camera node and a pose node when the camera sees the pose and is labeled with the corresponding displacement.

Let p_0 denote the pose that defines the common referential, and $[\mathbf{R}, \mathbf{t}]_c$ the pose parameters of camera c in this common referential. Since we have by definition $[\mathbf{R}, \mathbf{t}]_c = [\mathbf{R}, \mathbf{t}]_{c \leftarrow p_0}$, we first look for a path between the p_0 and c nodes, which we write as

$$p_0 \rightarrow c_{\sigma(1)} \rightarrow p_{\sigma(2)} \rightarrow c_{\sigma(3)} \rightarrow \dots p_{\sigma(n)} \rightarrow c \,.$$

where $\sigma(.)$ is a mapping function on the indices that defines the path. Note that the path alternates object pose nodes and camera nodes. It gives us a way to compute $[\mathbf{R}, \mathbf{t}]_c$ from the displacements we just computed since

$$\begin{aligned} [\mathbf{R}, \mathbf{t}]_c &= [\mathbf{R}, \mathbf{t}]_{c \leftarrow p_0} = \\ & [\mathbf{R}, \mathbf{t}]_{c \leftarrow p_{\sigma(n)}} \dots [\mathbf{R}, \mathbf{t}]_{c_{\sigma(3)} \leftarrow p_{\sigma(2)}} [\mathbf{R}, \mathbf{t}]_{p_{\sigma(2)} \leftarrow c_{\sigma(1)}} [\mathbf{R}, \mathbf{t}]_{c_{\sigma(1)} \leftarrow p_0} \,. \end{aligned}$$

The $[\mathbf{R}, \mathbf{t}]_p$ pose parameters of the calibration object can be estimated similarly.

Obviously, this will only work if the graph has one single connected component. In practice, assuming that no camera has a field of view that does not overlap at all any of the others, this is always the case if we move the calibration pattern sufficiently.

B.6. Refining the Estimation

Computing displacements by composing pairwise motions is effective but not particularly accurate. Therefore, to refine not only the pose parameters but also the intrinsic ones, we minimize with respect to all cameras simultaneously the sum of the reprojection errors for the point correspondences used during the detection step of Section B.3. This bundle adjustment is expressed as

$$\sum_{c=1}^{C} \sum_{p=1}^{P} \sum_{k=1}^{M(c,p)} \left\| \begin{pmatrix} u_{c,p,k} \\ v_{c,p,k} \end{pmatrix} - \text{proj}\left(\mathbf{K}_c, [\mathbf{R}, \mathbf{t}]_c [\mathbf{R}, \mathbf{t}]_p^{-1}, \begin{pmatrix} X_{c,p,k} \\ Y_{c,p,k} \\ 0 \end{pmatrix} \right) \right\|^2 , \tag{B.3}$$

where

- C is the number of cameras, P the number of poses of the calibration object, and $M(c,p)$ the number of matches found by the detection stage for camera c and pose p. $M(c,p) = 0$ if pose p of the calibration object is not seen by camera c;
- $[u_{c,p,k}, v_{c,p,k}]^\top$ and $[X_{c,p,k}, Y_{c,p,k}, 0]^\top$ are respectively are a 2–D point and a 3–D point matched by the detection step;
- $\text{proj}(\mathbf{K}, [\mathbf{R}, \mathbf{t}], \mathbf{M})$ returns the projection of 3–D point $\mathbf{M}$ under pose $[\mathbf{R}, \mathbf{t}]$ and internal parameters $\mathbf{K}$.

To increase the robustness of our algorithm, we introduce a simple robust estimator in Equation B.3 to eliminate potentially incorrect correspondences.

	Cam 1	Cam 2	Cam 3	Cam 4	Cam 5
τf	-0.16%	0.32%	0.27%	1.12%	0.56%
f	-0.08%	0.33%	0.19%	1.22%	0.53%
u_0	5.57%	6.49%	9.81%	5.85%	6.00%
v_0	0.11%	0.76%	3.15%	-11.7%	-3.56%

Table B.1.: Recovery of the intrinsic parameters that appear in the $\mathbf{K}_c$ matrix of Equation B.1. We express the differences between those estimated using the first and the second set of sequences as a percentage. In both cases, we used 200 homographies. The percentages for the focal lengths are very small. Those corresponding to the principal points are a bit larger, which is not surprising given the fact that they are known to have much less influence on the projection matrix.

B.7. Calibration Accuracy

To test the accuracy of the camera parameters that our system recovers, we first trained a classifier to recognize the interest points of the pattern of Figure B.2(a), as discussed in Appendix A. We then used a 5 cameras setup to record two different sets of video sequences by waving the pattern in front of the cameras. Finally, we performed the calibration independently for each set. Fig B.7 shows a typical frame with the detected pattern draw as a wireframe box.

Figure B.6(a) depicts the focal lengths recovered using the first set of sequences. They are shown as a function of the total number of homographies retained to perform the computation. When this number climbs above 140, the estimates become quite stable. It therefore does not make sense to use many more since the computational cost increases almost linearly with this number, as shown in Figure B.6(b).

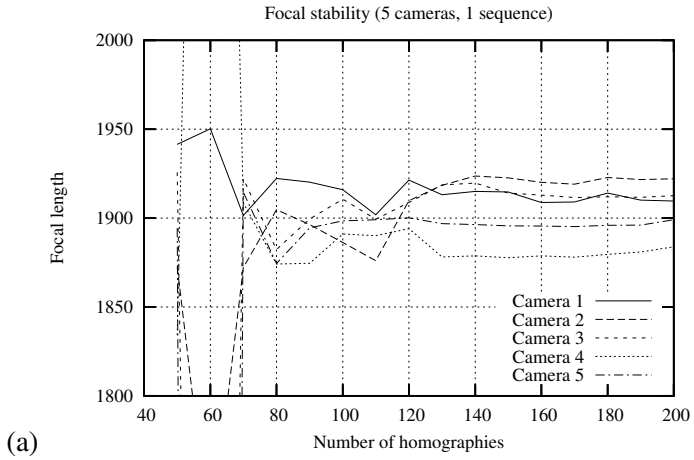
In Figure B.6(c) and Table B.1 we compare the results obtained independently using the two sets of video sequences. In Figure B.6(c), we superpose the focal length estimates, again drawn as a function of the total number of homographies retained. As before, once we use more than 140, the two estimates become very close. This is a very good indication that they are accurate since they were computed independently. As shown by Table B.1, this is true not only for the focal lengths but also for the principal point locations.

We have not obtained ground truth for the external camera parameters. However, the virtual object appears to be very stable with respect to the calibration pattern, which would not be the case if they were poorly recovered.

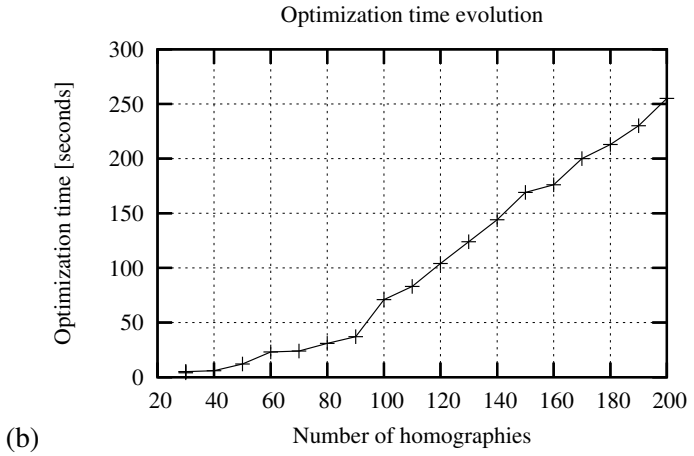
B.8. Internal Parameters from a Set of Homographies

The computation of internal parameters from a set of homographies is quite standard [104, 121], but we give it here for the sake of completeness. First, we write the matrix of internal parameters as

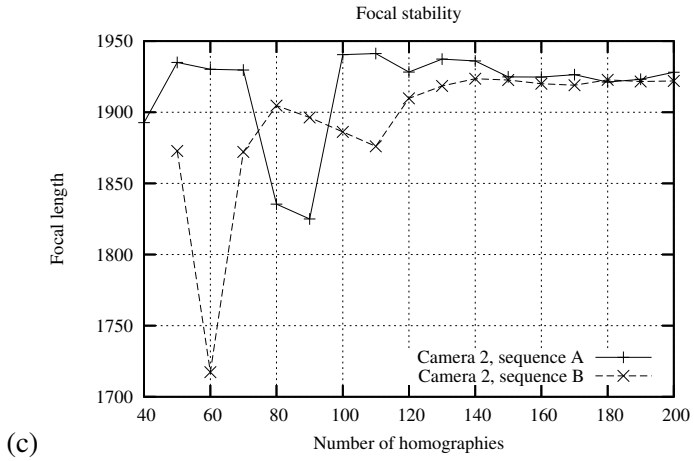
$$\mathbf{K} = \begin{bmatrix} \tau f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} ,$$



(a)



(b)



(c)

Figure B.6.: Geometric calibration of 5 cameras. (a) Focal lengths estimated for each camera as a function of the total number of homographies retained. The values stabilize once enough homographies are used. (b) The computational cost grows linearly as a function of the number of homographies. (c) Focal length of the second of 5 calibrated cameras computed independently using two different sets of sequences. Once enough homographies are used, the estimates become very close.

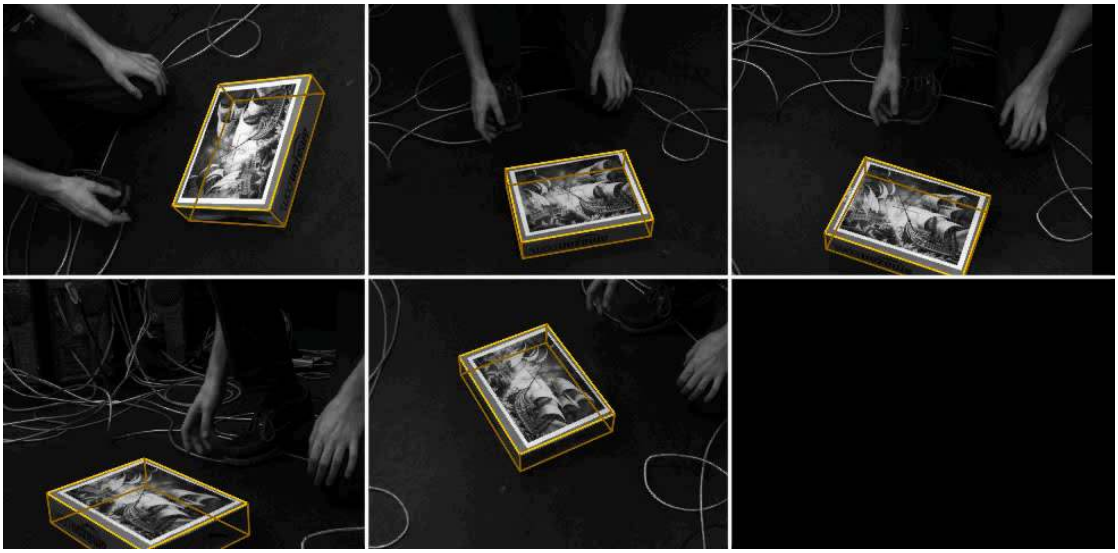


Figure B.7.: 3–D detection of a pattern using five calibrated cameras. All images are taken synchronously. This frame is part of the calibration sequence used to obtain the results of Fig B.6. The wireframe box shows that the 3–D calibration pattern reprojects accurately on the five cameras.

where f represents the focal length, τ the aspect ratio, and $(u_0, v_0)^\top$ the principal point. We want to estimate $\mathbf{K}$ from a set of homographies that map points on a planar object to points on captured images for different object positions.

We associate to each camera a projection matrix

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R} \mid \mathbf{t}]\ , \tag{B.4}$$

where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ a translation vector. Without loss of generality, we can choose $Z = 0$ as the plane of our calibration pattern. The relation between $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{t}$ and the homography $\mathbf{H}$ that maps a point $\mathbf{M} = [X, Y, 0]^\top$ of this plane to its corresponding 2–D point $\mathbf{m}{=}[u, v]^\top$ under perspective projection can be written as

$$\left[\ \mathbf{m}1\ \right] \propto \mathbf{P} \begin{bmatrix} \mathbf{M} \\ 1 \end{bmatrix} = \mathbf{K}\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3\ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{K}\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}\ , \tag{B.5}$$

where $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ respectively are the first, second and third column of the rotation matrix $\mathbf{R}$, and the symbol $\propto$ denotes proportionality. The homography projecting the plane $Z = 0$ to the image plane is then: $\mathbf{H} \propto \mathbf{K}\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$. For convenience, we introduce a 3×3 matrix $\mathbf{T}$ with

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & \mathbf{t}'_1 \\ 0 & 1 & \mathbf{t}'_2 \\ 0 & 0 & \mathbf{t}'_3 \end{bmatrix}\ ,$$

where $\mathbf{t}' = \mathbf{R}^{-1}\mathbf{t} = \mathbf{R}^\top\mathbf{t}$ that lets us write

$$\mathbf{H} \propto \mathbf{KRT} \,. \tag{B.6}$$

To find the relations between the coefficients of $\mathbf{H}$ and the internal parameters, let us now compute the product $\mathbf{H}^\top\omega\mathbf{H}$ where ω is the matrix $(\mathbf{K}\mathbf{K}^\top)^{-1}$, also known as the image of the absolute conic. We have

$$\begin{aligned}\mathbf{H}^\top\omega\mathbf{H} = \mathbf{H}^\top\mathbf{K}^{-\top}\mathbf{K}^{-1}\mathbf{H} &\propto (\mathbf{KRT})^\top\mathbf{K}^{-\top}\mathbf{K}^{-1}(\mathbf{KRT}) \\ &= \mathbf{T}^\top\mathbf{R}^\top\mathbf{K}^\top\mathbf{K}^{-\top}\mathbf{K}^{-1}\mathbf{KRT} \\ &= \mathbf{T}^\top\mathbf{T} = \begin{bmatrix} 1 & 0 & -\mathbf{t}'_1 \\ 0 & 1 & -\mathbf{t}'_2 \\ -\mathbf{t}'_1 & -\mathbf{t}'_2 & \|\mathbf{t}'\|^2 \end{bmatrix} .\end{aligned}$$

By considering the expressions of the elements of the 2×2 sub-matrix on the top-left of the $\mathbf{T}^\top\mathbf{T}$ matrix, we obtain the two following equations:

$$\begin{cases} \mathbf{h}_1^\top\omega\mathbf{h}_1 - \mathbf{h}_2^\top\omega\mathbf{h}_2 = 0 \\ \mathbf{h}_1^\top\omega\mathbf{h}_2 = 0 \end{cases} , \tag{B.7}$$

where $\mathbf{h}_i$ represents the column i of $\mathbf{H}$. It follows that:

$$\omega \propto \begin{bmatrix} 1 & 0 & -u_0 \\ 0 & \tau^2 & -\tau^2 v_0 \\ -u_0 & -\tau^2 v_0 & \tau^2 f^2 + u_0^2 + \tau^2 v_0^2 \end{bmatrix} . \tag{B.8}$$

By rearranging the terms of Equation B.7 and Equation B.8, we obtain the following linear system in some of the coefficients of ω:

$$\mathbf{AW} = \mathbf{h} \tag{B.9}$$

where

$$\mathbf{A} = \begin{bmatrix} 2(h_{11}h_{31} - h_{12}h_{32}) & h_{21}^2 - h_{22}^2 & 2(h_{21}h_{31} - h_{22}h_{32}) & h_{31}^2 - h_{32}^2 \\ h_{11}h_{32} + h_{12}h_{31} & h_{22}h_{21} & h_{32}h_{21} + h_{22}h_{31} & h_{32}h_{31} \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} \omega_{13} \\ \omega_{22} \\ \omega_{23} \\ \omega_{33} \end{bmatrix} \quad \text{and} \quad \mathbf{h} = \begin{bmatrix} h_{11}^2 - h_{12}^2 \\ h_{11}h_{12} \end{bmatrix} .$$

Each homography yields such a pair of equations. For each camera, this produces an over-constrained system that we solve in the least-squares sense. The internal parameters u_0, v_0, f, and τ can then be estimated from ω_{13}, ω_{22}, ω_{23}, and ω_{33}. In our implementation, they are refined by the final non-linear optimization.

B.9. Displacement between a Camera and a Planar Object from a Set of Homographies and the Internal Parameters

Once the internal camera parameters have been recovered from the whole sequence we can recover the rotation R and the translation t between a particular pose of the planar object and the camera as follows.

Recall from Equation B.6 that $\mathbf{H} \propto \mathbf{KRT}$, where $\mathbf{R}$ and $\mathbf{t}$ are expressed in a coordinate system attached to the planar object. Equivalently, we write:

$$\mathbf{RT} \propto \mathbf{K}^{-1}\mathbf{H} \, .$$

Since the columns of R should have a norm equal to 1, the scale factor can be retrieved, and a first estimation of these columns is obtained as:

$$\mathbf{r}_1 = \frac{\mathbf{K}^{-1}\mathbf{h}_1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|}, \quad \mathbf{r}_2 = \frac{\mathbf{K}^{-1}\mathbf{h}_2}{\|\mathbf{K}^{-1}\mathbf{h}_2\|}, \quad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \quad ,$$

where $\times$ denotes the cross-product of two vectors. Because the homographies are noisy, the resulting rotation matrix is not orthonormal and we correct it using the procedure given in the appendix of [121] which seeks the closest orthonormal matrix in the Frobenius sense using a Singular Value Decomposition. Similarly the translation vector $\mathbf{t}$ can be approximated as:

$$\mathbf{t} = \frac{2\mathbf{K}^{-1}\mathbf{h}_3}{\|\mathbf{K}^{-1}\mathbf{h}_1\| + \|\mathbf{K}^{-1}\mathbf{h}_2\|} \, .$$

Bibliography

[1] Artoolkit. http://www.hitl.washington.edu/artoolkit/.

[2] M. Bajura, H. Fuchs, and R. Ohbuchi. Merging virtual objects with the real world: seeing ultrasound imagery within the patient. *ACM SIGGRAPH*, pages 203–210, July 1992.

[3] S. Baker, I. Matthews, J. Xiao, R. Gross, T. Kanade, and T. Ishikawa. Real-time non-rigid driver head tracking for driver mental state estimation. In *World Congress on Intelligent Transportation Systems*, Oct. 2004.

[4] A. Bartoli, E. von Tunzelmann, and A. Zisserman. Augmenting images of non-rigid scenes using point and curve correspondences. In *Conference on Computer Vision and Pattern Recognition*, June 2004.

[5] A. Bartoli and A. Zisserman. Direct Estimation of Non-Rigid Registration. In *British Machine Vision Conference*, Kingston, UK, September 2004.

[6] A. Baumberg. Reliable Feature Matching across Widely Separated Views. In *Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.

[8] M. Berger. Resolving Occlusion in Augmented Reality: a Contour Based Approach without 3D Reconstruction. In *Conference on Computer Vision and Pattern Recognition*, page 91, 1997.

[9] C. Bichlmeier, F. Wimmer, S. Heining, and N. Navab. Contextual Anatomic Mimesis: Hybrid In-Situ Visualization Method for Improving Multi-Sensory Depth Perception in Medical Augmented Reality. In *International Symposium on Mixed and Augmented Reality*, pages 129–138, Nov. 2007.

[10] O. Bimber, F. Coriand, A. Kleppe, E. Bruns, S. Zollmann, and T. Langlotz. Superimposing pictorial artwork with projected imagery. *IEEE MultiMedia*, 12(1):16–26, 2005.

[11] O. Bimber, A. Grundhofer, G. Wetzstein, and S. Knodel. Consistent illumination within optical see-through augmented environments. In *International Symposium on Mixed and Augmented Reality*, pages 198–207, 7-10 Oct. 2003.

[12] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *European Conference on Computer Vision*, pages 329–342, 1996.

[14] V. Blanz and T. Vetter. A Morphable Model for The Synthesis of 3–D Faces. In *ACM SIGGRAPH*, pages 187–194, Los Angeles, CA, August 1999.

[15] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *International Conference on Computer Vision*, pages Vol I:105–112, Vancouver, Canada, July 2001.

[16] D. E. Breen, R. T. Whitaker, E. Rose, and M. Tuceryan. Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):11–22, 1996.

[17] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst.*, 89(2-3):114–141, 2003.

[18] L. Cohen and I. Cohen. Deformable models for 3-d medical images using finite elements and balloons. In *Conference on Computer Vision and Pattern Recognition*, pages 592–598, 1992.

[19] T. Cootes, G. Edwards, and C. Taylor. Active Appearance Models. In *European Conference on Computer Vision*, pages 484–498, Freiburg, Germany, June 1998.

[20] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, January 1995.

[21] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6), June 2001.

[22] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *Conference on Computer Vision and Pattern Recognition*, pages 53–60, 2006.

[23] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH*, July 1998.

[24] D. DeCarlo and D. Metaxas. Deformable Model-Based Shape and Motion Analysis from Images using Motion Residual Error. In *International Conference on Computer Vision*, pages 113–119, Bombay, India, 1998.

[25] H. Delingette, M. Hebert, and K. Ikeuchi. Deformable surfaces: A free-form shape representation. In *SPIE Geometric Methods in Computer Vision*, volume 1570, pages 21–30, 1991.

[26] G. Dewaele, F. Devernay, and R. Horaud. Hand motion from 3d point trajectories and a smooth surface model. In *European Conference on Computer Vision*, pages 495–507, May 2004.

[27] G. Drettakis, L. Robert, and S. Bougnoux. Interactive common illumination for computer augmented reality. In *Eurographics*, pages 45–56, June 1997.

[28] J. Ehara and H. Saito. Texture overlay for virtual clothing based on pca of silhouettes. In *International Symposium on Mixed and Augmented Reality*, pages 139–142, 2006.

[29] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density for visual surveillance. In *Proceedings of the IEEE*, volume 90, pages 1151–1163, July 2002.

[30] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Dynamic Free-Form Deformations for Animation Synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 1997.

[31] V. Ferrari, T. Tuytelaars, and L. V. Gool. Simultaneous Object Recognition and Segmentation by Image Exploration. In *European Conference on Computer Vision*, May 2004.

[32] M. Fiala. Magic mirror system with hand-held and wearable augmentations. *IEEE Virtual Reality Conference*, pages 251–254, 2007.

[33] G. D. Finlayson, M. S. Drew, and C. Lu. Intrinsic images by entropy minimization. In *European Conference on Computer Vision*, pages 582–595, May 2004.

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

[35] J. Fisher and D. Bartz. Utilizing image guided surgery for user interaction in medical augmented reality. Technical Report WSI-2005-04, University of Tbingen, 2005.

[36] R. Fransens, C. Strecha, and L. Van Gool. A mean field EM-algorithm for coherent occlusion handling in map-estimation problems. In *Conference on Computer Vision and Pattern Recognition*, 2006.

[37] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. In *Annual Conference on Uncertainty in Artificial Intelligence*, pages 175–181, 1997.

[38] P. Fua and Y. G. Leclerc. Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision*, 16:35–56, September 1995.

[39] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(8):930–943, 2003.

[40] V. Gay-Bellile, A. Bartoli, and P. Sayd. Direct estimation of non-rigid registrations with image-based self-occlusion reasoning. In *International Conference on Computer Vision*, Rio, Brazil, October 2007.

[41] S. Gibson and A. Murta. Interactive Rendering with Real-World Illumination. In *Eurographics Workshop on Rendering*, June 2000.

[42] G. Gordon, M. Billinghurst, M. Bell, J. Woodfill, B. Kowalik, and A. Erendi. The use of dense stereo range data in augmented reality. In *International Symposium on Mixed and Augmented Reality*, Los Alamitos, USA, Sept. 2002.

[43] S. Granger and X. Pennec. Multi-scale em-icp: A fast and robust approach for surface registration. In *European Conference on Computer Vision*, pages 418–432, Copenhagen, Denmark, 2002.

[44] W. Grimson, G. Ettinger, S. White, T. Lozano-Pérez, W. W. III, and R. Kikinis. An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality visualization. *IEEE Transactions on Medical Imaging*, 15(2):129–140, 1999.

[45] J. Gross and J. Yellen. *Graph theory and its applications*. CRC Press, Inc., Boca Raton, FL, USA, 1999.

[46] N. Gumerov, A. Zandifar, R. Duraiswami, and L. Davis. Structure of Applicable Surfaces from Single Views. In *European Conference on Computer Vision*, Prague, May 2004.

[47] I. Guskov. Kernel-based template alignment. In *Conference on Computer Vision and Pattern Recognition*, New York, NY, June 2006.

[48] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[49] M. Heikkila and M. Pietikainen. A texture-based method for modeling the background and detecting moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):657–662, April 2006.

[50] A. Hofhauser, C. Steger, and N. Navab. Harmonic deformation model for edge based template matching. In *International Conference on Computer Vision Theory and Applications*, pages 75–82, 2008.

[51] P. Huber. *Robust Statistics*. Wiley, New York, 1981.

[52] S. Ilić, M. Salzmann, and P. Fua. Implicit Meshes for Effective Silhouette Handling. *International Journal of Computer Vision*, 72(7), 2007.

[53] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld. Detection and location of people in video images using adaptive fusion of color and edge information. In *International Conference on Pattern Recognition*, pages 627–630 vol.4, 2000.

[54] A. Kambhamettu, D. Goldgoff, D. Terzopoulos, and T. Huang. *Handbook of Pattern Recognition and Image Processing: Computer Vision*, chapter Non Rigid Motion Analysis, pages 405–430. Academic Press, 1994.

[55] M. Kanbara and N. Yokoya. Real-Time Estimation of Light Source Environment for Photorealistic Augmented Reality. In *International Conference on Pattern Recognition*, 2004.

[56] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[57] G. Klein and T. Drummond. Robust visual tracking for non-instrumented augmented reality. In *International Symposium on Mixed and Augmented Reality*, pages 36–45, October 2003.

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

[59] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[60] V. Lepetit and M.-O. Berger. An intuitive tool for outlining objects in video sequences : Applications to augmented and diminished reality. In *Proceedings of International Symposium of Mixed Reality, Yokohama (Japan)*, March 2001.

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

[62] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, Sept. 2006.

[63] V. Lepetit, P. Lagger, and P. Fua. Randomized Trees for Real-Time Keypoint Recognition. In *Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[64] V. Lepetit, J. Pilet, and P. Fua. Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation. In *Conference on Computer Vision and Pattern Recognition*, Washington, DC, June 2004.

[65] V. Lepetit, J. Pilet, A. Geiger, A. Mazzoni, M. Oezuysal, and P. Fua. Bazar. http://cvlab.epfl.ch/software/bazar.

[66] W.-C. Lin and Y. Liu. Tracking dynamic near-regular textures under occlusion and rapid movements. In *European Conference on Computer Vision*, May 2006.

[67] X. Llado, A. D. Bue, and L. Agapito. Non-rigid 3D Factorization for Projective Reconstruction. In *British Machine Vision Conference*, Oxford, UK, September 2005.

[68] C. Loscos, G. Drettakis, and L. Robert. Interactive virtual relighting of real scenes. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):289–305, 2000.

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

[70] H. Maas. Image sequence based automatic multi-camera system calibration techniques. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(5-6):352–359, December 1999.

[71] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[72] S. R. Marschner and D. P. Greenberg. Inverse lighting for photography. In *Proceedings of the Fifth Color Imaging Conference, Society for Imaging Science and Technology*, 1997.

[73] I. Matthews and S. Baker. Active Appearance Models Revisited. *International Journal of Computer Vision*, 60:135–164, November 2004.

[74] T. McInerney and D. Terzopoulos. A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4d image analysis. *Computerized Medical Imaging and Graphics*, 19(1):69–83, 1995.

[75] D. Metaxas and D. Terzopoulos. Constrained deformable superquadrics and nonrigid motion tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993.

[76] K. Mikolajczyk and C. Schmid. An Affine Invariant Interest Point Detector. In *European Conference on Computer Vision*, pages 128–142. Springer, 2002. Copenhagen.

[77] L. Moccozet and N. Magnenat-Thalmann. Dirichlet Free-Form Deformation and their Application to Hand Simulation. In *Computer Animation*, 1997.

[78] E. Nakamae, K. Harada, T. Ishizaki, and T. Nishita. A montage method: the overlaying of the computer generated images onto a background photograph. *ACM SIGGRAPH*, 20(4):207–214, 1986.

[79] S. Nicolau, X. Pennec, L. Soler, and N. Ayache. A complete augmented reality guidance system for liver punctures: First clinical evaluation. In *Proceedings of the 8th Int. Conf. on Medical Image Computing and Computer-Assisted Intervention - MICCAI 2005, Part I*, volume 3749 of *LNCS*, pages 539–547, Palm Springs, CA, USA, October 26-29, 2005. Springer Verlag.

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

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

[82] A. Pentland. Automatic extraction of deformable part models. *International Journal of Computer Vision*, 4(2):107–126, 1990.

[83] J. Pilet, A. Geiger, P. Lagger, V. Lepetit, and P. Fua. An All-In-One Solution to Geometric and Photometric Calibration. In *International Symposium on Mixed and Augmented Reality*, Santa Barabara, CA, October 2006.

[84] J. Pilet, V. Lepetit, and P. Fua. Real-Time Non-Rigid Surface Detection. In *Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[85] A. Prati, I. Mikic, M. Trivedi, and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:918–923, 2003.

[86] L. Quan and Z. Lan. Linear N-Point Camera Pose Determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(7):774–780, jul 1999.

[87] G. Reitmayr and T. Drummond. Initialisation for visual tracking in urban environments. In *International Symposium on Mixed and Augmented Reality*, 2007.

[88] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *International Conference on Computer Vision*, Beijing, China, October 2005.

[89] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH*, 2004.

[90] M. Salzmann, S. Ilić, and P. Fua. Physically Valid Shape Parameterization for Monocular 3–D Deformable Surface Tracking. In *British Machine Vision Conference*, Oxford, UK, September 2005.

[91] M. Salzmann, F. Moreno-Noguer, V. Lepetit, and P. Fua. Closed-form solution to non-rigid 3d surface detection. In *European Conference on Computer Vision*, Marseille, France, October 2008.

[92] M. Salzmann, J. Pilet, S. Ilić, and P. Fua. Surface Deformation Models for Non-Rigid 3–D Shape Recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1481–1487, August 2007.

[93] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a Radiance Distribution to Superimpose Virtual Objects onto a Real Scene. *IEEE Transactions on Visualization and Computer Graphics*, 1999.

[94] F. Schaffalitzky and A. Zisserman. Multi-View Matching for Unordered Image Sets, or "How Do I Organize My holiday Snaps?". In *Proceedings of European Conference on Computer Vision*, pages 414–431, 2002.

[95] C. Scherrer, J. Pilet, P. Fua, and V. Lepetit. The Haunted Book. In *International Symposium on Mixed and Augmented Reality*, Cambridge, England, 2008.

[96] J. Schmidt, H. Niemann, and S. Vogt. Dense disparity maps in real-time with an application to augmented reality. In *IEEEWorkshop on Applications of Computer Vision*, pages 225–230, Orlando, USA, December 2002.

[97] S. Sclaroff and J. Isidoro. Active blobs: region-based, deformable appearance models. *Computer Vision and Image Understanding*, 89(2-3), 2003.

[98] T. Sederberg and S. Parry. Free-Form Deformation of Solid Geometric Models. *ACM SIGGRAPH*, 20(4), 1986.

[99] A. Shahrokni, F. Fleuret, T. Drummond, and P. Fua. Probabilistic Modeling of Texture Transition for Fast Tracking and Delineation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. In preparation.

[100] Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1778–1792, 2005.

[101] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic Tracking of 3D human Figures using 2D Image Motion. In *European Conference on Computer Vision*, June 2000.

[102] J. Stauder, R. Mech, and J. Ostermann. Detection of moving cast shadows for object segmentation. *IEEE Transactions on Multimedia*, 1(1):65–76, 1999.

[103] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Conference on Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[104] P. Sturm and S. Maybank. On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications. In *Conference on Computer Vision and Pattern Recognition*, pages 432–437, June 1999.

[105] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[106] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:703–714, 1991.

[107] L. Torresani, A. Hertzmann, and C. Bregler. Learning non-rigid 3d shape from 2d motion. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2003.

[108] L. Torresani, D. B. Yang, E. J. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *Conference on Computer Vision and Pattern Recognition*, pages 493–500, 2001.

[109] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *International Conference on Computer Vision*, pages 255–261 vol.1, 1999.

[110] T. Tuytelaars and L. V. Gool. Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions. In *British Machine Vision Conference*, pages 412–422, 2000.

[111] T. Ueshiba and F. Tomita. Plane-based calibration algorithm for multi-camera systems via factorization of homography matrices. In *International Conference on Computer Vision*, pages 966–973, 2003.

[112] L. Vacchetti, V. Lepetit, and P. Fua. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *International Symposium on Mixed and Augmented Reality*, Arlington, VA, November 2004.

[113] V. Vlahakis, N. Ioannidis, J. Karigiannis, M. Tsotros, M. Gounaris, D. Stricker, T. Gleue, P. Daehne, and L. Almeida. Archeoguide: An augmented reality guide for archaeological sites. *IEEE Computer Graphics and Applications*, 22(5):52–60, 2002.

[114] D. Wagner. *Handheld Augmented Reality*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, Inffeldgasse 16a/2nd floor, A-8010 Graz, Austria, 2007.

[115] D. Wagner, T. Langlotz, and D. Schmalstieg. Robust and Unobtrusive Marker Tracking on Mobile Phones. In *International Symposium on Mixed and Augmented Reality*, Nara, Japan, Sept. 2007.

[116] R. White and D. Forsyth. Retexturing single views using texture and shading. In *European Conference on Computer Vision*, volume LNCS 3954, pages 70–81, 2006.

[117] J. Wills and S. Belongie. A feature-based approach for determining long range optical flow. In *European Conference on Computer Vision*, Prague, Czech Republic, May 2004.

[118] M. M. Wloka and B. G. Anderson. Resolving occlusion in augmented reality. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 5–12, Monterey, California, United States, 1995.

[119] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. In *Photonics East, SPIE*, volume 2615, 1995.

[120] J. Xiao, J.-X. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. In *European Conference on Computer Vision*, pages 573–587, 2004.

[121] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, 2000.

[122] Z. Zhang, R. Deriche, O. Faugeras, and Q. Luong. A Robust Technique for Matching two Uncalibrated Images through the Recovery of the Unknown Epipolar Geometry. *Artificial Intelligence*, 78:87–119, 1995.

[123] J. Zhu and M. R. Lyu. Progressive finite newton approach to real-time nonrigid surface detection. In *International Conference on Computer Vision*, Rio, Brazil, October 2007.

[124] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.

Curriculum Vitae

Julien Pilet
EPFL CVLAB, Station 14, CH-1015 Lausanne, Switzerland
E-mail: julien.pilet@epfl.ch
Born July 6, 1979, Switzerland

Education

2003-2008 Ph.D. candidate in Computer Vision. Advisor: Pascal Fua.
Computer Vision Lab, Ecole Polytechnique Fédérale de Lausanne (EPFL)
Title: Augmented Reality for non-rigid surfaces

1998-2003 M.S. and B.S. in Computer Science
Ecole Polytechnique Fédérale de Lausanne (EPFL)

1995-1998 Swiss Federal Maturity, Latin and ancient Greek, Lausanne

Research and Development Experience

Augmented Reality Illumination and occlusion aware retexturing. 2007-2008.

Camera Calibration Automatic photometric and geometric multi-camera calibration.. Sources available under General Public License (GPL). 2006.

Non-rigid Surfaces Research on fast non-rigid surface detection. Best Paper award CVPR 2005.

Registration Implementation of a vision based tool for measuring mast twist. Used on board of the sailing boat 'Alinghi', winner of the America's Cup. 2003-2004.

PCB Design Development of *Armonie*, an embedded system based on the Intel XScale PXA255. 2002-2003.

Cross-Development Implementation of cross-development tools (JTAG cross-debugger). M.S. Thesis in collaboration with Stéphane Magnenat. 2002-2003.

Linux Kernel Ported Linux on *Armonie*, including development of a Linux boot loader. 2002-2003.

Teaching Experience

Lecture Foundation of Image Science. 7x2 hours, 30 students, 2006/2007.

Exercises Foundation of Image Science, 3x14 hours, 2005-2006, 2006-2007, 2007-2008.

Assistant for exercises of several classes: C/C++ programming, Embedded development.

Journal Publications

2007 *Fast non-rigid surface detection, registration and realistic augmentation*, **Julien Pilet**, Vincent Lepetit, and Pascal Fua, International Journal of Computer Vision.

2007 *Surface deformation models for non-rigid 3–d shape recovery*, Mathieu Salzmann, **Julien Pilet**, Slobodan Ilic, and Pascal Fua, IEEE Transactions on Pattern Analysis and Machine Intelligence.

Conference Publications

2008 *The Haunted Book*, Camille Scherrer, **Julien Pilet**, Pascal Fua, and Vincent Lepetit, International Symposium on Mixed and Augmented Reality, Cambridge, UK.

2008 *Making Background Subtraction Robust to Sudden Illumination Changes*, **Julien Pilet**, Christoph Strecha, and Pascal Fua, European Conference on Computer Vision, Marseille, France.

2007 *Retexturing in the presence of complex illumination and occlusions*, **Julien Pilet**, Vincent Lepetit, and Pascal Fua, International Symposium on Mixed and Augmented Reality, Nara, Japan.

2006 *An all-in-one solution to geometric and photometric calibration*, **Julien Pilet**, Andreas Geiger, Pascal Lagger, Vincent Lepetit, and Pascal Fua, International Symposium on Mixed and Augmented Reality, Santa Barbara, CA, USA.

2005 *Augmenting deformable objects in real-time*, **Julien Pilet**, Vincent Lepetit, and Pascal Fua, International Symposium on Mixed and Augmented Reality, Vienna, Austria.

2005 *Real-time non-rigid surface detection*, **Julien Pilet**, Vincent Lepetit, and Pascal Fua. Best Paper Award, Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA.

2004 *Point matching as a classification problem for fast and robust object pose estimation*, Vincent Lepetit, **Julien Pilet**, and Pascal Fua., Conference on Computer Vision and Pattern Recognition, Washington, DC, USA.

Award

Best Paper Award Computer Vision and Patern Recognition 2005, San Diego, CA.

Technical Skills

Programming Excellent skills: C++, C
Knowledge: C#, Java, PHP, Small Talk, Ada, Lisp, Pict, Funnel, ...
Assembly: ARM, x86 (including MMX/SSE). GPU programming.

Linux kernel Porting on an embedded platform and basic driver programming.

Operating systems Development experience with GNU/Linux, Microsoft Windows, Sun Solaris, SGI Irix, MacOS X, FreeBSD and MS-DOS.

IC design FPGA experience. Tools: VHDL, ModelSim, Leonardo.

PCB design Intel XScale based embedded system development.

Supervised Projects

2007 *Video-based foil immersion measurement for the Hydroptère*, Guillaume Bonnier, Internship.

2007 *Depth of field calibration for Augmented Reality*, Michael Desboeufs, Master project.

2006-2007 *Graphical user interface for a video-based Spinnaker analysis tool*, André Mazzoni, Commissioned development. Software written for and **used by Alinghi**, winner of the America's Cup.

2006 *Automatic multiple camera calibration*, Andreas Geiger, Student project.

2005 *Détection de surfaces déformables grâce aux contours multirésolution*, Stephane Decoppet, Master project.

2004 *3D Reconstruction for Augmented Reality*, Andre Mazzoni, Student project.

Languages

French Mother tongue

English Fluent

German Average

Japanese Currently learning

Spanish Basic knowledge

Latin, ancient Greek Resp. 7 and 6 years of studies