RONDA
Reconstruction of Non-rigid Surfaces from High Resolution Video

Marc Habermann
Saarland University
Saarbrücken, Germany

Submitted for the Degree of Master of Science
Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Betreuer/Supervisor:
Dr. Helge Rhodin, École polytechnique fédérale de Lausanne
Lausanne, Swiss

Gutachter/Reviewers:
Prof. Dr. Christian Theobalt, Max Planck Institute for Informatics
Saarbrücken, Germany
Prof. Dr. Hans-Peter Seidel, Max Planck Institute for Informatics
Saarbrücken, Germany

Dekan/Dean:
Prof. Dr. Frank-Olaf Schreyer, Saarland University
Saarbrücken, Germany

Eingereicht am/Thesis submitted:
11.07.2017

Universität des Saarlandes
Fachrichtung 6.2 - Informatik
Im Stadtwald - Building E 1 3
66123 Saarbrücken

Declarations

Eidesstattliche Erklärung/Statement in Lieu of an Oath:
Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.
I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Signature:

Marc Habermann

Einverständniserklärung/Declaration of Consent:
Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.
I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Signature:

Marc Habermann

Abstract

Over the last decade there was an enormous amount of research projects that tackled the problem of human motion capture, given a single view. At the same time, reconstructing non-rigidly deforming objects remains challenging, which can also be seen in video games, where character animations often look orders of magnitude better than for example cloth simulations. Estimating such motions is a difficult task and a lot of existing approaches therefore need a multi-view setup or depth cameras. Apart from these sophisticated recording conditions, there are also monocular methods, but they often have a low quality or restrict themselves to specific object categories, dense features, known lighting conditions or structured light sources.

The main goal of this work was to construct an approach that is as general and accurate as possible, while at the same time the recording setup should be simple and user-friendly. Therefore, the thesis presents a novel energy-based framework, named RONDA, that solves the challenging task of non-rigid motion estimation, given a single RGB video and a template of the object. To achieve a high accuracy, several penalties are combined where each of them is based on plausible assumptions. For example the thesis uses a term to ensure a photometric alignment and a smooth surface deformation. As a result, the proposed method can deal with a large amount of different motions and a variety of object categories.

To further refine the estimation for the case of woven fabrics, the thesis examined the local structures of different clothes and found that there are patterns, which can be used to improve the reconstructions. Due to the increasing video resolution of recording devices, these small patterns are also visible in the frames and a novel constraint is presented that exploits this. The combined energy is optimized on the graphics processing unit such that results are obtained at interactive frame rates. Quantitative and qualitative evaluations are shown to confirm the accuracy and generality of the proposed method.

Acknowledgements

I want to thank Professor Christian Theobalt for giving me the opportunity to write my thesis in the Graphics, Vision and Video Group, the supervision and the support during this time. I also want to thank Professor Hans-Peter Seidel for being my second reviewer.

Special thanks are due to my supervisor Doctor Helge Rhodin, who was really committed. Whenever I had questions or needed someone to discuss certain ideas, he took his time to talk about the problems and to find solutions for them together with me. But also Doctor Michael Zollhöfer and Doctor Weipeng Xu helped me to develop the ideas and thoughts presented in this work. I would like to thank them, too.

I really enjoyed the inspiring discussions and the atmosphere during the group meetings. I am happy and excited to continue collaborations in this great group. Therefore, I give thanks to these people.

At last, I want to thank my girlfriend Antonia Schliesing, my family, Sabrina Nowack, Max Augustin, Sebastian Cucerca, Gian-Luca Kiefer and Pascal Grittmann for proofreading the thesis.

Contents

List of Figures

List of Abbreviations

2D	Two dimensional.
3D	Three dimensional.
ARAP	As-rigid-as-possible.
CG	Conjugate Gradient method.
CNN	Convolutional neural network.
CPU	Central processing unit.
CUDA	Compute unified device architecture.
fps	Frames per second.
GPU	Graphics processing unit.
HOG	Histogram of Oriented Gradients.
IR	Infrared.
LSE	Linear system of equations.
MPI	Max Planck Institute.
NLSE	Non-linear system of equations.
NRSfM	Non-rigid structure from motion.
PCA	Principal component analysis.
RAM	Random-access memory.
RGB	Red green blue.
RONDA	Reconstruction of non-rigid surfaces.
SfS	Shape from shading.
SfT	Shape from texture.
SMPL	Skinned multi-person linear.

List of Symbols

C_{B}	Blue color channel of a pixel in the smoothed frame.
C_{G}	Green color channel of a pixel in the smoothed frame.
C_{R}	Red color channel of a pixel in the smoothed frame.
F	Number of faces of the template.
I_{Dir}	Texture direction image.
I_t	Discrete image function of one color or grey channel t.
I_{t_u}	Discrete image gradient of the color t along the u-axis.
I_{t_v}	Discrete image gradient of the color t along the v-axis.
M	Mask region.
N	Number of vertices of the template.
T_1	First order Taylor expansion.
T	Number of video frames.
$\alpha_{\max}$	Dominant frame/mesh angle.
α_u	Scale coefficient in u-direction.
α_v	Scale coefficient in v-direction.
γ	Skew coefficient between the u- and v-direction.
$\hat{\mathbf{V}}$	Initial template model in matrix form.
$\hat{f}$	Focal length.
$\lambda_{\mathrm{Acceleration}}$	Acceleration weight.
λ_{Arap}	As-rigid-as-possible weight.
λ_{Edge}	Edge weight.
$\lambda_{\mathrm{Laplacian}}$	Laplacian weight.
λ_{Photo}	Photometric alignment weight.
$\lambda_{\mathrm{Texture}}$	Texture weight.
$\lambda_{\mathrm{Velocity}}$	Velocity weight.
$\mathcal{I}_t$	Continuous image function of one color or grey channel t.

$\mathcal{N}$	Neighbours of a vertex of the template.
$\mathcal{R}$	Rotation function.
π	Projection function.
ρ	Threshold function for the texture term.
σ	Threshold function for the photometric alignment term.
$\mathbf{B}$	Barycentric coordinates for the faces of the mesh in matrix form.
$\mathbf{F}$	The faces of the mesh in matrix form.
$\mathbf{G}_w$	Gaussian kernel with width w.
$\mathbf{P}$	Intrinsic camera matrix.
$\mathbf{S}$	Sobel operator in matrix form.
$\mathbf{U}$	UV coordinate matrix.
$\mathbf{V}^t$	Mesh of the tth frame in matrix form.
$\mathbf{V}$	Current mesh estimate.
$\mathbf{\Phi}$	Euler angles in matrix form.
$\mathbf{c}_{\mathrm{B}_i}$	Blue color of the ith vertex of the template.
$\mathbf{c}_{\mathrm{G}_i}$	Green color of the ith vertex of the template.
$\mathbf{c}_{\mathrm{R}_i}$	Red color of the ith vertex of the template.
$\mathbf{d}$	Dominant frame/mesh/texture map gradient.
$\mathbf{h}$	Histogram.
b	Barycentric coordinate function.
c	Center point function.
$e_{\mathrm{Acceleration}}$	Acceleration energy term.
e_{Arap}	As-rigid-as-possible term.
e_{Edge}	Edge-length energy term.
$e_{\mathrm{Laplacian}}$	Laplacian smoothness energy term.
e_{Photo}	Photometric energy term.
e_{Texture}	Texture energy term.
e_{Velocity}	Velocity energy term.
h_u	Horizontal distance between neighbouring pixels .
h_v	Vertical distance between neighbouring pixels .
o_x	x-coordinate of the Principal Point.
o_y	y-coordinate of the Principal Point.
p_{a}	Pixel aspect.

r_u	Image resolution along the u-axis.
r_v	Image resolution along the v-axis.
s_u	Sensor size along the u-axis.
s_v	Sensor size along the v-axis.
u_0	First image coordinate of the Principal point.
v_0	Second image coordinate of the Principal point.

Chapter 1

Introduction

1.1 Motivation

In the early 70's, the film series Planet of the Apes caused a sensation and was commercially very successful. Looking particularly at the recording conditions at that time, the actors had to wear uncomfortable ape costumes because there were no other techniques available to produce natural appearing sequences and especially naturalistic motion. In the movie Dawn of the Planet of the Apes published in 2014, the situation changed drastically since computer graphics revolutionised making movies. Watching the animal character Caesar (see Figure 1.1 right) when he climbs through the post-apocalyptic scenes is just impressive, having in mind that all these sequences are computer animated.

But this is only one of many examples. In most today's movies one will find a lot of synthetic animations. This can be for example actors moving in a virtual environment or digital objects and characters, which are inserted into a video of the real world or even completely virtual scenes. The shift from classically recorded films to computer animated ones is caused by the significant improvement of rendering techniques and the higher computational power of modern hardware. Realistic images can be produced in an adequate amount of time at comparatively low cost.

As a result, one is able to produce nice-looking movies of fictional scenes which even agree with the physical laws of light transport and one can in fact use freely available software such as the Cycles render engine of Blender [10]. But what about the motion in a video? State-of-the-art rendering techniques do not imply realistic animations. However, humans are very sensitive to it and can distinguish artificial movements from natural ones. An article on the internet [11] refers to the work of Saygin et al. [12], who study the human perception, and says that "[...] the

Figure 1.1: Dawn of the Planet of the Apes. Left. Original recorded film sequence where the actor Andy Serkis wears a motion capture suit. Right. Final rendered movie where the motion of the actor was mapped to a 3D model of a monkey. Pictures where retrieved from [9].

researchers concluded that we're fine with anything that tries to look human–until it starts to move. If those movements aren't distinctly human, our brains don't know how to process what they're seeing.". So that apart from physically-based rendering, motion has to be close to reality, too. The computer science community also recognized that and Ren et al. [13] for example tried to build a data-driven approach, which automatically judges if a human motion is natural or not. People in film industries as well realized the evidence and huge companies nowadays have their own staff just to create natural-looking animations. There are two main approaches for them to try to achieve this. Either one animates the virtual object by hand or one records a real motion sequence and maps it to a three-dimensional (3D) model. The latter needs the right equipment. In case of character motion the actors often have to wear a suit with sensors or markers (see Figure 1.1 left) that makes it easier to track the position of the body parts. For general objects like the carpet in Figure 1.2 and also for humans one often uses depth cameras or a multi-view setup to record the sequence without requiring markers. After the data acquisition, motion reconstruction algorithms are used to extract the movement which can then be used for further applications.

But both approaches have some drawbacks. When it comes to creating animation by hand, everyone can try this in modeling softwares, nevertheless it is really difficult and time-consuming. Imagine one wants to animate a human body model like the SMPL model of Loper et al. [14] consisting of 23 joints, and one also has to choose the right shape parameters. To record sequences over thousands of frames, it takes a lot of time and needs a fair bit of skill to place the joints in the right locations. In case of automated reconstruction techniques, multi-view setups and body suits are impractical for private users since they are expensive and calibrating the devices

is a time-consuming process. Large film enterprises have their own green screen studios and they also engage many experts, who are able to control the devices. But ordinary people do not have such conditions. Apart from that, depth cameras are still more expensive than RGB ones and most people feel less comfortable with them. Additionally, one cannot use the huge amount of freely available RGB-only video footage on social media platforms like YouTube, Facebook or Twitter.

On these grounds, one of the main goals of the thesis was that the recording setup is just a single camera to keep it as simple as possible and to be able to use a lot of existing online content. Focusing on general objects, the thesis presents a method named RONDA, which is able to reconstruct non-rigid surface deformations, such as clothes moving in the wind, from a single RGB video and a template mesh of the object (see Figure 1.2). Given this input, the algorithm automatically produces a deformation sequence of the virtual model that coincides with the movement in the video. To solve this challenging task, the thesis created an energy-based approach, which consists of different cost terms including photometric alignment and several smoothness constraints. The resulting non-convex function is then efficiently optimized using the computational power of modern graphics processing units (GPU). Besides this, the thesis examines how the amount of useful information changes when one records high resolution videos since most of today's devices can produce such formats. Smartphones for example can already film 4K videos. The proposed work then shows how to use this additional knowledge for better non-rigid motion estimation. Therefore, RONDA focuses on the reconstruction of woven fabrics. Constraining the setting to clothes and being given the additional high resolution information has the advantage that one can build additional assumptions based on the textural appearance. These findings are integrated into the energy function which further improves the quality of reconstructions as will be shown later.

Since the proposed method uses only one RGB camera, one can get rid of the above described problems at the price of needing a template and high resolution video data. Today's cameras are at low cost (one can even use a smartphone to record a video) and there is neither expert knowledge nor a professional equipment needed. Furthermore, one is able to use online video content. Therefore, the process of extracting motions from real scenes becomes easier, faster and cheaper with the proposed approach.

Given the animation data, typical applications are, as already mentioned, character animations in movies. But the 3D deformation sequence can also be used to improve the quality of motion in video games. Moreover, the method could also be beneficial for researchers. Since the growing field of Machine Learning, in particular

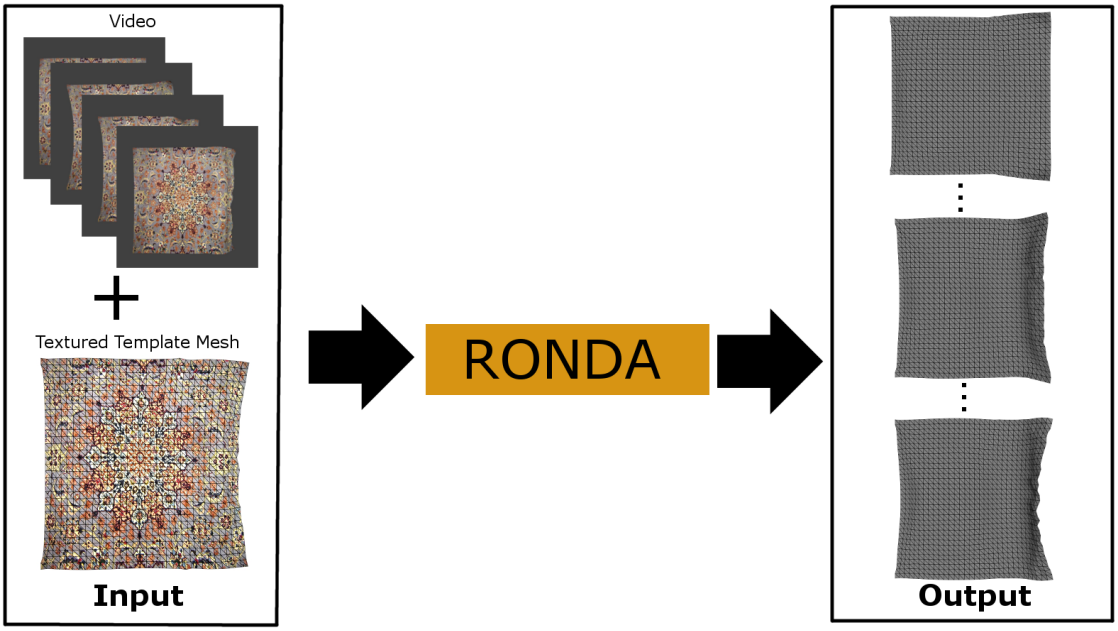

Figure 1.2: Overview of RONDA. It takes a video and a textured template mesh as input. The proposed approach then computes the non-rigid deformation of the object shown in the video. The output is a sequence of meshes which can be used for further applications.

Neural Networks, becomes more and more important, the demand for the training data also ascends. Especially motion data is rare and not easy to generate. Therefore, the thesis could be practical to create motion data bases in a quick and easy fashion.

All in all, RONDA solves the challenging task of estimating non-rigid deformations. The core advance is a novel texture-direction term that exploits the regular structures visible in high-resolution video, such as the yarn patterns of fabrics. But the thesis' method can also handle other types of objects because it integrates several cost terms already proposed by previous approaches. Typical applications reach from movies over games up to science.

1.2 Challenges

The thesis tries to reconstruct 3D motion from two-dimensional (2D) videos or more precisely from 2D projections of the original object. According to the definition this is an inverse problem, which is in general hard to solve. Estimating non-rigid deformations from monocular video is, in fact, under-constrained since one wants to

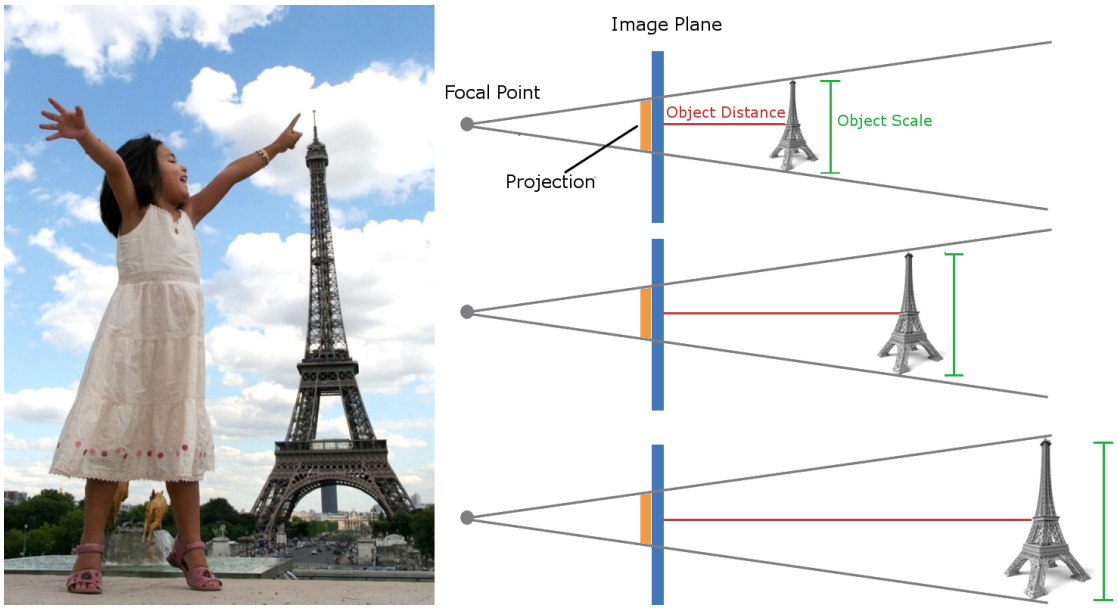

Figure 1.3: Left. The child and the Eiffel Tower seems to be the same size. Right. Different pairs of distances and object sizes lead to the same projection. Parts of the graphic were retrieved from [15, 16].

estimate higher dimensional data (3D deformation) from lower dimensional input (2D video). As a consequence, there are multiple possible solutions for the same problem. In the following, some examples are explained to illustrate the challenges.

Scale ambiguity. Most of the people will implicity know what scale ambiguity is, since there are a lot of photos on the internet like the one in Figure 1.3 left, that illustrates it well. It comes from the fact that if one moves an object further away one can compensate the decrease of the projection size by scaling the object (see Figure 1.3 right). This implies that different camera distances and their corresponding scale factors can lead to the same image, which makes it hard to estimate the true underlying geometry from a single picture.

Convex-concave ambiguity. The concept behind this ambiguity is that the projection of a convex surface can be imitated by a concave one and vice versa (see Figure 1.4 left). Especially in the orthographic projection it is difficult to distinguish between concave and convex shapes caused by the fact that the projection lines are parallel to each other. Consequentially, a convex shape can be converted into a concave one by mirroring it along the plane which is orthogonal to the projection lines such that the image of the new surface still looks like the old one. This principle also holds for concave shapes. If one wants to get the same effect with a perspective camera, one additionally has to stretch the surface to trick the observer.

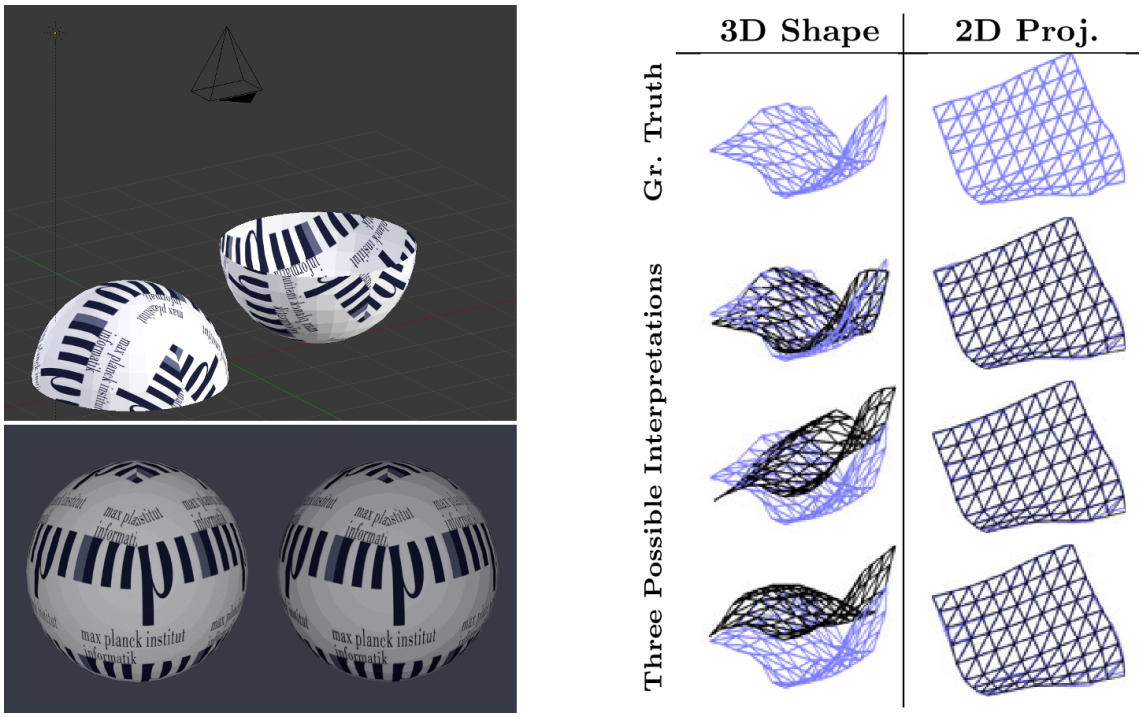

Figure 1.4: Top left. Virtual scene containing a convex and a concave surface. Bottom left. Render result of the scene. Note that the two projections look almost the same despite some shading differences. Right. Blue geometry represents the ground truth mesh and the corresponding 2D projection. Black geometries are other shapes that lead to almost the same 2D projection. Parts of the graphic were retrieved from [17].

General deformation ambiguity. In general, different deformations can lead to the same projection. Moreno-Noguer and Porta [17] visualized this phenomena very well in their work as shown in Figure 1.4 right.

Being aware of these challenges the thesis presents different constraints which help to solve the above ambiguities with the result that the initial ill-posed problem becomes well-posed.

1.3 Overview

In the following chapter (Chapter 2) an overview of existing approaches for motion reconstruction is given. Therefore, it is distinguished between the recording equipments which are used by the approaches for example multi-view methods. Chapter 3 introduces the mathematical ingredients and the corresponding notations. Next, the underlying energy function for estimating non-rigid surface motion in the general case is explained (Chapter 4). The special instance of fabrics is introduced in

Chapter 5. Having the energy function the next chapter (Chapter 6) explains the structure of the GPU-based optimization framework that is used to solve the non-convex energy function. Afterwards, the results are shown (Chapter 7) and some limitations and future work are discussed (Chapter 8). Finally, Chapter 9 gives a summary of the presented work.

Chapter 2

Related Work

2.1 Multi-view Methods

First, some multi-view methods will be reviewed since they are also related to the thesis' topic and they help to get a better understanding of the bigger picture of reconstructions of non-rigid deformations. Multi-view means that several cameras at different locations are used to record the scene.

Already in 2001 Carceroni and Kutulakos [18] proposed a template-free method that was able to recover shape, reflectance and motion given seven different camera views of the object and full knowledge about the lighting conditions. Therefore, they introduced so-called dynamic surfels representing the shape, reflectance and motion in a high dimensional space. To find the correct surfel parameters they solve an energy function based on the generalized temporal brightness constancy assumption, which means that surface points do not change their brightness value over time and space (apart from specular reflections).

A few years later, Pons et al. [19] published a variational multi-view approach that estimates non-rigid deformations of unknown objects. They use the image of one camera view to project it back onto the current surface guess. The "textured" geometry is then reviewed from another position and compared with the corresponding frame to get an estimate if the actual projections of this geometry are consistent over different views. They developed an energy functional based on this idea and optimize it by using gradient descend. Compared to the previous paper they deliver a more detailed surface geometry.

The last approach mentioned here is the one of Perriollat and Bartoli [20], that focuses, in contrast to the other methods, on paper-like surfaces. This has the advantage that they were able to build a parametric model of the geometry which

Figure 2.1: Results of Zollhöfer et al. [1]. From left to right. Frames from the input video. Reconstructed 3D geometry. Re-targeted facial expression. Re-textured version of the estimated mesh. The graphic was retrieved from [1].

has a lower dimensionality than the non-parametric ones at the expense of worse generalization. To estimate the non-rigid deformation of the surface they use a multiple view fitting algorithm.

Although the presented approaches produce good results they nevertheless need a multi-view architecture which, as already said in the introduction, is really impractical for ordinary users.

2.2 RGB-D Methods

To simplify the recording setup, the following approaches use a single depth camera instead of multiple RGB-only devices. Liao et al. [21] proposed a method that tracks a deformable object given its depth. Since they only use a single camera, one of the main issues is occlusion. They tackle this problem by assuming that the motion in a short time interval is continuous and predictable. To estimate the occluded parts of the surface, too, they use the reconstructions from different frames where the occluded parts were probably visible. The partial surfaces are then aligned with the help of a mesh warping algorithm. Finally, a volumetric method is used to merge these pieces.

In contrast to Liao, Zollhöfer et al. [1] proposed a newly invented RGB-IR stereo camera to capture deformable objects placed close to the recording device. Afterwards, their customized patch-match stereo algorithm estimates the RGB-D data. To generalize well with respect to the shape of the objects they initially create a template mesh by scanning the object. Given this input, they perform non-rigid registration between the template and the RGB-D data using an as-rigid-as-possible constraint. Real-time is realized by an efficient GPU-based solver framework which is also the baseline for the thesis' optimization. Their results can be seen in Figure 2.1.

Newcombe et al. [22] presented a real-time method that is able to reconstruct a static scene using the data of a Microsoft's Kinect sensor. During streaming the algorithm refines a single implicit model of the recorded environment. Therefore, they need the current camera position and viewing direction, which is estimated by comparing the live depth data with the relative depth of the current scene model using a coarse-to-fine iterative closest point algorithm.

The main drawback of the previous approach is that they assume that the scene is static in most parts. To overcome this problem, Newcombe et al. [23] proposed in their follow-up work a new method, called DynamicFusion, to be able to handle dynamic environments. Therefore, they added a six dimensional motion field, that captures the deformation of the surfaces.

Although these methods perform great in non-rigid motion reconstruction one has to bear in mind that they use depth sensors. At the moment, these devices are still more costly than cameras and most users feel more comfortable with ordinary cameras than with depth sensors. The main disadvantage of this kind of hardware is that one cannot use the huge amount of RGB-only video footage provided by social media platforms like YouTube, Facebook or Twitter. On these grounds, the thesis focuses on RGB methods and the next section presents the related work in this research area.

2.3 RGB-only Methods

There are a lot of publications in the field of estimating non-rigid deformations from monocular video. To get a well-structured overview of them, the thesis divided the approaches into three categories based on the constraints, that they impose to make sure that the problem formulation is well-posed. Clearly, some papers belong to multiple classes, nevertheless their main contributions can be uniquely categorized.

2.3.1 Appearance Constraints

Publications, which constrain the surface appearance of the tracked object, are the first out of the three categories. In the following, approaches based on the idea of so-called structured light sources are reviewed since they build the bridge between the previously discussed RGB-D methods. After that, shape from texture (SfT) approaches are examined, because they are strongly related to the proposed texture-based constraint.

Structured light sources. To start with the first subcategory of appearance constraints, the paper of Hernández et al. [24] is presented. They use structured light sources that illuminate the object of interest, similar to depth cameras which also use light patterns. As a consequence, the appearance of the surface is constrained by the light. A camera records the deforming lit object and they propose an algorithm to estimate the surface normals that are then used to compute depth maps per frame. Afterwards, the first image of the sequence is used to build a mesh, which is deformed over time such that it matches the depth data of the corresponding frame. Additionally, they constrain the motion with an optic flow term and an as-rigid-as-possible guideline. Zhang et al. [25] and Weise et al. [26] also proposed methods based on structured light sources. All their motion reconstructions look reasonable and nice but they again have a customized and more difficult setup since the user has to install light sources. These illuminants disallow the use of existing videos recorded under ordinary lighting conditions.

Shape from texture. Since a main part of RONDA consists of the newly invented texture term, which helps to further restrict the solution space for the case of fabrics, the corresponding related work should also be mentioned at this point. They all have in common that they assume texture patterns on the surface that is another form of an appearance constraint. Already in 1992, Gårding [27] retrieves the object's shape by computing the textural distortion caused by the projection. Loh and Hartley [28] proposed a more generalized method that does not rely on certain types of textures and viewing conditions. While their methods produce reasonable deformations they are still of poor quality. Rao et al. [2] went in a different direction since they proposed a method which is able to compute the directions of a texture by using the gradient of a Gaussian. In particular, their method proceeds in five steps. First, they smooth the image with a Gaussian kernel, followed by the computation of the gradient image. Next, these two-dimensional gradients are converted to angles and an averaging is applied in a certain neighbourhood to obtain the dominant angle of the region. Finally, they propose a coherence measure such that regions, where no unique gradient direction is present, are detected. A result is shown in Figure 2.2. The work of Rao and RONDA have in common that both do texture analysis and up to the third step they are equal, but the thesis approach differs in the last two ones which will be explained later and it additionally uses these orientations to estimate the shape of an object. Liang et al. [3] built up on this idea and tried to create a flattened virtual model of curved documents visualized in Figure 2.2, given a single image of it. They estimate the shape of the paper with the help of the texture direction created by the text on the document.

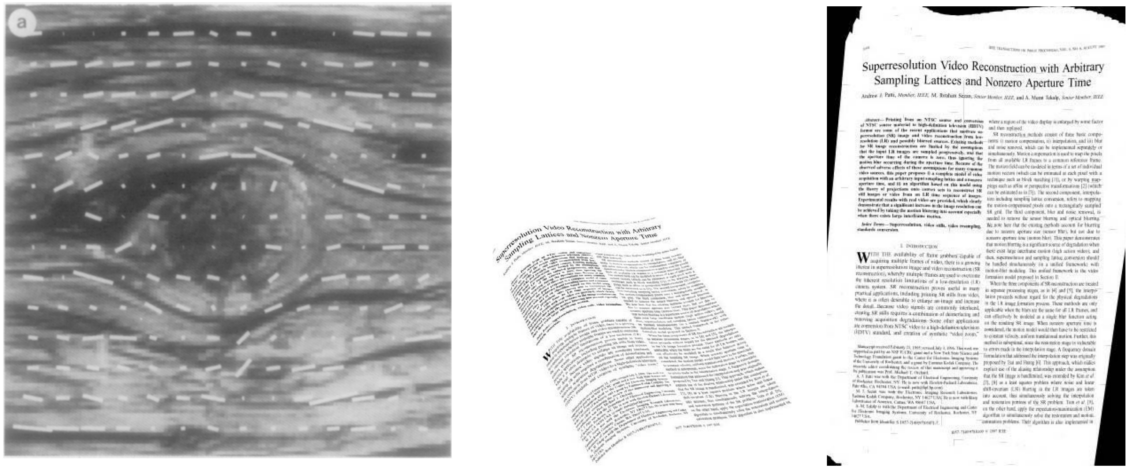

Figure 2.2: Results of Rao et al. [2] and Liang et al. [3]. Left. Image of a texture which has line-like patterns. The lines mark the estimated texture orientations of Rao's approach. Right. In the middle one can see an input image of a curved document and on the right side one can see the flattened reconstruction of Liang et al. [3]. The graphics were retrieved from [2] and [3].

But in contrast to the proposed approach, they represent the object as a developable surface and use vanishing points for the reconstruction which is different from RONDA's energy-based method that uses a vertex model. Although the last two mentioned approaches are not directly related to the thesis' problem, they build the main inspiration for the proposed texture-based constraint. White and Forsyth [29] combined shape-from-texture and shape-from-shading (SfS). The texture information alone introduces a so-called "texture normal ambiguity" which means that two surface normals are possible to achieve the observed projected texture. To get rid of this problem they use a shading constraint that leads to an unique solution. While their reconstruction can compete with multi-view results, they nevertheless only show results where the objects have dense patterns on the surface, whereas the thesis can also interpolate information in non-textured surface regions.

On the one hand, structured light sources are an interesting direction for solving the non-rigid motion estimation. But their complicated setup and the implied restrictions contradict the thesis' goal of a versatile and easy-to-deploy solution. On the other hand, shape from texture approaches by nature heavily rely on dense patterns or features. But they can still be of use if they are combined with other techniques shown in later chapters.

2.3.2 Geometric Constraints

Next, approaches, which assume certain knowledge about the geometry of the filmed objects, are discussed. It starts with the least restricted setting, namely template-free methods, followed by parametric models, that already rely on basis shapes. Afterwards, automated template acquisition methods and template-based non-rigid surface deformation approaches are reviewed. The latter assumes the most knowledge about the geometry.

Template-free methods. There were several template-free in literature also called non-rigid structure from motion (NRSfM) approaches, which tried to solve the described problem by using global models [30, 31, 32, 33, 34], but their main drawback was that they were not able to reconstruct strong deformations. Russell et al. [35] use multiple local models instead of a global one such that he can also capture larger deformations. But they only show results where features can easily be generated. On the contrary, RONDA can also deal with low textured regions where feature extraction is challenging. Garg et al. [36] introduced a variational approach to solve the problem of non-rigid structure from motion. While their reconstructed geometries have a high level of detail, their main drawback is that they rely on dense 2D correspondences to estimate the mesh. Therefore, their method stands and falls with the computation of these correspondences. Apart from that, an orthographic camera model is chosen, which is not valid if the object is close to the camera. Dai et al. [37] go in a different direction because they tried to build a prior-free NRSfM approach. On the one hand, it is theoretically beneficial to have a method without any constraint toward the surface or the camera motions. But on the other hand, their reconstructions have a low quality and they give only a rough estimate of the true geometry of the object such that it is impractical for real applications.

Parametric models. Bregler et al. [38] introduced a method to recover non-rigid shape deformations from a single video sequence without using a template. Nevertheless, their approach needs a set of basis shapes for the corresponding object category, e.g. faces. Additionally, their reconstruction quality is really poor compared to today's algorithms. Garrido et al. [4] proposed a method to capture non-rigid and detailed face geometries. The motion and the mesh have a high quality, which can be seen in Figure 2.3 and they can directly be used for further applications. However, they need a 3D scan of the subject's face, a blend shape model and they are only able to reconstruct faces, whereas the thesis wants to focus on general objects.

Figure 2.3: Results of Garrido et al. [4]. Top row. Frames of the input video. Bottom row. The frames overlayed with the estimated geometries. Note that the expressions are realistic and even small wrinkles were reconstructed. The graphic was retrieved from [4].

Template acquisition. In recent years, researchers invented a lot of ideas to easily generate a template model given multiple images such that it is not too complicated for ordinary users. This can be either in real-time like the proposed method of Pan et al. [39] or offline approaches like the ones of Labatut et al. [40] or Pollefeys et al. [41]. Agisoft [42] even invented a commerical program that one can buy. It produces a fine-detailed and textured geometry given a set of images of the object. An example output can be seen in Figure 2.4. Since the model acquisition nowadays is no longer a big problem the thesis focuses on building a template-based approach and in the following papers will be presented, which also pursue this strategy or assume that 3D to 2D correspondences are known.

Template-based methods. In 2007 Salzmann et al. [43] gave a mathematical explanation of ambiguities "[...] that occur when tracking a generic deformable surface under monocular perspective projection given 3–D to 2–D correspondence.". They also proposed a solution to this problem by taking the image sequence into account instead of a single frame. However, they rely on densely textured surfaces and 3D to 2D correspondences, whereas RONDA can also handle partially uniform regions and it only needs a manual initialization where the user has to place the template such that it correctly projects onto the first frame. One year later, Salzmann et al. [8] proposed a closed-form solution, which does not rely on 3D to 2D

Figure 2.4: From left to right, the first three images are examples out of 34 input pictures. The last picture is a rendered version of the 3D model produced by the Agisoft software. Note that the quality of the shirt is really good although the input only consists of 34 images.

correspondences for the input image sequence. Instead, they assume that a separate picture is given where the 3D configuration of the shape is known. Using this information they are able to reconstruct inelastic motions shown in the video. To make their method work for larger meshes too, they developed a deformation model, that was obtained by applying a Principal Component Analysis (PCA) to a set of training motions where the geometry was known. Although they do not have to know the shape, which the template has in the first frame, they still need a configuration but just for another individual picture. In consequence, they just shifted the initial constraint compared to previous approaches. Salzmann et al. [44] also published a method that uses local surface models to reconstruct different types of materials and shapes. With these models they can estimate geometries, which have a low amount of texture. Instead, the thesis uses a local smoothness prior so that it is also able to reproduce low textured meshes. Shen et al. [45] proposed a method to make the reprojection error more robust with respect to outliers and larger motions between consecutive frames using a different penalization norm for their feature-based data term. Although the thesis does not directly work on 3D to 2D correspondences but on a photometric error, which will be explained later, it is also robust to outliers by a customized penalty function. In 2010 Moreno-Noguer et al. [17] tried to overcome the previously in Section 1.2 described ambiguities by selecting multiple so-called candidate shapes. To choose the final one out of the set they use a shading and motion constraint. However, it is assumed that there is only one light source present which is often not the case, especially in in-door conditions. This makes their approach less general compared to RONDA.

Figure 2.5: Results of Yu et al. [5]. Left. The top row shows some frames of the input video. Below one can see the reconstruction of Garg et al. [36] and Yu et al. [5], which has a better quality since the overall shape is more similar to the original one in the input video. Right. Top row shows frames of the "Teddy sequence". The bottom row shows Yu's reconstructions. The graphic was retrieved from [5].

Brunet et al. [46] proposed an energy-based reconstruction method. The problem boils down to minimizing a non-convex function that consists of three terms responsible for the correct reprojection, preservation of edge lengths and surface smoothness. Likewise, the thesis uses such constraints but they are formulated in a different way and additional terms are imposed to make the problem easier to solve and to be more robust with respect to ambiguities. Unlike Perriollat et al. [47] and most other papers, Malti et al. [48] proposed a pixel-based approach instead of feature-based ones. RONDA also makes use of this idea to formulate the photometric error term that will be discussed later. Östland et al. [49] use the so-called Laplacian formalism proposed by Sorkine et al. [50] in combination with control points on the mesh to obtain a well-conditioned system of equations. Inspired by that the thesis uses a Laplacian surface constraint. Bartoli et al. [51] proof in their rather theoretical work that "template-based isometric surface reconstruction from a single view registered to the template generally has a single solution". They also proposed the first analytical algorithm to solve this type of problem which goes in a different direction compared to the thesis' energy-based approach. In contrast to all previously discussed methods, Varol et al. [6] invented an algorithm where a latent variable model is learned from a labeled training set. Nevertheless, they only showed results of highly textured surfaces. The last paper in this paragraph is the one of Yu et al. [5]. They proposed an energy-based non-convex optimization formalism to solve the challenging task of reconstruction of non-rigid deformations given a template and a single video. Since their results have a high quality (see Figure 2.5) and they

are able to reconstruct a variety of objects, they contribute the most to the ideas that will be presented in this thesis and some parts of their energy function are also a component of this work. However, they use the Ceres solver [52], which runs on the central processing unit (CPU), whereas the thesis uses a GPU-based architecture that is much faster and RONDA proposes additional regularization terms to further reduce the solution space.

What one can learn from these approaches is that if one wants to be template-free this has a certain cost since the problem is really challenging and in general ill-posed. As the above papers showed one has to choose if one wants to build a high quality geometry at the price of relying on dense correspondences or if one wants to be as general as possible by accepting that the quality of the geometry is lower. None of the limitations complies with the thesis' claims and therefore they offer no opportunity to solve the problem. On the other hand, more recent parametric models like the one of Garrido et al. [4] achieve very accurate and realistic results. Nevertheless, the object category has to be constrained for example to faces whereas RONDA wants to be able to reconstruct general objects. The template acquisition nowadays only needs a single camera and since it is accurate too, the thesis builds on the idea of a template-based framework. Previous work in this research field was very successful in terms of quality and generality. However, all approaches have certain drawbacks and none of them focuses on the case of woven fabrics.

2.3.3 Deformation Constraints

While the previous methods all have in common that they implicitly or explicitly assume that the surface is inelastic (also called isometric), there are also efforts to less constrain the types of deformations. The approaches of Moreno-Noguer et al. [53] and Malti et al. [54] show that the elastic case can also be handled by for example introducing a shading constraint. Since the thesis wants to focus on woven fabrics, which are inelastic most of the time, this case will not be discussed any further at this point. Between elastic and inelastic surfaces there is also a third category where the edges can shrink but not grow. Salzmann et al. [55, 56] argued in their work that if one has sharp folds, like it is the case for clothes, they can only be well reconstructed if the edges are allowed to shrink. This is caused by the fact that while the geodesic distance on a surface can remain constant, the Euclidean one can shrink (see Figure 2.6). The thesis instead consciously keeps the edge length constraint which will be discussed later and overcomes this problem by subdividing the mesh until the edge lengths are small enough to handle these kind of folds.

Figure 2.6: Left. Unfolded surface where the two dots mark the vertices of the underlying mesh. Middle. Folded surface. Note that geodesic distance between the two points remains the same but the Euclidean one (marked as dotted line) is smaller. Right. By inserting additional vertices (green dots) one is able to reconstruct the true surface without allowing the edges to shrink. Parts of the graphic were retrieved from [56].

2.4 Summary

In summary, one can see that none of the above methods is completely customized towards the described problem of reconstruction of non-rigid surfaces and especially fabrics given a very simple hardware setup. Multi-view methods are able to produce high quality reconstructions but the setting is not user-friendly. The same holds for RGB-D methods and approaches, which use structured light sources. Being template-free or using parametric models comes with a certain trade-off in terms of quality or generality. Since the template acquisition nowadays is accurate, general and easy-to-deploy, it forms the perfect basis for the proposed algorithm. Recent template-based approaches often need 3D to 2D correspondences or highly textured surfaces. Instead, RONDA is versatile and fast. In addition, it combines several advantages from previous papers, does not rely on features and it is also able to reconstruct low textured regions. Next, the mathematical ingredients to solve a non-linear system of equations are explained and basic as well as more advanced image analysis techniques are introduced.

Chapter 3

Math Background

3.1 Linear and Non-linear System of Equations

Since the thesis tries to estimate non-rigid deformations by optimizing a multi-dimensional function, the notions of *linear systems of equations* (LSE) and *non-linear systems of equations* (NLSE) have to be introduced. The intuition behind these terms is that one has multiple equations in the same unknowns, also called *variables*. An example could be

$$\begin{aligned} 2x - 3y &= -2 \\ 4x + y &= 24. \end{aligned} \tag{3.1}$$

Here, two equations are given and $x, y \in \mathbb{R}$ are the unknowns. One says that a system is *solvable* if variable values can be found such that all equations are fulfilled. The above example is solvable because $(x, y) = (5, 4)$ is a solution.

LSE are systems where the equations are linear in the unknowns as it is the case in Equation 3.1. In general, they have the form

$$\begin{array}{ccccccccccc} \mathbf{A}_{1,1}\mathbf{x}_1 & + & \mathbf{A}_{1,2}\mathbf{x}_2 & + & \cdots & + & \mathbf{A}_{1,n-1}\mathbf{x}_{n-1} & + & \mathbf{A}_{1,n}\mathbf{x}_n & = & \mathbf{b}_1 \\ \mathbf{A}_{2,1}\mathbf{x}_1 & + & \mathbf{A}_{2,2}\mathbf{x}_2 & + & \cdots & + & \mathbf{A}_{2,n-1}\mathbf{x}_{n-1} & + & \mathbf{A}_{2,n}\mathbf{x}_n & = & \mathbf{b}_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots & & \vdots \\ \mathbf{A}_{m,1}\mathbf{x}_1 & + & \mathbf{A}_{m,2}\mathbf{x}_2 & + & \cdots & + & \mathbf{A}_{m,n-1}\mathbf{x}_{n-1} & + & \mathbf{A}_{m,n}\mathbf{x}_n & = & \mathbf{b}_m. \end{array} \tag{3.2}$$

The variables are $\mathbf{x}_j \in \mathbb{R}$. $\mathbf{A}_{i,j} \in \mathbb{R}$ are also called *coefficients* and $\mathbf{b}_i \in \mathbb{R}$ are referred to as *constants* where $i \in 1, 2, ..., m$ and $j \in 1, 2, ..., n$. n is the number of unknowns and m is the count of equations.

Using matrix notation, Equation 3.2 can be reformulated as

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.3}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$ are defined as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,n} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{A}_{2,n} \\ \vdots & \vdots & \ddots & \cdots \\ \mathbf{A}_{m,1} & \mathbf{A}_{m,2} & \cdots & \mathbf{A}_{m,n} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_m \end{bmatrix}. \tag{3.4}$$

In contrast to LSE, NLSE are not linear with respect to the variables which makes them harder to solve. For example

$$\begin{aligned} 3x - y &= -2 \\ 2x^2 - y &= 0 \end{aligned} \tag{3.5}$$

is a non-linear system since the second equality is quadratic in x. Next, the term *optimization* is defined mathematically and it is shown how it relates to LSE and NLSE.

3.2 Non-linear Least Squares Optimization

The goal of an optimization is to find the values for the parameters $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)^\top$ such that they minimize a function $e(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ that is also called *cost* or *energy*. Mathematically, it can be written as

$$\arg\min_{\mathbf{x} \in \mathbb{R}^n} e(\mathbf{x}). \tag{3.6}$$

In case of least squares optimization the cost has the form

$$e(\mathbf{x}) = \sum_{i=0}^{m} (r_i(\mathbf{x}))^2 \tag{3.7}$$

$$= \|r(\mathbf{x})\|^2, \tag{3.8}$$

where $r(\mathbf{x}) = (r_1(\mathbf{x}), ..., r_m(\mathbf{x}))^\top : \mathbb{R}^n \to \mathbb{R}^m$ is also called *residuals*. The necessary condition for a minimum of a function $f(x) : \mathbb{R} \to \mathbb{R}$ which takes a single argument, is that the derivative $f'(x)$ is zero. Similarly, the necessary condition for a minimum of $e(\mathbf{x})$ in Equation 3.6 that takes multiple arguments, is that the partial derivatives $\frac{\partial e(\mathbf{x})}{\partial \mathbf{x}_j}$ have to be zero for all $j \in 1, ..., n$. Using the gradient notation

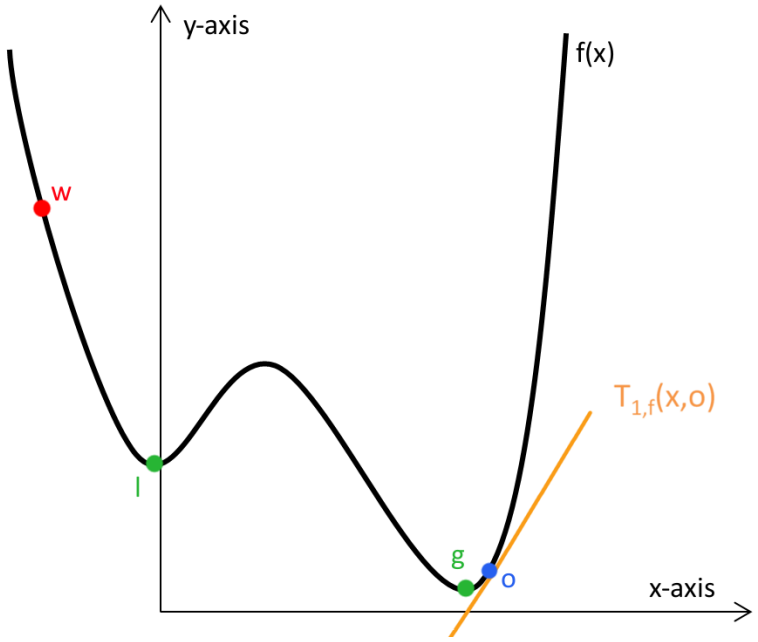

Figure 3.1: One-dimensional first-order Taylor expansion. $T_{1,f}(x, o)$ linearly approximates the non-convex function f at o. Note that for the one-dimensional case the Taylor expansion is equal to the tangent line of the function f at location o.

$\nabla f(\mathbf{y}) : \mathbb{R}^n \to \mathbb{R}^n$ defined as $\nabla f(\mathbf{y}) = \left(\frac{\partial f(\mathbf{y})}{\partial \mathbf{x}_1}, \frac{\partial f(\mathbf{y})}{\partial \mathbf{x}_2}, ..., \frac{\partial f(\mathbf{y})}{\partial \mathbf{x}_n}\right)^\top$ one gets the system of equations

$$\nabla e(\mathbf{x}) = \mathbf{0}. \tag{3.9}$$

In the least squares case this boils down to solving

$$\frac{\partial e(\mathbf{x})}{\partial \mathbf{x}_j} = \sum_{i=0}^{m} 2 r_i(\mathbf{x}) \frac{\partial r_i(\mathbf{x})}{\partial \mathbf{x}_j} = 0 \tag{3.10}$$

for all $j \in 1, ..., n$. Since the thesis' cost function is non-linear in $\mathbf{x}$, the system of equations 3.9 has not a closed-form solution and one has to use so-called *iterative methods*. The key concept is that one initializes the parameters $\mathbf{x}$ with an initial guess $\mathbf{x}_1 \in \mathbb{R}^n$. Then at each step Equation 3.9 has to be approximated with a LSE and the guess of the previous iteration $\mathbf{x}^t \in \mathbb{R}^n$ is refined to the new value $\mathbf{x}^{t+1} \in \mathbb{R}^n$.

To apply this strategy one first needs some mathematical tools, starting with the multi-dimensional *first-order Taylor series expansion* which is defined as

$$T_{1,f}(\mathbf{x}, \mathbf{o}) = f(\mathbf{o}) + \langle \nabla f(\mathbf{o}), \mathbf{x} - \mathbf{o} \rangle \tag{3.11}$$

$$= f(\mathbf{o}) + \sum_{j=0}^{m} \frac{\partial f(\mathbf{o})}{\partial \mathbf{x}_j} (\mathbf{x}_j - \mathbf{o}_j). \tag{3.12}$$

It gives a local estimate of a function around the point $\mathbf{o} \in \mathbb{R}^n$. One says that $T_{1,f}(\mathbf{x}, \mathbf{o})$ *linearly approximates* f at $\mathbf{o}$. An example of an one-dimensional first-order Taylor expansion can be seen in Figure 3.1. Next, the *Jacobian* $\mathbf{J}_f(\mathbf{y})$ of a function $f(\mathbf{y}) : \mathbb{R}^n \to \mathbb{R}^m$ is defined as

$$\mathbf{J}_f(\mathbf{y}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{y})}{\partial \mathbf{x}_1} & \frac{\partial f_1(\mathbf{y})}{\partial \mathbf{x}_2} & \cdots & \frac{\partial f_1(\mathbf{y})}{\partial \mathbf{x}_n} \\ \frac{\partial f_2(\mathbf{y})}{\partial \mathbf{x}_1} & \frac{\partial f_2(\mathbf{y})}{\partial \mathbf{x}_2} & \cdots & \frac{\partial f_2(\mathbf{y})}{\partial \mathbf{x}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{y})}{\partial \mathbf{x}_1} & \frac{\partial f_m(\mathbf{y})}{\partial \mathbf{x}_2} & \cdots & \frac{\partial f_m(\mathbf{y})}{\partial \mathbf{x}_n} \end{pmatrix} \tag{3.13}$$

and the corresponding transpose is

$$\mathbf{J}_f^\top(\mathbf{y}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{y})}{\partial \mathbf{x}_1} & \frac{\partial f_2(\mathbf{y})}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_m(\mathbf{y})}{\partial \mathbf{x}_1} \\ \frac{\partial f_1(\mathbf{y})}{\partial \mathbf{x}_2} & \frac{\partial f_2(\mathbf{y})}{\partial \mathbf{x}_2} & \cdots & \frac{\partial f_m(\mathbf{y})}{\partial \mathbf{x}_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1(\mathbf{y})}{\partial \mathbf{x}_n} & \frac{\partial f_2(\mathbf{y})}{\partial \mathbf{x}_n} & \cdots & \frac{\partial f_m(\mathbf{y})}{\partial \mathbf{x}_n} \end{pmatrix}. \tag{3.14}$$

Coming back to the iterative scheme the variable update from $\mathbf{x}^t$ to $\mathbf{x}^{t+1}$ can be achieved by adding $\Delta\mathbf{x} \in \mathbb{R}^n$, which results in

$$\mathbf{x} \approx \mathbf{x}^{t+1} = \mathbf{x}^t + \Delta\mathbf{x}. \tag{3.15}$$

After that, the Taylor expansion can be applied to $r_i(\mathbf{x})$ and one obtains the so-called *linearised residual*

$$r_i(\mathbf{x}) \approx T_{1,r_i}\left(\mathbf{x}, \mathbf{x}^t\right) = r_i\left(\mathbf{x}^t\right) + \sum_{j=0}^{n} \frac{\partial r_i(\mathbf{x}^t)}{\partial \mathbf{x}_j}\left(\mathbf{x}_j - \mathbf{x}_j^t\right) \tag{3.16}$$

$$= r_i\left(\mathbf{x}^t\right) + \sum_{j=0}^{n} \left(\mathbf{J}_r(\mathbf{x}^t)\right)_{i,j} \Delta\mathbf{x}_j, \tag{3.17}$$

where the second equality follows from the definition of the Jacobian 3.13 and from the Equation 3.15. $\left(\mathbf{J}_r(\mathbf{x}^t)\right)_{i,j}$ is the entry of the Jacobian of $r(\mathbf{x}^t)$ at row i and column j. For the sake of readability it is abbreviated as $\mathbf{J}_{i,j}$ and $\mathbf{J}_r(\mathbf{x}^t)$ is equivalent to $\mathbf{J}$. The partial derivative of the linearised residual with respect to $\mathbf{x}_j$ is then

$$\frac{\partial r_i(\mathbf{x})}{\partial \mathbf{x}_j} = \mathbf{J}_{i,j}. \tag{3.18}$$

For all $j \in 1, ..., n$, plugging 3.18 and 3.17 into Equation 3.10 yields

$$2\sum_{i=1}^{m} \mathbf{J}_{i,j}\left(r_i(\mathbf{x}^t) + \sum_{k=1}^{n} \mathbf{J}_{i,k}\Delta\mathbf{x}_k\right) = 0 \tag{3.19}$$

$$\Leftrightarrow 2\sum_{i=1}^{m}\sum_{k=1}^{n} \mathbf{J}_{i,j}\mathbf{J}_{i,k}\Delta\mathbf{x}_k = -2\sum_{i=1}^{m} \mathbf{J}_{i,j} r_i(\mathbf{x}^t). \tag{3.20}$$

Using matrix notation one obtains

$$2\mathbf{J}^\top \mathbf{J}\Delta\mathbf{x} = -2\mathbf{J}^\top r\left(\mathbf{x}^t\right). \tag{3.21}$$

Note that the last equation is linear in the parameter Δx. In consequence, one gets an LSE which can be solved with the algorithm explained in the next section.

3.3 Conjugate Gradient Method

Conjugate gradient (CG) was originally proposed by Hestenes and Stiefel [57] in 1952. It is a method to numerically minimize the quadratic function

$$g(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} - \mathbf{b}^\top \mathbf{x}. \tag{3.22}$$

If the matrix $\mathbf{A}$ is positive definite this is equivalent to solving Equation 3.3.

As initialisation one chooses a starting point $\mathbf{x}_1$ for the unknown $\mathbf{x}$. Afterwards, the error $\mathbf{e}^1 \in \mathbb{R}^n$ at time one is computed as $\mathbf{e}^1 = \mathbf{b} - \mathbf{A}\mathbf{x}^1$ and one defines $\hat{\mathbf{e}}^1 = \mathbf{e}^1$. Furthermore, one sets $\mathbf{p}^1 = \mathbf{e}^1$ and $\hat{\mathbf{p}}^1 = \hat{\mathbf{e}}^1$. Afterwards, the iterations start and execute the variable updates

$$\begin{aligned}
\alpha^t &= \frac{\hat{\mathbf{e}}^t \mathbf{e}^t}{\hat{\mathbf{p}}^t \mathbf{A}\mathbf{p}^t} \\
\mathbf{e}^{t+1} &= = \mathbf{e}^t - \alpha^t \mathbf{A}\mathbf{p}^t \\
\hat{\mathbf{e}}^{t+1} &= \hat{\mathbf{e}}^t - \alpha^t \mathbf{A}^\top \hat{\mathbf{p}}^t \\
\beta^t &= \frac{\hat{\mathbf{e}}^{t+1} - \mathbf{e}^{t+1}}{\hat{\mathbf{e}}^t \mathbf{e}^t} \\
\mathbf{p}^{t+1} &= \mathbf{e}^{t+1} + \beta^t \mathbf{p}^t \\
\hat{\mathbf{p}}^{t+1} &= \hat{\mathbf{e}}^{t+1} + \beta^t \hat{\mathbf{p}}^t \\
\mathbf{x}^{t+1} &= \mathbf{x}^t + \alpha^t \mathbf{p}^t,
\end{aligned} \tag{3.23}$$

where $\hat{\mathbf{e}}^1, \mathbf{p}^1, \hat{\mathbf{p}}^1, \mathbf{e}^t, \hat{\mathbf{e}}^t, \mathbf{p}^t, \hat{\mathbf{p}}^t, \mathbf{e}^{t+1}, \hat{\mathbf{e}}^{t+1}, \mathbf{p}^{t+1}, \hat{\mathbf{p}}^{t+1} \in \mathbb{R}^n$ and $\alpha^t, \beta^t \in \mathbb{R}$. In the last line the current guess is refined. The outline of the CG algorithm is partially based on the version of the book *Numerical Recipes* [58]. Figure 3.2 shows a Conjugate Gradient optimization. With this method one can now solve the linear system of equations shown in Equation 3.21 with respect to $\Delta\mathbf{x}$ by choosing the variable assignment

$$\begin{aligned}
\mathbf{A} &= 2\mathbf{J}^\top \mathbf{J} \\
\mathbf{x} &= \Delta\mathbf{x} \\
\mathbf{b} &= -2\mathbf{J}^\top r\left(\mathbf{x}^t\right).
\end{aligned} \tag{3.24}$$

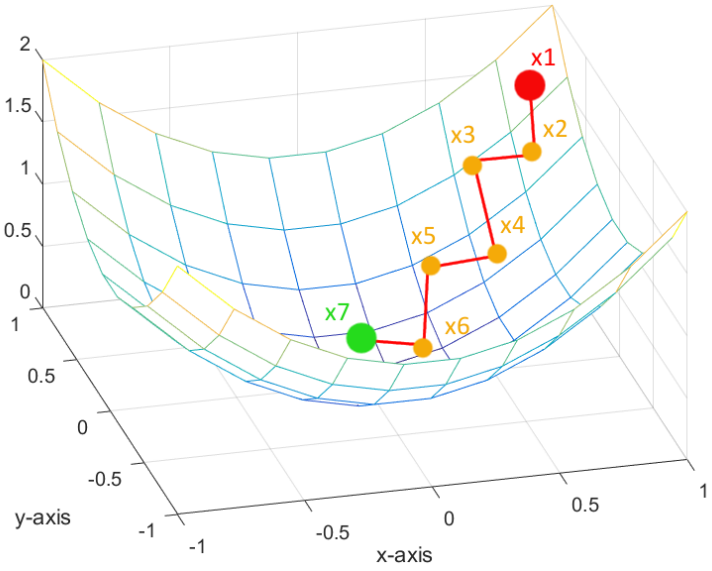

Figure 3.2: Visualization of the Conjugate Gradient method for the 2D function $f(x, y) = x^2 + y^2$ that corresponds to setting $\mathbf{A} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. The red dot marks the initial guess. The orange ones are intermediate results and the green point marks the final solution.

As a result, one has a solution for the linear approximation of the original non-linear system. But to solve the later one, another step has to be taken, which is the content of the next section.

3.4 Gauss-Newton Method

To get the final solution one can use the so-called Gauss-Newton method that is also greatly explained by Stephen Boyd and Lieven Vandenberghe [59]. The idea behind is that one can reformulate Equation 3.21 as

$$\Delta\mathbf{x} = -\left(\mathbf{J}^\top\mathbf{J}\right)^{-1}\mathbf{J}^\top r\left(\mathbf{x}^t\right). \tag{3.25}$$

Plugging this into Equation 3.15 yields

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \left(\mathbf{J}^\top\mathbf{J}\right)^{-1}\mathbf{J}^\top r\left(\mathbf{x}^t\right). \tag{3.26}$$

Since one already knows from the previous section what the value of $\Delta\mathbf{x}$ is, this boils down to a simple addition, also called *Gauss-Newton step*. Nevertheless, the non-linearity of the thesis' cost $e(\mathbf{x})$ also introduces some difficulties since one wants to find the global optimum, but the function can have multiple local minima where the

algorithm can stuck. For example, the function shown in Figure 3.2 does not have such an issue since it is convex meaning that there is only one minimum. In contrast, the function shown in Figure 3.1 has two minima. Depending on the initialization one can obtain different solutions using the CG method. While choosing $x_1 = o$ leads to the global optimum g, the initialization $x_1 = w$ only finds a local minimum l. Thus, one should choose an initial guess close to the global solution. Assuming that the initialization fulfils this requirement, the thesis' non-linear energy function, which will be explained in the next chapter, can be solved using the above concepts.

3.5 Camera Model

Since RONDA uses camera projections for example to compute the photometric error, the underlying perspective camera model should be explained that is similar to the one discussed in the book *Fundamentals of Computer Graphics* [60]. Without loss of generality it is assumed that the device is placed at the global origin and the viewing direction, also called Principal Axis, coincides with the global z-axis. Because of this and the fact that the other two axis are also parallel to the corresponding global ones, the Camera Coordinate System is equal to the Global Coordinate System and one can get rid of the Extrinsic Camera Matrix. It remains to introduce the Intrinsic Camera Matrix

$$\mathbf{P} = \begin{pmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.27}$$

where the entries are a defined as

$$\begin{aligned} \alpha_u &= \frac{\hat{f}}{s_u} r_u \\ \alpha_v &= \frac{\hat{f}}{s_v} r_v \\ u_0 &= o_x r_u + \frac{r_u}{2} \\ v_0 &= o_y r_v + \frac{r_v}{2} \\ \gamma &= p_\mathrm{a} r_u. \end{aligned} \tag{3.28}$$

$s_u, s_v \in \mathbb{R}$ are the size of the camera sensor along the u- and v-direction. $r_u, r_v \in \mathbb{R}$ form the resolution of the video. $o_x, o_y \in \mathbb{R}$ are the x- and y-coordinates of the Principal Point in the Camera Coordinate System. It is defined as the location where the Principal Axis and the Image Plane intersect. Usually, it is assumed that

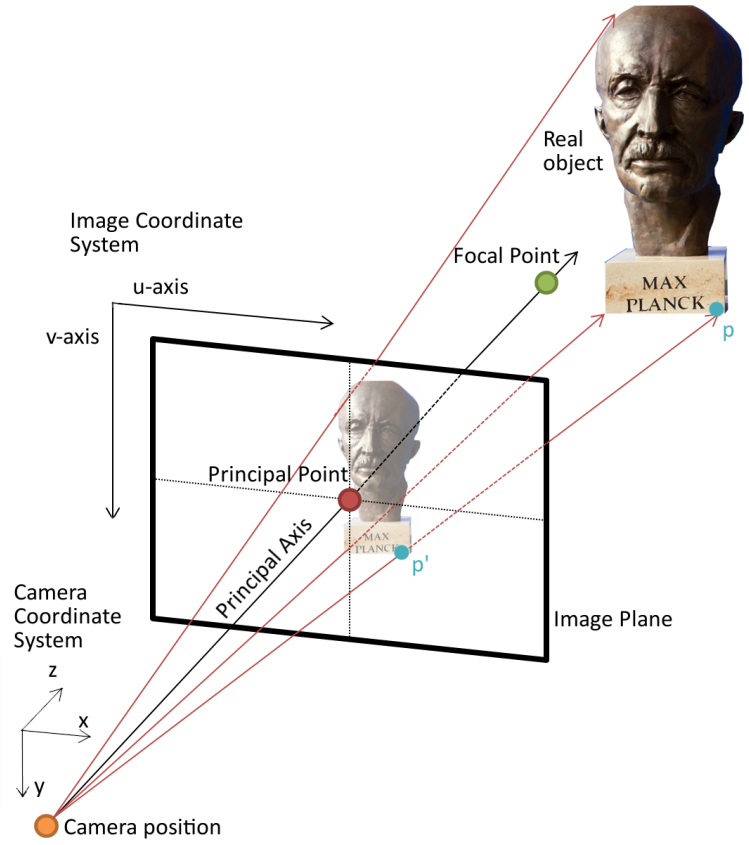

Figure 3.3: The perspective camera model that illustrates the components of the Intrinsic Camera Matrix. The blue dots visualize the projection $\pi(\mathbf{p})$ of the point $\mathbf{p} \in \mathbb{R}^3$ at the real object onto the Image plane at location $\mathbf{p}' \in \mathbb{R}^2$.

the point $(u_0, v_0)^\top$ lies at the center of the image. Thus, o_x and o_y are set to zero. $\hat{f} \in \mathbb{R}$ is the focal length of the camera and it is defined as the distance between the Focal Point and the Principal Point. Finally, $p_\mathrm{a} \in \mathbb{R}$ is the Pixel Aspect which can be computed as the ratio of the pixel width to the pixel height. The above terms are visualized in Figure 3.3.

Having this matrix, the projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ can be defined as

$$\pi(\mathbf{p}) = \begin{pmatrix} \pi_1(\mathbf{p}) \\ \pi_2(\mathbf{p}) \end{pmatrix} = \begin{pmatrix} \frac{\alpha_u \mathbf{p}_1 + \gamma \mathbf{p}_2 + u_0 \mathbf{p}_3}{\mathbf{p}_3} \\ \frac{\alpha_v \mathbf{p}_2 + v_0 \mathbf{p}_3}{\mathbf{p}_3} \end{pmatrix}. \tag{3.29}$$

It takes a 3D point $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)^\top$ and returns the corresponding position on the Image Plane (see Figure 3.3). Since the thesis makes use of the *OpenCV* library [61] it also follows their convention of the Image Coordinate System where the u-axis goes from left to right and the v-axis goes from top to bottom, starting at the top-left corner.

3.6 Images, Image Gradients and Edges

The thesis' energy function operates on images. Therefore, they should be formally defined as well as their gradients since both are needed to compute the partial derivatives for certain parts of the cost terms. In the real world an image consists of two-dimensional continuous functions $\mathcal{I}_t(u, v) : \mathbb{R}^2 \to \mathbb{R}$. They take a 2D coordinate and return the corresponding value of a certain color channel $t \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}, \mathrm{S}\}$ representing the red, green and blue components or in case of a grey scale image, it returns the grey value. But the pictures, which people store on their computers, are just sampled and quantized versions of the real ones. The thesis represents a color image as three discrete functions $I_{\mathrm{R}}(i, j), I_{\mathrm{G}}(i, j), I_{\mathrm{B}}(i, j) : \mathbb{N}^2 \to \{0, ..., 255\}$, one for each color channel. In case of grey scale images, there is just a single function $I_{\mathrm{S}}(i, j) : \mathbb{N}^2 \to \{0, ..., 255\}$. They take a pixel position $(i, j) \in \mathbb{N}^2$ and return the quantized color or grey value ranging from zero to 255.

Since one has such a discrete version of an image, the gradient $\nabla\mathcal{I}(i, j) = \left(\frac{\partial\mathcal{I}_t(i,j)}{\partial u}, \frac{\partial\mathcal{I}_t(i,j)}{\partial v}\right)^\top$ cannot be computed analytically. Instead, it has to be approximated with the help of finite differences. The thesis uses

$$\begin{aligned} \frac{\partial\mathcal{I}_t(i, j)}{\partial u} &\approx \frac{I_t(i + 1, j) - I_t(i - 1, j)}{2h_u} = I_{t_u}(i, j) \\ \frac{\partial\mathcal{I}_t(i, j)}{\partial v} &\approx \frac{I_t(i, j + 1) - I_t(i, j - 1)}{2h_v} = I_{t_v}(i, j) \end{aligned} \tag{3.30}$$

as gradient approximation $\nabla I_t(i, j) = \left(\frac{\partial I_{t_u}(i,j)}{\partial u}, \frac{\partial I_{t_v}(i,j)}{\partial v}\right)^\top$ in the u- and v-direction of the image $\mathcal{I}_t$ at position $(i, j) \in \mathbb{N}^2$, which agrees with the definitions in the literature [62, 63]. $h_u, h_v \in \mathbb{R}$ are the grid sizes along the corresponding directions that are assumed to be one. The gradient magnitude can then be defined as $\|\nabla I_t(i, j)\|$.

Since the edges or line-like patterns in an image are one of the key components of this work, it is natural to ask how they can be detected. An intuitive definition of an edge would be that these are areas where a high difference is present between the values of neighbouring pixels. Thus, if a pixel is an edge then the gradient at that location should have a maximum or a minimum. In different words, the gradient magnitude has to be larger than a predefined threshold.

3.7 Image Smoothing

Smoothing is a fundamental operation on images and an adequate strategy to remove noise and to create a gradient direction in flat regions. Hence, the thesis also applies a blur operation over the frames. The math behind is a discrete 2D convolution of a kernel matrix $\mathbf{K} \in \mathbb{R}^{w \times w}$ with the image function I_t. The convolution operator $*$ is defined as

$$(I_t * \mathbf{K})(i,j) = \sum_{k=1}^{w} \sum_{l=1}^{w} \mathbf{K}_{k,l} I_t(i+k-w+1), j+l-w+1) \tag{3.31}$$

and the convolution of two matrices is

$$(\mathbf{A} * \mathbf{K})(i,j) = \sum_{k=1}^{w} \sum_{l=1}^{w} \mathbf{K}_{k,l} \mathbf{A}_{(i+k-w+1, j+l-w+1)}. \tag{3.32}$$

It is assumed that $w \in \mathbb{N}$ is an odd number. To smooth the image, the Gaussian kernel matrix $\mathbf{G}_w$ is used where the underlying 2D Gaussian distribution has zero mean and a standard deviation

$$\sigma = \frac{3}{10}\left(\frac{w}{2} - 1\right) + \frac{8}{10} \tag{3.33}$$

based on the *OpenCV* implementation of the smoothing function. The entries of $\mathbf{G}_w$ are the sampled values of this distribution. At image regions where the kernel goes beyond the boundary, imaginary pixels are created by mirroring the existing ones along the boundary axis. The same principle holds for the case of two matrices.

Coming back to image gradients one can now rewrite the gradient approximation as

$$I_{t_u}(i,j) = (I_t * \mathbf{D}_u)(i,j) \tag{3.34}$$

$$I_{t_v}(i,j) = (I_t * \mathbf{D}_v)(i,j), \tag{3.35}$$

where

$$\mathbf{D}_u = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{D}_v = \frac{1}{2}\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \tag{3.36}$$

3.8 Sobel Operator

Usually, the problem with the above derivative approximations is that they smooth along the u- and v-direction, but not perpendicular to it. In addition, they are only able to detect very narrow edges. To compute the gradient without these issues, one can convolve the image with Desai's modified version [64] of the *Sobel operators* [65] $\mathbf{S}_{u_w}$, $\mathbf{S}_{v_w}$ which are defined as

$$\mathbf{S}_{u_w} = \mathbf{L}_v \mathbf{D}_{u_w} \tag{3.37}$$

$$\mathbf{S}_{v_w} = \mathbf{D}_{v_w} \mathbf{L}_u, \tag{3.38}$$

where $\mathbf{D}_{u_w} \in \mathbb{R}^{1\times w}$ and $\mathbf{D}_{v_w} \in \mathbb{R}^{w\times 1}$ are

$$\mathbf{D}_{u_w} = \begin{pmatrix} -1 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}, \mathbf{D}_{v_w} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \tag{3.39}$$

and $\mathbf{L}_u \in \mathbb{R}^{1\times 3}$, $\mathbf{L}_v \in \mathbb{R}^{3\times 1}$ are

$$\mathbf{L}_u = \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}, \mathbf{L}_v = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}. \tag{3.40}$$

$w \in \mathbb{N}$ has to be an odd number and defines the kernel size along the corresponding direction. If one has smooth edges instead of sharp ones, w can be adapted such that they can also be detected.

3.9 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) was first proposed by Dalal and Triggs [66], but in the context of pedestrian detection. HOG can be seen as a feature descriptor for images that analyses the local gradient directions. The thesis uses a modified version of the original approach and computes for each pixel of an input image a corresponding histogram which represents the feature. The algorithm is described in the following paragraph.

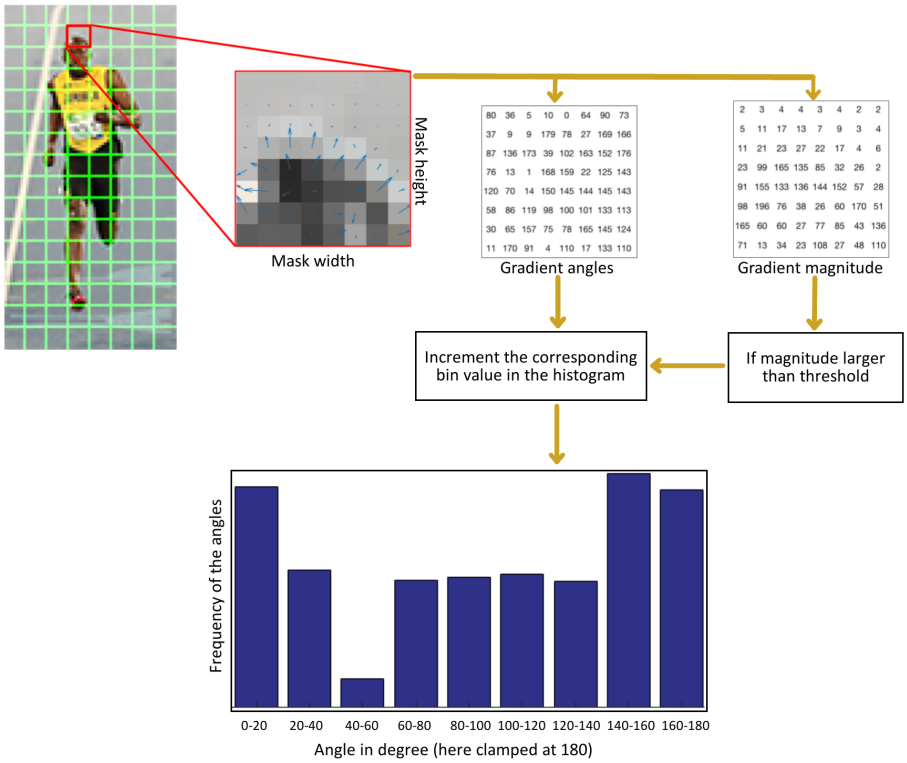

Figure 3.4: Overview of Histogram of Oriented Gradients. The blue arrows represent the gradients and the length corresponds to their magnitude. In this particular example the angles are only counted from zero to 180 and since there are nine bins, each of them has a range of 20 degrees. Parts of the graphic were retrieved from [67].

An overview of the method is shown in Figure 3.4. First, the color image I_{R}, I_{G}, I_{B} is converted into a grey scale one I_{S}. For a pixel (i, j), the *mask region* M is defined as

$$M_{i,j} = \{(x, y) \in \mathbb{N}^2 | i - m_u \leq x \leq i + m_u \wedge j - m_v \leq y \leq j + m_v\}, \tag{3.41}$$

where $m_u, m_v \in \mathbb{R}$ are the mask size. Now, the image gradients for each element (x, y) of $M_{i,j}$ can be computed with the Sobel operator

$$\begin{pmatrix} I_{\mathrm{S}_u}(x, y) \\ I_{\mathrm{S}_v}(x, y) \end{pmatrix} = \begin{pmatrix} (\mathbf{S}_{u_w} * I_{\mathrm{S}})(x, y) \\ (\mathbf{S}_{v_w} * I_{\mathrm{S}})(x, y) \end{pmatrix}, \tag{3.42}$$

where $w \in \mathbb{R}$ is again the kernel size as defined above. The corresponding gradient angle is

$$\alpha = \tan^{-1}\left(\frac{I_{S_v}(x, y)}{I_{S_u}(x, y)}\right) \tag{3.43}$$

and the respective gradient magnitudes can also be calculated. After that, a histogram is created, which has b bins such that it divides the 360 degrees into b pieces.

For example, if one has $b = 6$, then each bin covers 60 degrees and the first bin contains the angles between zero and 60 degrees. If the gradient magnitude is larger than a certain threshold value, the frequency of the bin that falls into the range of the corresponding angle, is incremented. This thresholding step is applied because of the fact that small gradient magnitudes often belong to noise. In consequence, they are not meaningful for the histogram. At the end, one has for each pixel (i, j) in the image a histogram $\mathbf{h}_{i,j} \in \mathbb{R}^b$ that contains the values of the bins. This gives an estimate of the frequencies of the gradient angles around (i, j) in a vectorized form.

Chapter 4

RONDA - Estimating Non-rigid Surface Motion

After the mathematical ingredients are stated, the thesis' contributions can be introduced. In the following chapter a solution for tracking general non-rigid deformations is proposed. First, a formal description of the problem will be given and notations are defined. After that, the different assumptions and the resulting constraints are explained which together form the energy function.

4.1 Formal Description of the Problem

RONDA's goal is to estimate the non-rigid deformation of an object from a single RGB video. Since this problem is in general severely under-constrained, it is assumed that a template mesh is given. Hence, a formal definition and the notations of the terms *video*, *template mesh* and *deformation* are necessary to mathematically state the objective.

Video. One kind of input is the video showing the deformations of the object. It consists of T frames where each one is represented as three image functions $I^t_{\mathrm{R}}(i,j)$, $I^t_{\mathrm{G}}(i,j)$ and $I^t_{\mathrm{B}}(i,j)$ corresponding to the color channel and the time $t \in \{1,...,T\}$. As stated in the previous chapter, the camera does not move and the Extrinsic Camera Matrix is the identity. Furthermore, it is assumed that the intrinsic parameters of the recording device are known such that the Intrinsic Camera Matrix $\mathbf{P}$ as well as the projection function $\pi(\mathbf{p})$ for a point $\mathbf{p} \in \mathbb{R}^3$ can be computed.

Template mesh. The template mesh $\hat{\mathbf{V}} \in \mathbb{R}^{N\times 3}$, which has N vertices is defined as the matrix

$$\hat{\mathbf{V}} = \begin{pmatrix} \hat{\mathbf{V}}_{1,1} & \hat{\mathbf{V}}_{1,2} & \hat{\mathbf{V}}_{1,3} \\ \hat{\mathbf{V}}_{2,1} & \hat{\mathbf{V}}_{2,2} & \hat{\mathbf{V}}_{2,3} \\ \vdots & \vdots & \vdots \\ \hat{\mathbf{V}}_{N,1} & \hat{\mathbf{V}}_{N,2} & \hat{\mathbf{V}}_{N,3} \end{pmatrix}, \tag{4.1}$$

where each row contains the coordinates of one mesh point. According to that

$$\hat{\mathbf{V}}_i = \begin{pmatrix} \hat{\mathbf{V}}_{i,1} \\ \hat{\mathbf{V}}_{i,2} \\ \hat{\mathbf{V}}_{i,3} \end{pmatrix} \tag{4.2}$$

is defined as the ith vertex of the template in vector form. This notation is also used for the following matrices. Furthermore, it is assumed that the geometry at time one roughly agrees with the true shape shown in the video. Without losing expressive power, the mesh has to be triangulated, which means that each of the F faces is a triangle. The edges of the template are implicitly given as the mapping $\mathcal{N}(i)$. It takes the index $i \in \{1, 2, ..., N\}$ of a certain vertex $\hat{\mathbf{V}}_i$ and returns the set of indices sharing an edge with $\hat{\mathbf{V}}_i$. The faces of the mesh are represented as the matrix

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_{1,1} & \mathbf{F}_{1,2} & \mathbf{F}_{1,3} \\ \mathbf{F}_{2,1} & \mathbf{F}_{2,2} & \mathbf{F}_{2,3} \\ \vdots & \vdots & \vdots \\ \mathbf{F}_{F,1} & \mathbf{F}_{F,2} & \mathbf{F}_{F,3} \end{pmatrix}, \tag{4.3}$$

where $\mathbf{F} \in \{1, ..., N\}^{F\times 3}$. Each row contains the vertex indices of one triangle. According to that $\mathbf{F}_j$ is the jth face of the mesh. Beside the geometry it is also assumed that the UV map

$$\mathbf{U} = \begin{pmatrix} \mathbf{U}_{1,1} & \mathbf{U}_{1,2} \\ \mathbf{U}_{2,1} & \mathbf{U}_{2,2} \\ \vdots & \vdots \\ \mathbf{U}_{N,1} & \mathbf{U}_{N,2} \end{pmatrix}, \tag{4.4}$$

where $\mathbf{U} \in \mathbb{N}^{N\times 2}$, is known. Each row contains the pixel coordinates for the corresponding vertex. Because of the fact that the texture map I_{TM} is also given, the color of vertex i can be computed by a simple look up at the position $\mathbf{U}_i$. These values are stored in the variables $\mathbf{c}_{\mathrm{R}_i}, \mathbf{c}_{\mathrm{G}_i}, \mathbf{c}_{\mathrm{B}_i} \in \{0, ..., 255\}$. Together they form the three vectors $\mathbf{c}_{\mathrm{R}}, \mathbf{c}_{\mathrm{G}}, \mathbf{c}_{\mathrm{B}} \in \{0, ..., 255\}^N$, which contain the colors of all mesh points.

Deformation. Non-rigid deformations can be represented as vector fields for each time step. They contain the 3D displacements for the vertices going from frame t to $t+1$. Instead of computing them in explicit form, the thesis directly estimates the resulting geometry after adding the vector field to the old mesh position. According to that the shape at time $t+1$ is

$$\mathbf{V}^{t+1} = \begin{pmatrix} \mathbf{V}_{1,1}^{t+1} & \mathbf{V}_{1,2}^{t+1} & \mathbf{V}_{1,3}^{t+1} \\ \mathbf{V}_{2,1}^{t+1} & \mathbf{V}_{2,2}^{t+1} & \mathbf{V}_{2,3}^{t+1} \\ \vdots & \vdots & \vdots \\ \mathbf{V}_{N,1}^{t+1} & \mathbf{V}_{N,2}^{t+1} & \mathbf{V}_{N,3}^{t+1} \end{pmatrix}. \tag{4.5}$$

Statement of the problem. Given $\hat{\mathbf{V}}$ and the video frames, RONDA is interested in sequentially computing the geometry $\mathbf{V}^{t+1}$ at time $t+1$ for all $t \in \{1, ..., T\}$. It starts with $t = 1$ and ends with $t = T$. For each time step the deformation estimation can be formulated as the minimization problem

$$(\mathbf{V}^{t+1}, \mathbf{\Phi}^{t+1}) = \underset{\mathbf{V}, \mathbf{\Phi} \in \mathbb{R}^{N \times 3}}{\arg\min}\ e_{\text{Arap}}(\mathbf{V}, \mathbf{\Phi}) + \sum_{i=1}^{6} e_i(\mathbf{V}), \tag{4.6}$$

where $e_i(\mathbf{V}) : \mathbb{R}^{N \times 3} \to \mathbb{R}$, $e_{\text{Arap}}(\mathbf{V}, \mathbf{\Phi}) : \mathbb{R}^{N \times 3} \times \mathbb{R}^{N \times 3} \to \mathbb{R}$ are the cost terms. They help to ensure that the mesh deformations agree with the motion in the video by penalizing movements, which violate the underlying assumptions. $\mathbf{V}$, $\mathbf{\Phi}$ are the jointly optimized variables. $\mathbf{V}$ represents the mesh estimate and $\mathbf{\Phi}$ are local rotations needed for the function e_{Arap}. They will be explained later in more detail. One can see that the cost is split into seven terms. Six of them are the content of the remaining part of this chapter. Together they build a framework, which is a combination of existing approaches, such that general object deformations can be tracked.

4.2 Photometric Alignment

The overall goal is that the reconstructed geometry should match with the shape of the real object in the actual frame. Because one has no access to the ground truth, a direct measure of mesh differences cannot be applied. Nevertheless, the vertex color $(\mathbf{c}_{\text{R}_i}, \mathbf{c}_{\text{G}_i}, \mathbf{c}_{\text{B}_i})$ of the template is known and it is assumed that it remains constant. One can then measure the difference between $(\mathbf{c}_{\text{R}_i}, \mathbf{c}_{\text{G}_i}, \mathbf{c}_{\text{B}_i})$ and the color observed at the pixel location in the frame where $\mathbf{V}_i$ is projected onto. This builds a strong source of information, which can be used to indirectly measure the correctness of the estimation. Based on the work of Yu et al. [5], the thesis uses a cost term

$e_{\text{Photo}}(\mathbf{V}) : \mathbb{R}^{N\times 3} \to \mathbb{R}$ that minimizes these color differences. In the following, the penalty is also called *photometric alignment* and it is defined as

$$\begin{aligned} e_{\text{Photo}}(\mathbf{V}) = \sum_{i=0}^{N} & \left(\sigma_p \left(C_{\text{R}} \left(\pi \left(\mathbf{V}_i\right)\right) - \mathbf{c}_{\text{R}_i}\right)\right)^2 \\ & + \left(\sigma_p \left(C_{\text{G}} \left(\pi \left(\mathbf{V}_i\right)\right) - \mathbf{c}_{\text{G}_i}\right)\right)^2 \\ & + \left(\sigma_p \left(C_{\text{B}} \left(\pi \left(\mathbf{V}_i\right)\right) - \mathbf{c}_{\text{B}_i}\right)\right)^2. \end{aligned} \tag{4.7}$$

For every vertex $\mathbf{V}_i$, it consists of three parts where each corresponds to one of the color channels $c \in \{\text{R}, \text{G}, \text{B}\}$. $C_c\left(u, v\right) : \mathbb{R}^2 \to \{0, ..., 255\}$ is defined as

$$C_c\left(u, v\right) = \begin{cases} \left(I_c^{t+1} * \mathbf{G}_w\right)\left(\lceil u \rceil, \lceil v \rceil\right) & \text{, if } 0 \leq \lceil u \rceil < r_u \ \wedge \ 0 \leq \lceil v \rceil < r_v \\ \infty & \text{, else.} \end{cases} \tag{4.8}$$

The function takes the location on the image plane where $\mathbf{V}_i$ is projected onto. The pixel coordinates are obtained by rounding up the real valued 2D coordinates. If they do not lie inside the image, the function outputs infinity since no measure can be made. Otherwise, the color of the smoothed frame is returned where $\mathbf{G}_w$ is the Gaussian kernel with width w. The smoothing is applied for two reasons. On the one hand, it eliminates noise and on the other hand, it creates non-zero image gradients at regions where an edge between two flat areas is present. Because otherwise having zero image gradients results in vanishing partial derivatives and the solution of the optimization could not be refined as one can see in Figure 4.1. The function $\sigma_p(x) : \mathbb{R} \to \mathbb{R}$ is defined as

$$\sigma_p(x) = \begin{cases} x & \text{, if } |x| < p \\ 0 & \text{, else,} \end{cases} \tag{4.9}$$

where $p \in \{0, ..., 255\}$. It takes the color differences $\left(C_c\left(\pi\left(\mathbf{V}_i\right)\right) - \mathbf{c}_{c_i}\right)$ and thresholds it such that the 2D locations which lie outside of the image do not contribute to the solution. At the same time projections where the color differences are too large are also set to zero. For example if one defines $p = 200$, then all color differences above 200 are excluded from the minimization. The idea behind this is that occlusions often lead to such strong mismatches and the photometric alignment of occluded vertices obviously contains no information and therefore it should not be part of the optimization.

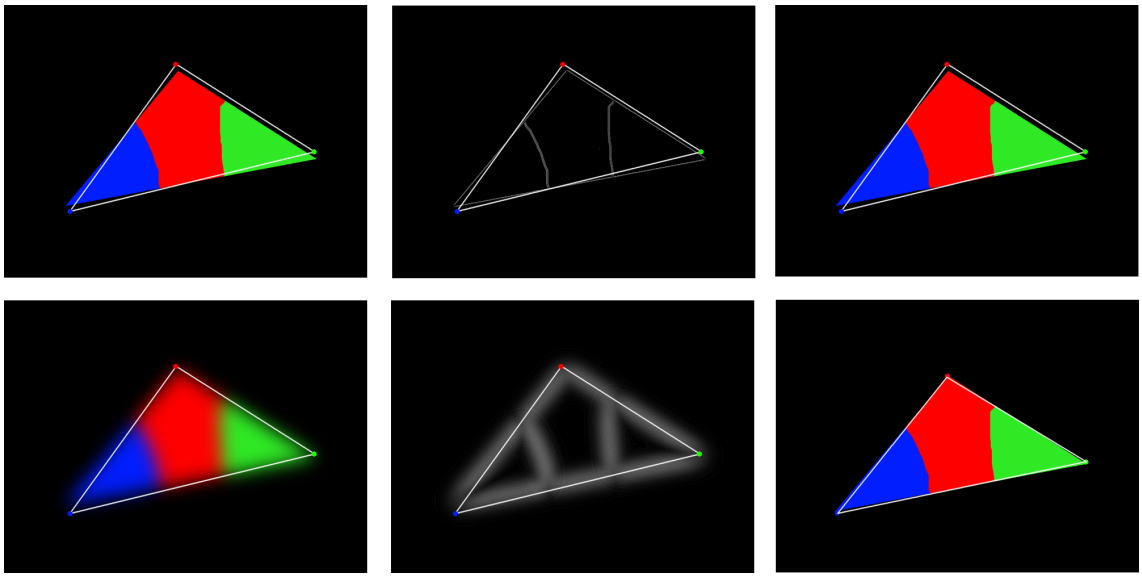

Figure 4.1: First column. The top graphic shows an input frame overlayed with the previous geometry estimate which does not perfectly project onto the triangle in the frame. On the bottom one can see the same frame smoothed with a Gaussian kernel. Middle column. Both pictures show the image gradient magnitude where the brightness increases with the magnitude, overlayed with the previous shape estimate. Right column. Geometry after the minimization. Note that the reconstruction of the unsmoothed image did not move because the image gradients were zero at the locations where the vertices are projected onto. Instead the reconstruction of the smoothed frame is close to the ground truth.

4.3 Laplacian Surface Deformation

The idea is that if a vertex $\mathbf{V}_i$ changes its position, his neighbours $\mathbf{V}_j$ with $j \in \mathcal{N}(i)$ should also be influenced such that the overall shape is still spatially smooth compared to the template mesh $\hat{\mathbf{V}}$. This is the case especially for clothes, but also for other objects like paper. The constraint $e_{\text{Laplacian}}(\mathbf{V}) : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}$ is therefore defined as

$$e_{\text{Laplacian}}(\mathbf{V}) = \sum_{i=0}^{N} \sum_{j\in\mathcal{N}(i)} \left\| (\mathbf{V}_i - \mathbf{V}_j) - (\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j) \right\|^2, \tag{4.10}$$

where $\|\cdot\|$ is the Euclidean norm. In the following, it is also called *Laplacian surface* constraint. The main advantage of this term, is that it can propagate the deformation estimate over multiple vertices (see Figure 4.2) and at poorly textured regions one can still reconstruct the shape, if the surrounding vertices deliver enough information.

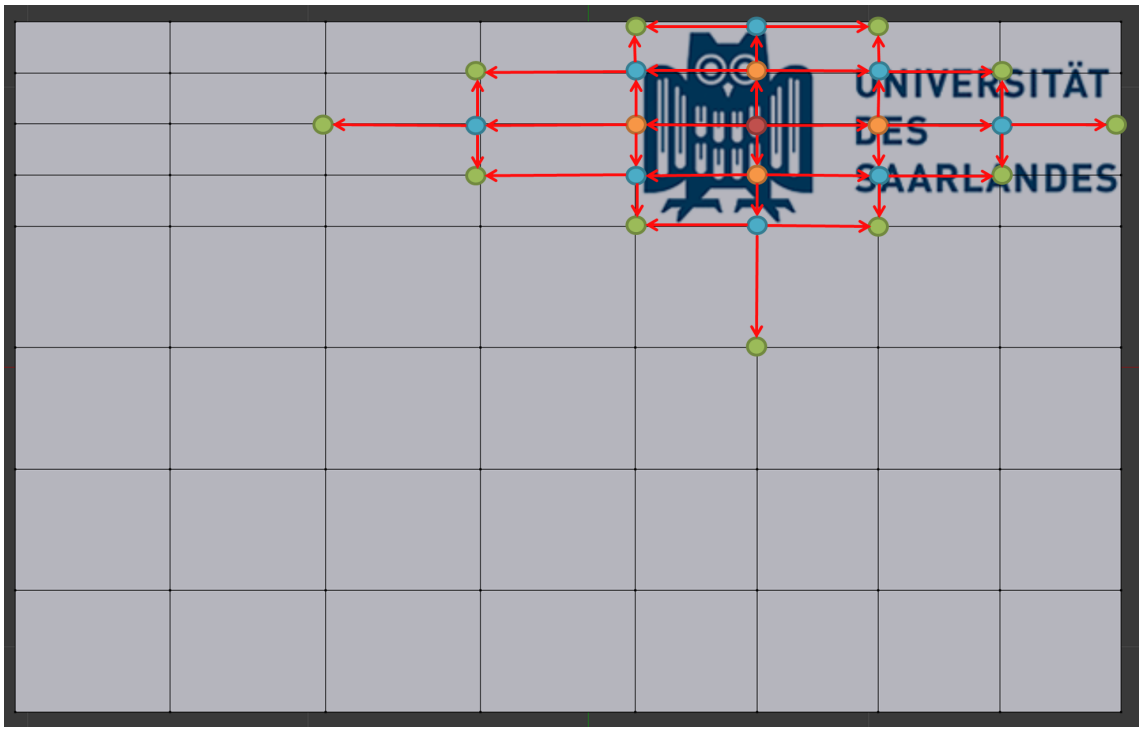

Figure 4.2: Information propagation of the Laplacian surface constraint. Only the upper right corner of the paper mesh contains texture information. In consequence, the photometric alignment can compute the deformations just in this area. Nevertheless, with the help of the Laplacian surface constraint the motion, e.g. translations, can be carried from the red dot to the non-textured parts of the mesh (lower green dots). The red arrows mark the deformation propagation.

4.4 Preservation of Edge Lengths

The next constraint is based on the assumption that the object has a non-flexible surface, which is for example the case for most woven fabrics. This implies that the edge lengths should be preserved compared to the template. The corresponding cost function $e_{\text{Edge}}(\mathbf{V}) : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}$ is defined as

$$e_{\text{Edge}}(\mathbf{V}) = \sum_{i=0}^{N} \sum_{j\in\mathcal{N}(i)} \left(\|\mathbf{V}_i - \mathbf{V}_j\| - \left\|\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j\right\| \right)^2. \tag{4.11}$$

This constraint can help to solve the previously mentioned scale ambiguity since it ensures that the overall mesh size remains constant. Figure 4.3 illustrates the influence of this cost term.

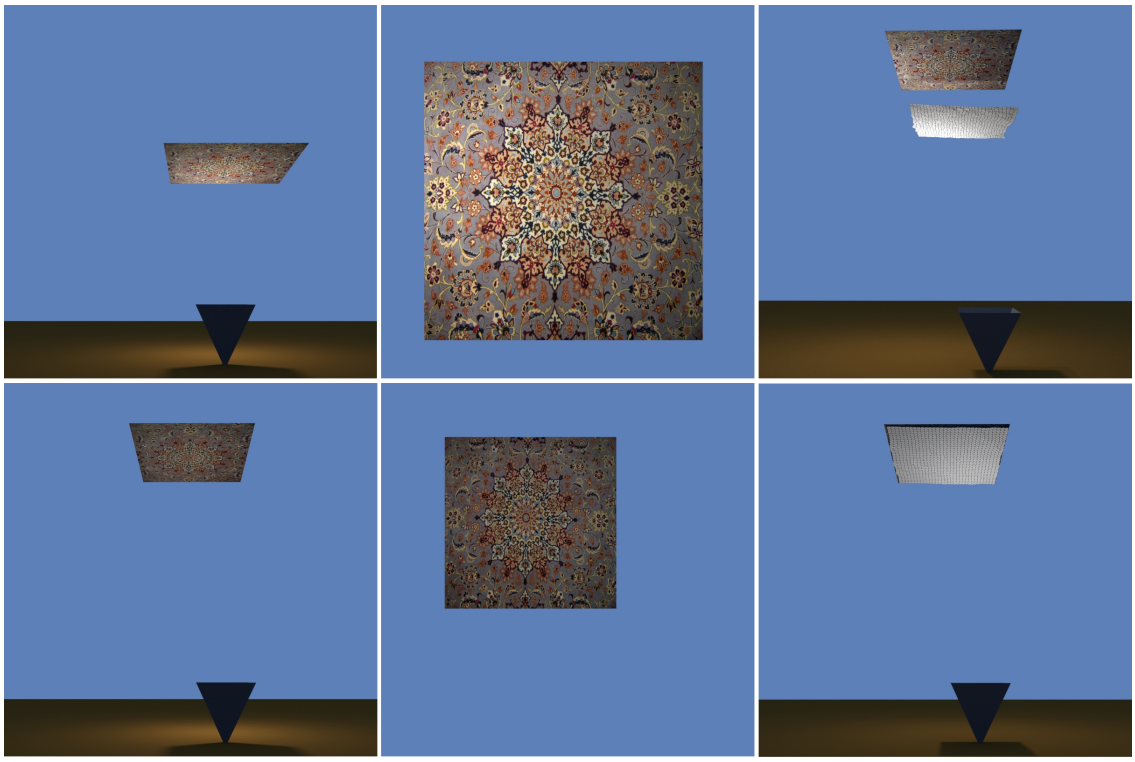

Figure 4.3: Influence of the edge length constraint. Left column. Side view of the first and last frame where the triangle represents the camera and the carpet is the tracked object. Middle column. First and last frame. Right column. On the top one can see the reconstruction of the object for the last frame without the edge length constraint and below the one which included it. Note that the upper estimate is far away from the ground truth since the geometry shrank whereas the one below keeps the original edge lengths and it therefore gives a more accurate result.

4.5 Smooth Motion

Since today's cameras can record videos with a high number of frames per second (fps), only a short time is passed between two consecutive images. In consequence, the displacement of the vertices between t and $t+1$ also has to be small. $e_{\text{Velocitiy}}(\mathbf{V}) : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}$ that is defined as

$$e_{\text{Velocitiy}}(\mathbf{V}) = \sum_{i=0}^{N} \left\| \mathbf{V}_i - \mathbf{V}_i^t \right\|^2, \tag{4.12}$$

ensures this by penalizing large motions. Hence, the search space for the vertex $\mathbf{V}_i$ can be reduced to a sphere with a small radius centered at the old position of the vertex $\mathbf{V}_i^t$. Another advantage is that it can also help to recover the shape at low textured regions since it does not depend on textural information.

4.6 Smooth Direction of Motion over Time

The next constraint is based on the assumption that the motion direction changes only smoothly over time. This is often the case since objects need a certain time to accelerate, to change the moving direction and to stop. Therefore, the cost term

$$e_{\text{Acceleration}}(\mathbf{V}) = \sum_{i=0}^{N} \left\| \left(\mathbf{V}_i - \mathbf{V}_i^t\right) - \left(\mathbf{V}_i^t - \mathbf{V}_i^{t-1}\right) \right\|^2 \tag{4.13}$$

penalizes large deviations of the actual moving direction from the previous one that is defined as the difference $\mathbf{V}_i^t - \mathbf{V}_i^{t-1}$.

4.7 As-rigid-as-possible Deformations

Finally, the as-rigid-as-possible (ARAP) constraint

$$e_{\text{Arap}}(\mathbf{V}, \mathbf{\Phi}) = \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \left\| (\mathbf{V}_i - \mathbf{V}_j) - \mathcal{R}_i(\mathbf{\Phi})(\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j) \right\|^2, \tag{4.14}$$

which is also used by Yu et al. [5] and Zollhöfer et al. [1], allows local rotations for each of the mesh vertices as long as the relative position with respect to their neighbourhood remains constant. The local rotation function

$$\mathcal{R}_i(\mathbf{\Phi}) = \begin{pmatrix} \mathcal{R}_{i,1,1}(\mathbf{\Phi}) & \mathcal{R}_{i,1,2}(\mathbf{\Phi}) & \mathcal{R}_{i,1,3}(\mathbf{\Phi}) \\ \mathcal{R}_{i,2,1}(\mathbf{\Phi}) & \mathcal{R}_{i,2,2}(\mathbf{\Phi}) & \mathcal{R}_{i,2,3}(\mathbf{\Phi}) \\ \mathcal{R}_{i,3,1}(\mathbf{\Phi}) & \mathcal{R}_{i,3,2}(\mathbf{\Phi}) & \mathcal{R}_{i,3,3}(\mathbf{\Phi}) \end{pmatrix} \tag{4.15}$$

$$= \begin{pmatrix} \cos(\theta)\cos(\phi) - \sin(\theta)\cos(\beta)\sin(\phi) & \sin(\theta)\cos(\phi) + \cos(\theta)\cos(\beta)\sin(\phi) & \sin(\beta)\sin(\phi) \\ -\cos(\theta)\sin(\phi) - \sin(\theta)\cos(\beta)\cos(\phi) & -\sin(\theta)\sin(\phi) + \cos(\theta)\cos(\beta)\cos(\phi) & \sin(\beta)\cos(\phi) \\ \sin(\theta)\sin(\beta) & -\cos(\theta)\sin(\beta) & \cos(\beta) \end{pmatrix} \tag{4.16}$$

per vertex i is derived through the Euler angles [68]. θ, β and ϕ are the rotations according to the x-, y- and z-axis such that each local transformation has three parameters. For the sake of readability the index i was dropped. More formally, one has $\theta = \theta_i$, $\beta = \beta_i$ and $\phi = \phi_i$. These parameters for all vertices are then stored in the matrix

$$\mathbf{\Phi} = \begin{pmatrix} \theta_1 & \beta_1 & \phi_1 \\ \vdots & \vdots & \vdots \\ \theta_N & \beta_N & \phi_N \end{pmatrix} \tag{4.17}$$

that has a size of $N \times 3$.

Chapter 5

RONDA - Non-rigid Motion of Woven Fabrics

5.1 Idea

The first question that arises is: Why should one focus on fabrics? The answer is that while their was a huge success in the field of human motion capture, the recovered animations of clothes still appear synthetic, if one looks for example at computer games. At the same time, fabrics have some nice properties such that reconstruction algorithms can make use of them as explained in the following. Inspired by that demand and these characteristics, the thesis presents a novel texture term, which can be combined with the previous energy functions to refine the estimation of non-rigid motions for the case of woven fabrics.

Therefore, a closer look was taken to the characteristics of clothes to build an additional constraint. Most of the studied garments show line-like structures caused by the manufacturing process. Since they are very small one was not able to capture them at large scale in the past. But due to technological developments, the video resolution of the recording devices increases steadily and nowadays, the textural lines can be seen if one zooms into a video. Figure 5.1 illustrates the difference between low and high resolution videos regarding the richness of details.

So, these patterns can now be captured. But to really compute a texture direction and how they can be of use, previous approaches [3, 69, 2, 70] were examined and parts of their concepts are adapted to form the proposed energy function. In the next section, the algorithm, which finds the directions of textures, is explained for the case of line-like patterns.

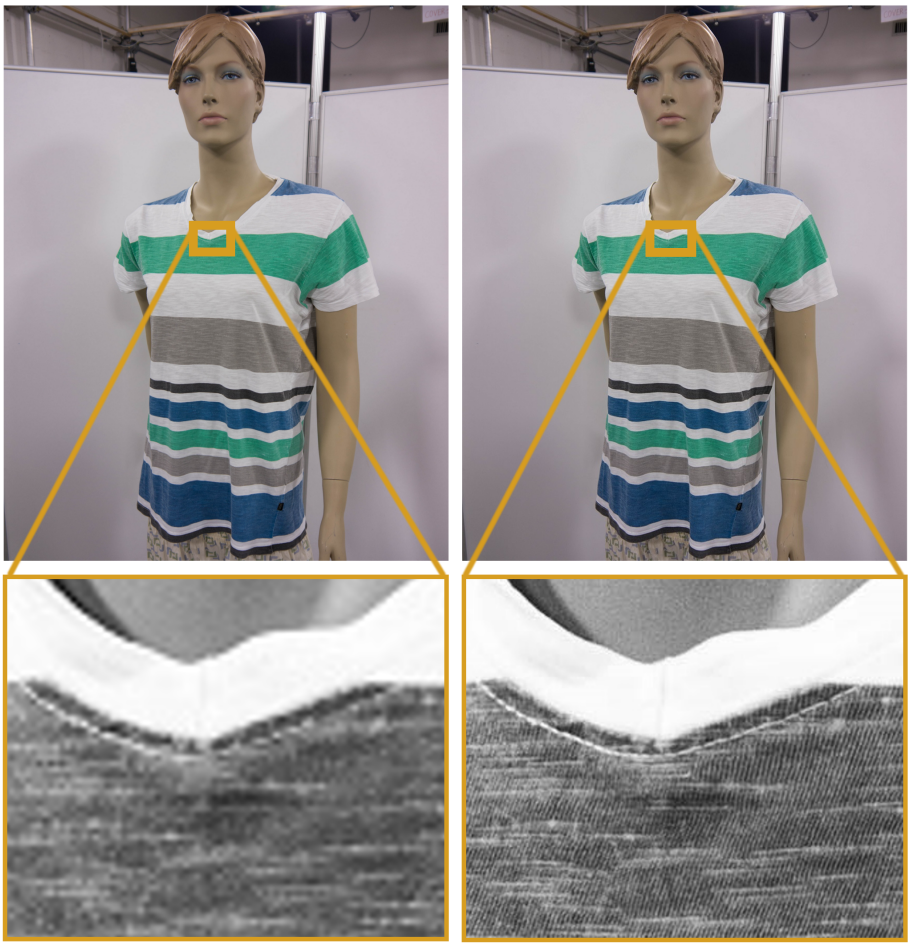

Figure 5.1: Comparison between low and high resolution frames regarding the richness of details. Top left. Downscaled image with a resolution of 853 × 1065. Top right. Original image with a resolution of 3409 × 4257. Bottom. The corresponding zoomed versions, which are converted to grey to better visualize the details. Note that while the line patterns are clearly visible at high resolution, they cannot be seen at low resolution.

5.2 Computation of Directions of a Texture

In Chapter 3, the concept of HOG was introduced that provides a tool to describe the local texture directions where each pixel (i, j) of an input image is related to a histogram $\mathbf{h}_{i,j}$ of gradient angles. This method can also be applied to pictures of fabrics. In the following, it is assumed that the number of histogram bins is 360, such that each bin belongs to one degree. Furthermore, the angle direction is clockwise, where zero degree is parallel to the u-axis. An example is shown in Figure 5.2. Here, the special characteristic of woven fabrics becomes visible. Caused by the line-like patterns, there are two more emerging texture gradient angles α and β, which are perpendicular to the lines. The histogram in Figure 5.2 validates this since there

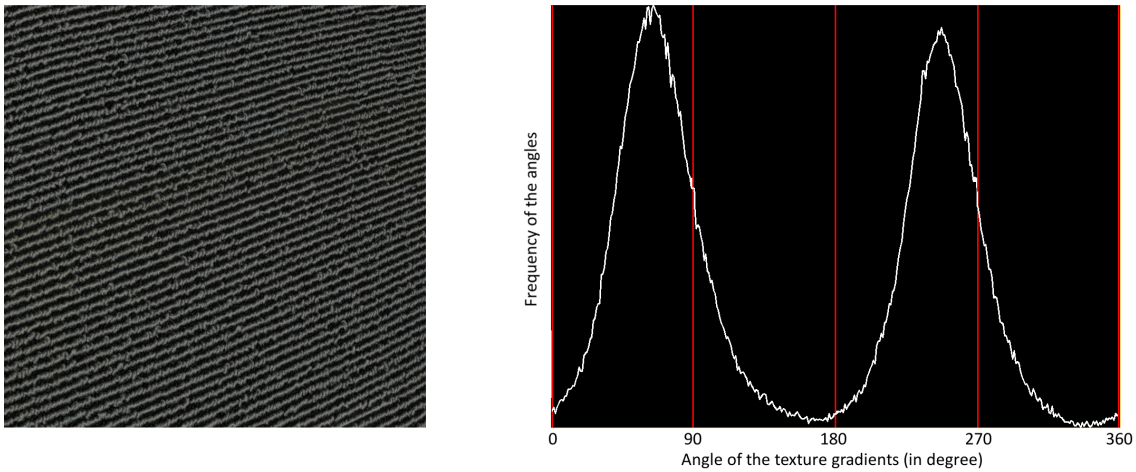

Figure 5.2: Histogram of Oriented Gradients of woven fabrics. Left. The mask region with the center pixel in the middle. Right. The corresponding histogram.

are two maxima, one at about 75 degree and one at 255 degree. These are exactly the angles orthogonal to the one of the line patterns (165 degree).

In the following, only these two so-called *dominant frame angles* are of interest because they provide the most characteristic information of the pattern in the input image at (i, j). Since, β can be computed out of α by taking $((\alpha + 180) \mod 360)$ and vice versa, only one of the two values has to be stored. Therefore, one can compute the dominant frame angle $\alpha_{\max,i,j}$ for (i, j) as

$$\alpha_{\max,i,j} = \begin{cases} \text{maxLoc}\left(\mathbf{h}_{i,j}\right) & \text{, if } p < \max\left(\mathbf{h}_{i,j}\right) \\ \infty & \text{, else,} \end{cases} \tag{5.1}$$

where $p \in \mathbb{N}$ is a predefined threshold parameter. The maxLoc-operator returns the position of the bin having the highest frequency and the max operator determines the largest value of the histogram vector. The thresholding is applied such that flat regions do not return a dominant angle because they indeed have none. After that, $\alpha_{\max,i,j}$ is converted to a 2D direction, which is referred to as the *dominant frame gradient* $\mathbf{d}_{\mathrm{F},i,j} \in \mathbb{R}^2$ that is defined as

$$\mathbf{d}_{\mathrm{F},i,j} = \begin{cases} \left(\cos\left(\alpha_{\max,i,j}\right), \sin\left(\alpha_{\max,i,j}\right)\right)^\top & \text{, if } \alpha_{\max,i,j} \leq 360 \\ \left(0, 0\right)^\top & \text{, else.} \end{cases} \tag{5.2}$$

Finally, these directions are stored in two images

$$I_{\mathrm{Dir},u}(i, j) = \left(\mathbf{d}_{\mathrm{F},i,j}\right)_1, I_{\mathrm{Dir},v}(i, j) = \left(\mathbf{d}_{\mathrm{F},i,j}\right)_2. \tag{5.3}$$

All in all, one has for each pixel (i, j) of an input image the corresponding 2D dominant frame gradient that is the most occurring direction in the mask region, stored in $I_{\mathrm{Dir},u}(i, j), I_{\mathrm{Dir},v}(i, j)$.

5.3 Derivation of the Texture-based Constraint

So far, the computation of the dominant frame gradients is known. Now, it will be explained how they can help to form a constraint. An overview of the method is given in Figure 5.3.

UV map and HOG. Since the UV coordinates $\mathbf{U}_k, \mathbf{U}_m, \mathbf{U}_l$ for each vertex of the triangle $\mathbf{F}_i = (k, m, l)$ are given, one can compute the pixel position $\mathbf{c}_{\mathrm{TM}} \in \mathbb{R}^2$ of the triangle's center point in the texture map (marked as blue dot in Figure 5.3) as

$$\mathbf{c}_{\mathrm{TM}} = \frac{1}{3}\mathbf{U}_k + \frac{1}{3}\mathbf{U}_m + \frac{1}{3}\mathbf{U}_l. \tag{5.4}$$

Now, the mask region around $\lceil \mathbf{c}_{\mathrm{TM}} \rceil$ is defined as the 2D bounding box of the triangle, where $\lceil \cdot \rceil$ rounds up all elements of a vector. The histogram of the center point in the texture map can be calculated and with the above concept one can determine the resultant dominant frame gradient $\mathbf{d}_{\mathrm{TM}}$.

Barycentric coordinates. By adding $\mathbf{d}_{\mathrm{TM}}$ to the center, the *barycentric point* $\mathbf{b}_{\mathrm{TM}} = \mathbf{c}_{\mathrm{TM}} + \mathbf{d}_{\mathrm{TM}}$ is obtained (marked in yellow in Figure 5.3). In general, one can take any point on the half-line $\mathbf{c}_{\mathrm{TM}} + s\mathbf{d}_{\mathrm{TM}}$ where $s > 0$. But the cost term which will be shown in the next section is independent of the choice of s due to a normalization and therefore, the thesis sets s to one. To express $\mathbf{b}_{\mathrm{TM}}$ as a linear combination of the triangles' UV coordinates, RONDA solves the equation

$$\mathbf{B}_{i,1}\mathbf{U}_k + \mathbf{B}_{i,2}\mathbf{U}_m + \mathbf{B}_{i,3}\mathbf{U}_l = \mathbf{b}_{\mathrm{TM}} \tag{5.5}$$

for the *barycentric coordinates* $\mathbf{B}_{i,1}, \mathbf{B}_{i,2}, \mathbf{B}_{i,3}$ of the face $\mathbf{F}_i$. They form together with the other triangles the barycentric coordinates matrix

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} & \mathbf{B}_{1,3} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} & \mathbf{B}_{2,3} \\ \vdots & \vdots & \vdots \\ \mathbf{B}_{F,1} & \mathbf{B}_{F,2} & \mathbf{B}_{F,3} \end{pmatrix}. \tag{5.6}$$

Each row represents the texture gradients for the respective triangle of the mesh in an implicit form, which becomes clearer in the next paragraph.

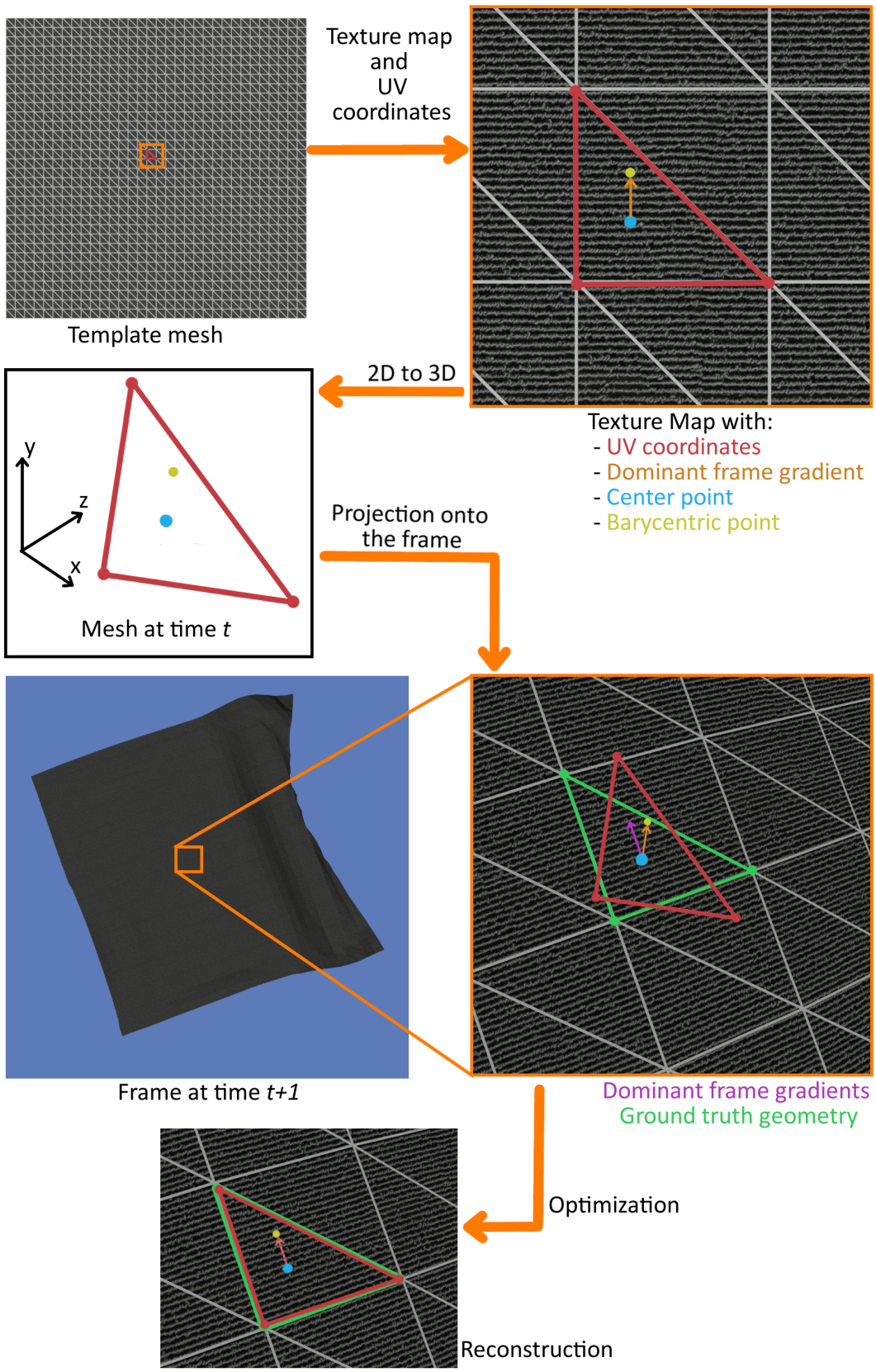

Figure 5.3: Overview of the texture term that illustrates the different steps explained in Section 5.3.

2D to 3D. Since the 2D point $\mathbf{b}_{\mathrm{TM}}$ can be represented as a linear combination, one can easily compute the corresponding 3D barycentric point $\mathbf{b}_{\mathrm{3D}} \in \mathbb{R}^3$ on the mesh $\mathbf{V}$ as

$$\mathbf{b}_{\mathrm{3D}} = \mathbf{B}_{i,1}\mathbf{V}_k + \mathbf{B}_{i,2}\mathbf{V}_m + \mathbf{B}_{i,3}\mathbf{V}_l \tag{5.7}$$

as well as the 3D center point

$$\mathbf{c}_{\mathrm{3D}} = \frac{1}{3}\mathbf{V}_k + \frac{1}{3}\mathbf{V}_m + \frac{1}{3}\mathbf{V}_l. \tag{5.8}$$

The barycentric coordinates remain constant, so that $\mathbf{b}_{\mathrm{3D}}$, $\mathbf{c}_{\mathrm{3D}}$ only depend on the mesh vertices. In consequence, they can be changed by deforming the surface.

Dominant mesh gradients. Given $\mathbf{b}_{\mathrm{3D}}, \mathbf{c}_{\mathrm{3D}}$ and the camera intrinsics, the 3D points can be projected into the current frame $t+1$ and one obtains $\mathbf{b}_{\mathrm{F}} = \pi(\mathbf{b}_{\mathrm{3D}}) \in \mathbb{R}^2$ and $\mathbf{c}_{\mathrm{F}} = \pi(\mathbf{c}_{\mathrm{3D}}) \in \mathbb{R}^2$. The difference $\mathbf{d}_{\mathrm{M}} = \mathbf{b}_{\mathrm{F}} - \mathbf{c}_{\mathrm{F}} \in \mathbb{R}^2$ is the projected texture gradient of the mesh, which is called the *dominant mesh gradient*. At the same time, one can retrieve the dominant frame gradient $\mathbf{d}_{\mathrm{F}}$ (marked as violet arrow in Figure 5.3) of the frame $t+1$ at the location of $\mathbf{c}_{\mathrm{F}}$ by the image look up

$$\mathbf{d}_{\mathrm{F}} = \begin{pmatrix} I^{t+1}_{\mathrm{Dir},u}(\lceil \mathbf{c}_{\mathrm{F}} \rceil) \\ I^{t+1}_{\mathrm{Dir},v}(\lceil \mathbf{c}_{\mathrm{F}} \rceil) \end{pmatrix}. \tag{5.9}$$

Similar to the photometric alignment, RONDA penalizes deviations of the two gradients $\mathbf{d}_{\mathrm{M}}, \mathbf{d}_{\mathrm{F}}$ since they should align with each other, if the deformation is correctly estimated. Next, the concrete energy function for the above described method will be shown.

5.4 Texture-based Constraint

$e_{\mathrm{Texture}}(\mathbf{V}) : \mathbb{R}^{N\times 3} \to \mathbb{R}$ ensures that the dominant mesh gradient and the dominant frame gradient at the center location of the triangle $\mathbf{F}_i = (k, m, l)$ are aligned and it is defined as

$$e_{\mathrm{Texture}}(\mathbf{V}) = \sum_{i=0}^{F} \left\| \rho_p \left(\mathbf{d}_{\mathrm{M},i}, \mathbf{d}_{\mathrm{F},i} \right) \right\|^2. \tag{5.10}$$

Here,

$$\mathbf{d}_{\mathrm{M},i} = \frac{\pi\left(b\left(\mathbf{F}_i\right)\right) - \pi\left(c\left(\mathbf{F}_i\right)\right)}{\left\| \pi\left(b\left(\mathbf{F}_i\right)\right) - \pi\left(c\left(\mathbf{F}_i\right)\right) \right\|} \tag{5.11}$$

represents the dominant mesh gradient of face $\mathbf{F}_i$ where the division is applied to normalize the vector and

$$\mathbf{d}_{\mathrm{F},i} = \begin{pmatrix} I^{t+1}_{\mathrm{Dir},u}\left(\pi\left(c\left(\mathbf{F}_i\right)\right)\right) \\ I^{t+1}_{\mathrm{Dir},v}\left(\pi\left(c\left(\mathbf{F}_i\right)\right)\right) \end{pmatrix} \tag{5.12}$$

is the normalized dominant frame gradient at the location of the face's center where $I^{t+1}_{\mathrm{Dir},u}, I^{t+1}_{\mathrm{Dir},v}$ are the texture gradient images of frame $t+1$. The function $c(\mathbf{F}_i) : \{1, ..., N\}^3 \rightarrow \mathbb{R}^3$ computes the 3D center point of a triangle as

$$c(\mathbf{F}_i) = \frac{1}{3}\mathbf{V}_k + \frac{1}{3}\mathbf{V}_m + \frac{1}{3}\mathbf{V}_l \tag{5.13}$$

and $b(\mathbf{F}_i) : \{1, ..., N\}^3 \rightarrow \mathbb{R}^3$ calculates the 3D barycentric point

$$b(\mathbf{F}_i) = \mathbf{B}_{i,1}\mathbf{V}_k + \mathbf{B}_{i,2}\mathbf{V}_m + \mathbf{B}_{i,3}\mathbf{V}_l. \tag{5.14}$$

Finally, $\rho_p(\mathbf{x}, \mathbf{y}) : \{\mathbf{x}, \mathbf{y} \in \mathbb{R}^2 | \, \|\mathbf{x}\| \in \{0, 1\} \wedge \|\mathbf{y}\| \in \{0, 1\}\} \rightarrow \mathbb{R}^2$ is defined as

$$\rho_p(\mathbf{x}, \mathbf{y}) = \begin{cases} \min(\mathbf{x} - \mathbf{y}, \mathbf{x} + \mathbf{y}) & \text{, if } \|\min(\mathbf{x} - \mathbf{y}, \mathbf{x} + \mathbf{y})\| < p \\ & \quad \wedge \mathbf{x} \neq \mathbf{0} \\ & \quad \wedge \mathbf{y} \neq \mathbf{0} \\ 0 & \text{, else.} \end{cases} \tag{5.15}$$

It computes the minimum of the differences between $\mathbf{x}, \mathbf{y}$ and $\mathbf{x}, -\mathbf{y}$. The reason for this is as already mentioned above, that there are two dominant frame gradients for the case of line patterns. Since it is assumed that the initialization is close to the ground truth, RONDA takes the minimum of the two possible directions. But the function only returns this for the case that they are similar up to a threshold p, because completely wrong directions are often induced by occlusions and noise and therefore the corresponding cost should not be counted. Beside this condition, $\mathbf{x}$ has to be a non-zero vector. Otherwise, it would mean that the dominant mesh gradient was undefined because there were no line patterns present at the corresponding part of the surface. Similarly, $\mathbf{y}$ has to be non-zero because the opposite would imply that there is no dominant frame gradient present at the specified location in the current frame.

5.5 Effects of the Texture Term

One could also ask if it is not sufficient to use a penalty term based on first order similarity metrics like the image gradient instead of the more complex HOG descriptor. The reason why Histogram of Oriented Gradients outperforms the others for the case of fabrics is that HOG is more robust with respect to small noisy gradient directions, which are often present in pictures of clothes. Another advantage is that if the pattern is interrupted for example by hard shadows, the proposed approach

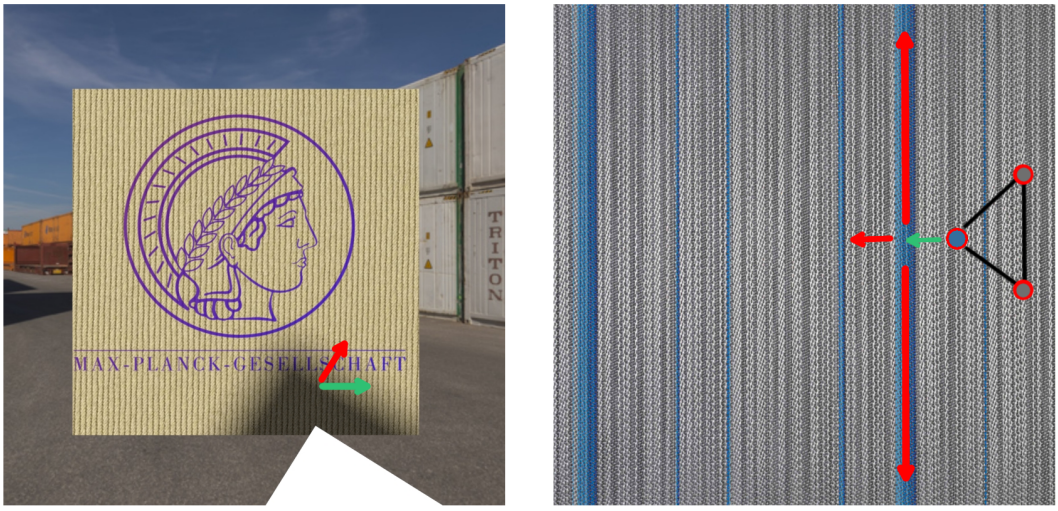

Figure 5.4: Effects of the texture term. Left. Red arrow shows the ordinary texture gradient that is clearly influenced by the shadow. The green arrow shows the dominant frame gradient of the HOG descriptor. It is not affected by the shadow since there are enough small gradients corresponding to the texture pattern. Right. One can see a triangle where the blue vertex projects wrongly. The green arrow represents the correct deformation of the vertex. The red arrows mark further moving directions, which would also result in a zero penalty of the photometric term.

is still able to correctly recover the dominant angles of the lines since the frequency of gradient directions corresponding to the structure of the fabrics are often higher than the ones caused by the shadow boundary. These findings are illustrated in Figure 5.4 left.

Apart from that, the photometric term cannot well handle line-like structures in clothes. The high frequent patterns imply that there are a lot of locations having the same color so that it is difficult to decide which one is the right projection area. Assuming a vertex v has the same color as a line structure in the fabrics, then $e_{\text{Photo}}(\mathbf{V})$ would return zero wherever v lies along the line and this is obviously wrong. Additionally, due to the density of the patterns it can also happen that the vertex skips one structure if the size of the update step is too large such that it goes to the neighbouring same-colored line, which is also wrong (see Figure 5.4 right). But if the photometric cost and the texture term are combined, one obtains an additional constraint such that it reduces the number of possible deformations.

Chapter 6

Optimization

6.1 Energy Function

After the different cost terms were introduced, one obtains the final energy function

$$\begin{aligned} e\left(\mathbf{V}, \mathbf{\Phi}\right) = & \lambda_{\text{Photo}} e_{\text{Photo}}\left(\mathbf{V}\right) \\ & + \lambda_{\text{Laplacian}} e_{\text{Laplacian}}\left(\mathbf{V}\right) \\ & + \lambda_{\text{Edge}} e_{\text{Edge}}\left(\mathbf{V}\right) \\ & + \lambda_{\text{Velocity}} e_{\text{Velocitiy}}\left(\mathbf{V}\right) \\ & + \lambda_{\text{Acceleration}} e_{\text{Acceleration}}\left(\mathbf{V}\right) \\ & + \lambda_{\text{Arap}} e_{\text{Arap}}\left(\mathbf{V}, \mathbf{\Phi}\right) \\ & + \lambda_{\text{Texture}} e_{\text{Texture}}\left(\mathbf{V}\right), \end{aligned} \tag{6.1}$$

where the lambdas are the weights, which are set before the optimization starts and afterwards they are kept constant. In the beginning their values can be freely chosen such that they force a high or a small influence of the corresponding cost term regarding the solution of the minimization. Setting the weights to zero means that the respective part of the energy function is not used at all.

6.2 Solver Architecture

$e\left(\mathbf{V}, \mathbf{\Phi}\right)$ is sequentially optimized for each of the T frames, which results in the deformed meshes $\mathbf{V}^t$ and the rotation parameters $\mathbf{\Phi}^t$ where $t \in \{1, ..., T\}$. The initial values of $\mathbf{V}$ at each time step are set to the result of the optimization for the previous frame and the rotation angle values are zero at the beginning. For the case of $t = 2$, the starting guess of $\mathbf{V}$ is equal to the template mesh $\hat{\mathbf{V}}$.

The architecture of the solver that is used to minimize $e(\mathbf{V}, \mathbf{\Phi})$, is based on the work of Zollhöfer et al. [1]. In particular, the presented energy function forms a non-linear least squares optimization problem and each of the constraints can be expressed in terms of residuals. For example $e_{\text{Velocity}}(\mathbf{V})$ can be written as

$$e_{\text{Velocity}}(\mathbf{V}) = \sum_{i=0}^{N} \left\| \mathbf{V}_i - \mathbf{V}_i^t \right\|^2 \tag{6.2}$$

$$= \sum_{i=0}^{N} \left(\mathbf{V}_{i,x} - \mathbf{V}_{i,x}^t \right)^2 + \left(\mathbf{V}_{i,y} - \mathbf{V}_{i,y}^t \right)^2 + \left(\mathbf{V}_{i,z} - \mathbf{V}_{i,z}^t \right)^2 \tag{6.3}$$

$$= \sum_{i=0}^{N} \left(r_{\text{Velocity},i,x}(\mathbf{V}) \right)^2 + \left(r_{\text{Velocity},i,y}(\mathbf{V}) \right)^2 + \left(r_{\text{Velocity},i,z}(\mathbf{V}) \right)^2, \tag{6.4}$$

where

$$r_{\text{Velocity},i,d}(\mathbf{V}) = \mathbf{V}_{i,d} - \mathbf{V}_{i,d}^t \tag{6.5}$$

for $d \in \{x, y, z\}$. Given this notation, one already saw in Chapter 3 that minimizing $e(\mathbf{V}, \mathbf{\Phi})$ boils down to solving the LSE

$$2\mathbf{J}_r^\top \mathbf{J}_r \Delta\mathbf{V} = -2\mathbf{J}_r^\top r\left(\mathbf{V}^k, \mathbf{\Phi}^k\right), \tag{6.6}$$

where $r\left(\mathbf{V}^k, \mathbf{\Phi}^k\right) : \mathbb{R}^{N\times 3} \times \mathbb{R}^{N\times 3} \to \mathbb{R}^M$ is the function that takes the current mesh and rotation estimate $\mathbf{V}^k$, $\mathbf{\Phi}^k$ with N vertices and returns the vector that contains all M residuals of the constraints after the kth solver iteration. $\mathbf{J}_r$ is the Jacobian of $r\left(\mathbf{V}^k, \mathbf{\Phi}^k\right)$. In the Appendix A, one can see how the entries of the matrices $\mathbf{J}_r^\top \mathbf{J}_r$ and $\mathbf{J}_r$ can be derived. The variables $\Delta\mathbf{V}$ of the LSE are the update vector, which refines $\mathbf{V}^k$.

The minimization problem is then solved in two nested loops. The inner one computes the solution $\Delta\mathbf{V}$ of the linear system of equations 6.6 by multiple iterations of the Conjugate Gradient method. Afterwards, the outer loop performs a Gauss-Newton step using the intermediate result $\Delta\mathbf{V}$.

If one looks closely at the structure of the energy functions, one can see that the texture term has two times F residuals. The photo, velocity and acceleration constraint have three times N residuals and finally the Laplacian, ARAP and the edge terms have residuals that also depend on the neighbourhood structure. Assuming that each vertex has six neighbours one gets N times 18 residuals for the two former costs and N times six for the latter one. Therefore, $r\left(\mathbf{V}^k, \mathbf{\Phi}^k\right)$ has the size $M = 2F + 3N + 3N + 3N + 18N + 18N + 6N$. Since the meshes often have more than 5,000 vertices and 7,000 faces (that corresponds to 30,000 variables and

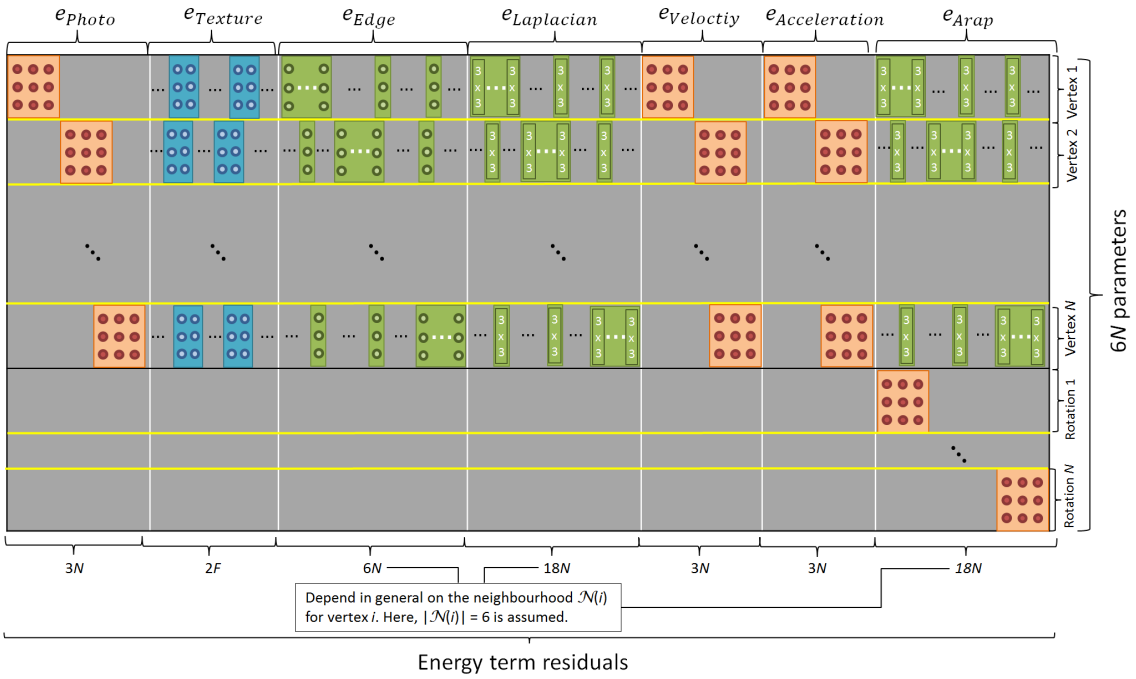

Figure 6.1: The transposed Jacobian of the energy function. Note that e_{Photo}, e_{Veloctiy} and $e_{\text{Acceleration}}$ only have non-zero 3×3 entries along their diagonal. For $e_{\text{Laplacian}}$ and e_{Arap} each row i has one bigger block of non-zero entries of size $3 \times 3|\mathcal{N}(i)|$, which corresponds to the neighbourhood of the vertex $\mathbf{V}_i$ and 3×3 non-zero blocks that appear if $\mathbf{V}_i$ is in the neighbourhood of another vertex. Additionally, e_{Arap} has tridiagonal entries for the rotation parameters. e_{Edge} has a similar structure as $e_{\text{Laplacian}}$ but the bigger blocks have a size of $3 \times |\mathcal{N}(i)|$ and the smaller ones are just 3×1 columns. Note that for $e_{\text{Laplacian}}$, e_{Edge} and e_{Arap} the number of columns of all smaller blocks together and the one of the bigger block at the same row are equal due to the bidirectional structure of the neighbourhood. e_{Texture} has single non-zero 3×2 blocks for row i if $\mathbf{V}_i$ is part of the face belonging to this column. So one can see that the matrix is sparse in most parts.

269,000 residuals), the Jacobian has dimensions of 269,000×30,000, which would require a lot of memory. Luckily, the matrices have a sparse structure, as shown in Figure 6.1 and therefore, they do not have to be stored entirely. Instead, the entries are computed on demand.

The solver is implemented on the GPU using Nvidia's Compute Unified Device Architecture (CUDA) [71] that strongly speeds up the computation due to the parallelism of the kernel units on the graphics card. One can exploit the fact that each row of the above LSE can be calculated concurrently at different kernels. Therefore, one core is assigned to one mesh vertex $\mathbf{V}_i$ and it computes the rows $3i-2, 3i-1, 3i$ of Equation 6.6 as well as the rows corresponding to its rotation

parameters. More precisely, for the left hand side of Equation 6.6 the $3 \times M$ matrix, which is formed by the rows $3i-2, 3i-1, 3i$ of $\mathbf{J}_r^\top$ is multiplied with the $M \times 3$ matrix defined by the columns $3i-2, 3i-1, 3i$ of $\mathbf{J}_r$. For the right side of Equation 6.6, the same $3 \times M$ matrix as above is multiplied with the residual vector. The computation of the rotation parameters follows the same strategy.

Chapter 7

Results

All the results were performed on a XMG P505 notebook, which has 16GB RAM, a NVIDIA GeForce GTX 970M with 3GB RAM and a Intel Core i7-4720 with 2.60GHz. Furthermore, the frame directions were computed in a preprocessing stage in order to safe test time. In the next step, synthetic scenes will be evaluated. After that, the reconstructions of real recordings will be shown, followed by an ablation analysis, where the influence of the texture term is demonstrated. At last, possible applications will be presented. The test data, the thesis' results and a demo video are available on the project's webpage[1]. The estimated deformations can be best examined by watching the video.

7.1 Synthetic Scenes

The scenes are modelled, textured and animated in Blender [10]. The mesh of the first frame serves as the geometry of the template. The UV coordinates and the texture map are initially known, since they were already created during the modelling stage. To get a better visualization of the estimated deformation and to be able to compare it with the input image sequence, the reconstructed motions are rendered in the same scene as the original geometry in the input video.

Carpet. The carpet sequence in Figure 7.1 shows a rotating object. Due to the regularization terms of the energy function, the reconstruction is close to the ground truth shape although the object's texture is challenging, because of the fact that it is similar to the background color and that it has large flat regions, where the photometric term is less informative.

[1] `http://www.ronda.3dmodellierung.com/`

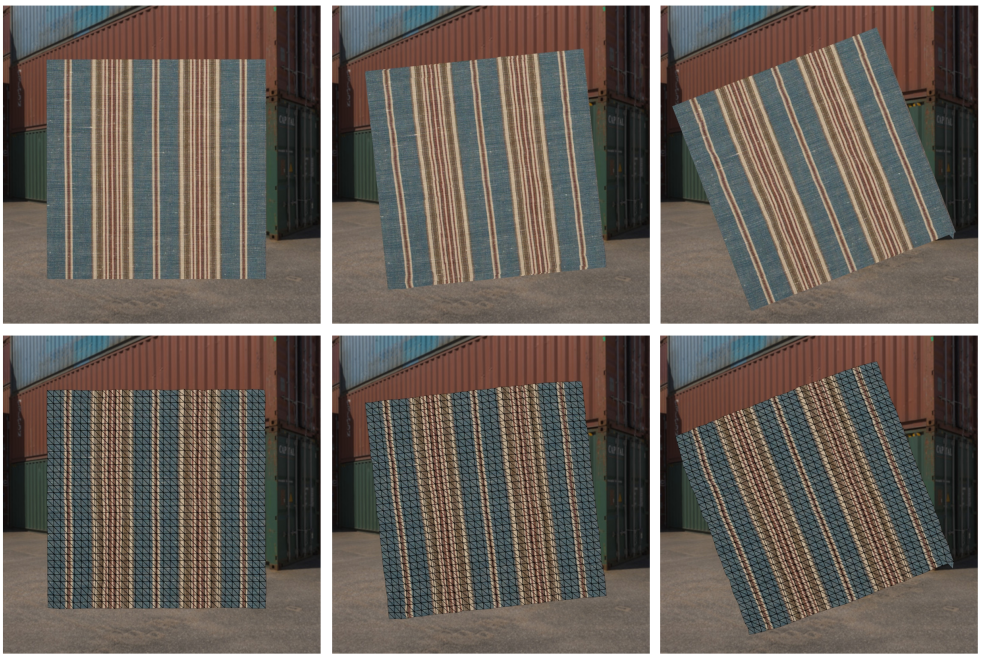

Figure 7.1: Reconstruction of the synthetic carpet sequence. The top row shows different frames of the video. The one below illustrates the rendered reconstruction using the same background as the original video.

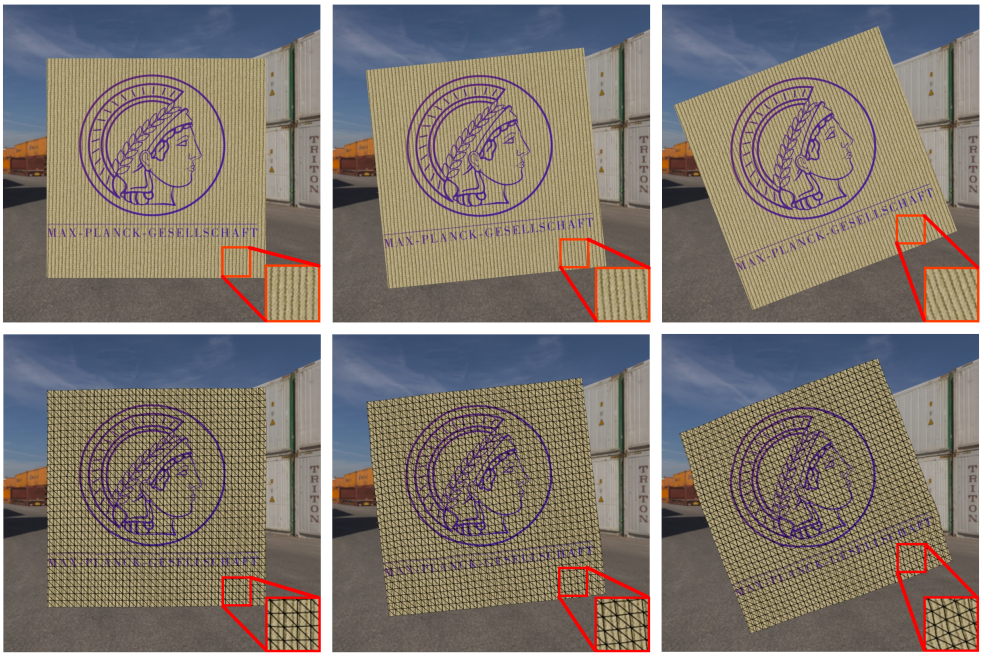

Figure 7.2: Reconstruction of the synthetic MPI logo sequence. Top row. Different frames of the input video. Bottom row. The corresponding reconstructions, that are rendered in the same environment as the original sequence.

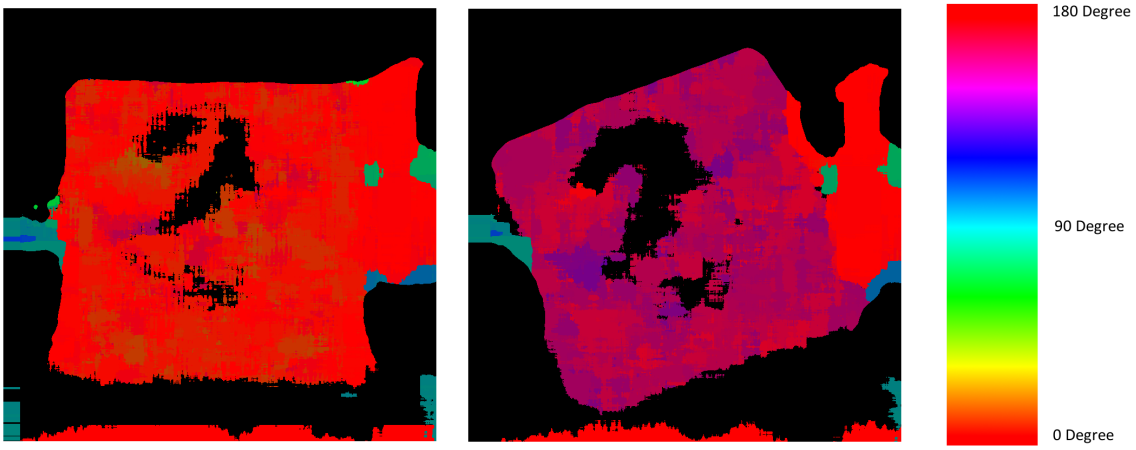

Figure 7.3: Angles of the MPI logo sequence. The color encoding represents the dominant frame angle for each pixel. The images show the angles of the left and right frame in Figure 7.2. It becomes obvious that they are robustly estimated in the region of the carpet and lie around zero degree for the left frame and around 150 degree for the other frame. But the proposed texture descriptor is also able to identify flat regions that are marked in black, where no dominant frame angle can be found.

MPI logo. The MPI logo sequence in Figure 7.2 shows the same rotation as in the scene before, but with a different background and another object texture. Again, RONDA is able to reconstruct the rigid deformation. The correct rotation is caused by the combination of the texture term and the photometric penalty. On the one hand, the latter cost provides only a small amount of information near the object boundaries, since these parts of the mesh have a dense pattern with very similar colors. On the other hand, the texture term is more informative exactly at these locations, since the line-like structure is not interrupted by the logo. In consequence, the combination of both leads to the best result. Figure 7.3 illustrates the estimated dominant gradient directions of the frames corresponding to the left and right column in Figure 7.2. One can see that they are correctly and robustly detected in the object region. But the flat image areas, represented as black pixels, are identified too.

Synthetic lines. The last synthetic test setting is a high resolution video of a carpet that has the characteristic line-like pattern. The sequence in Figure 7.4 shows a non-rigid deformation and the corresponding reconstructions for different frames. It can be seen that the proposed approach is able to track the deformation and due to the specialized texture term, it can also recover regions where the photometric term is less informative. The right column shows a comparison with respect to the ground truth geometry that validates the accuracy of the reconstruction.

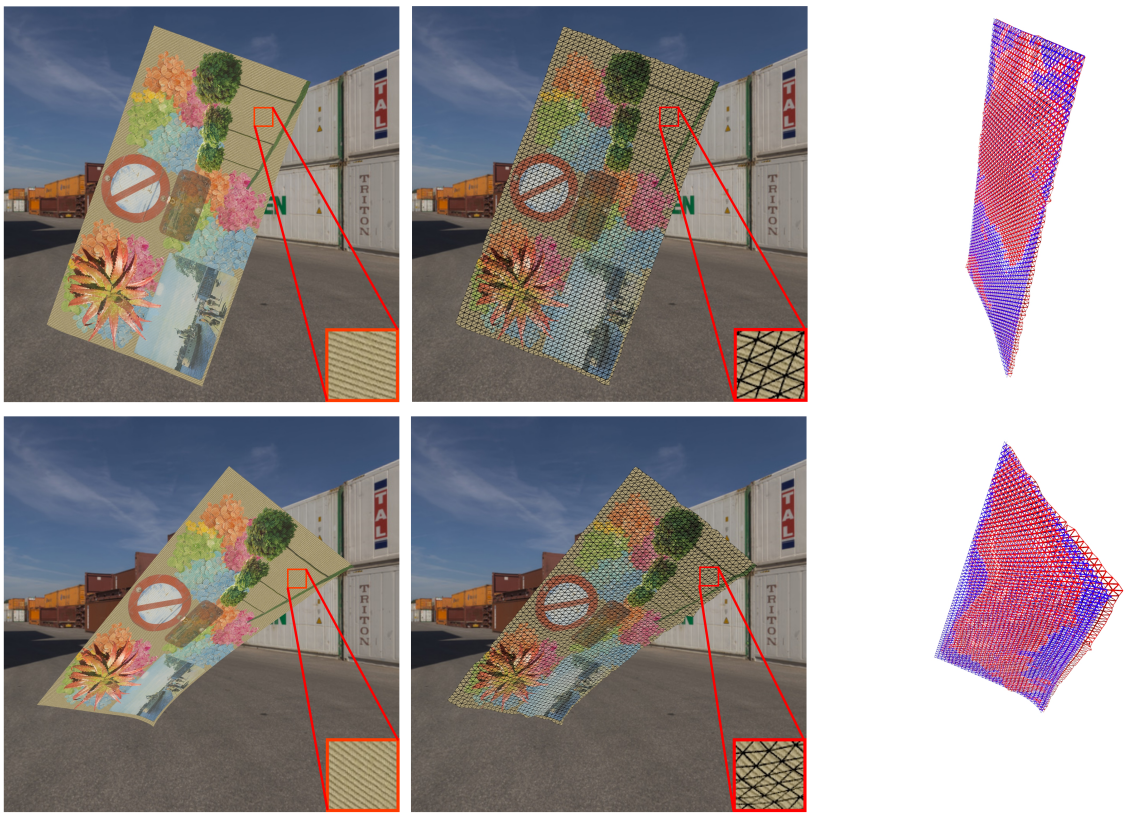

Figure 7.4: Reconstruction of the synthetic line sequence. Top row from left to right. One can see an input frame of the synthetic line sequence, followed by the estimated deformed geometry rendered in the same environment as the original video. On the right, the ground truth deformation (blue) and the corresponding reconstruction (red) for the same frame is shown. The bottom row illustrates the result for another frame of the sequence. It applies to both rows that the reconstruction is close to the ground truth.

7.2 Real Scenes

Next, real recordings of different types of objects will be evaluated. In the beginning, new sequences are tested, followed by videos from existing approaches.

7.2.1 New Recordings

To test the thesis' method and especially the proposed texture term, new videos were recorded since existing data did not capture line-like cloth patterns. The scenes were

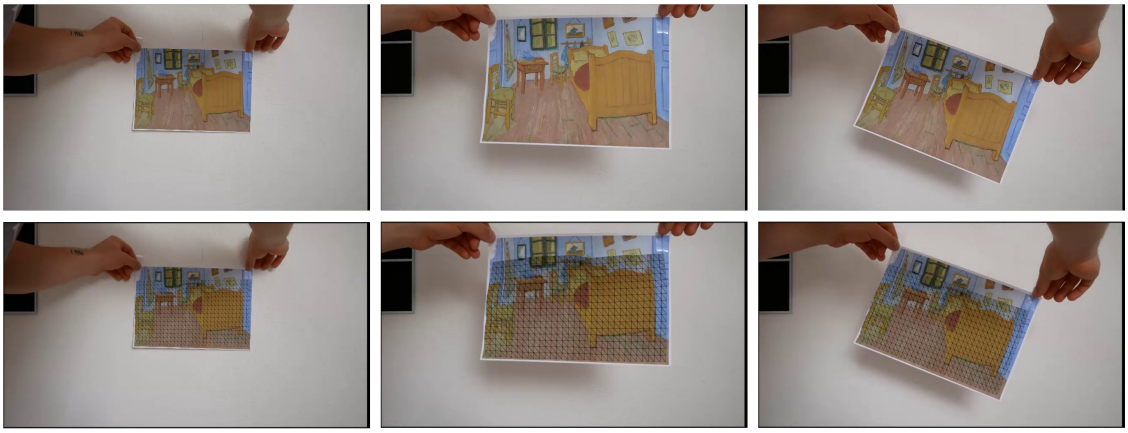

Figure 7.5: Reconstruction of the Van Gogh sequence. Top row. Input frames. Bottom row. Corresponding reconstructions projected in the input frame. Note that the proposed approach can handle fast translations as well as rotations.

filmed with a Panasonic GX80[2] and a lens[3], which has a focal length of 12mm. The sensor size is 17.3×13mm with an pixel aspect ratio of $4 : 3$. The template meshes are modelled by hand in Blender. The first frame serves as the texture map and the UV coordinates are the projected locations of the vertices in the image plane. It is assumed that all vertices of the mesh are visible in the first frame to avoid wrong vertex colors caused by occlusions.

Van Gogh. The Van Gogh sequence in Figure 7.5 visualizes a piece of paper with a printed Van Gogh painting. The video shows fast translations and rotations of the object. RONDA is able to reproduce them well since the smoothing level of the Gaussian can be adjusted such that the photometric term is able to find meaningful gradients for large vertex displacements, too.

Rigid kids painting. Figure 7.6 shows the rigid kids painting sequence, where similar transformations are performed as in the scene before.

Non-rigid kids painting. This video shows non-rigid transformations that are more challenging than rigid ones. Figure 7.7 shows three input frames, the reconstructed geometry reprojected into the frame and the same meshes from a different viewing position. One can see that the proposed approach is able to reconstruct the convexity of the deformed object.

[2] `http://www.panasonic.com/de/consumer/foto-video/lumix-g-wechselobjektivkameras/dmc-gx80.html`

[3] `http://www.panasonic.com/de/consumer/foto-video/lumix-g-objektive/h-fs12032e.html`

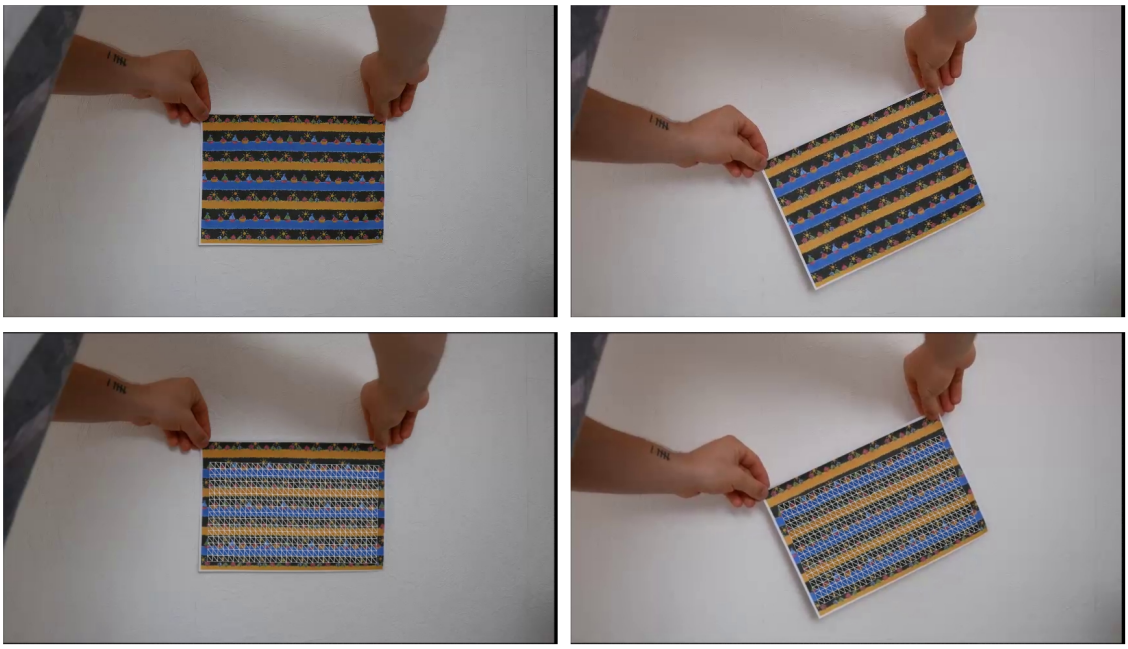

Figure 7.6: Reconstruction of the rigid kids painting sequence. Top row. Input frames. Bottom row. Corresponding reconstructions projected in the input frame. Although the texture color is constant in large regions, RONDA can reproduce the underlying shape due to the smoothness constraints.

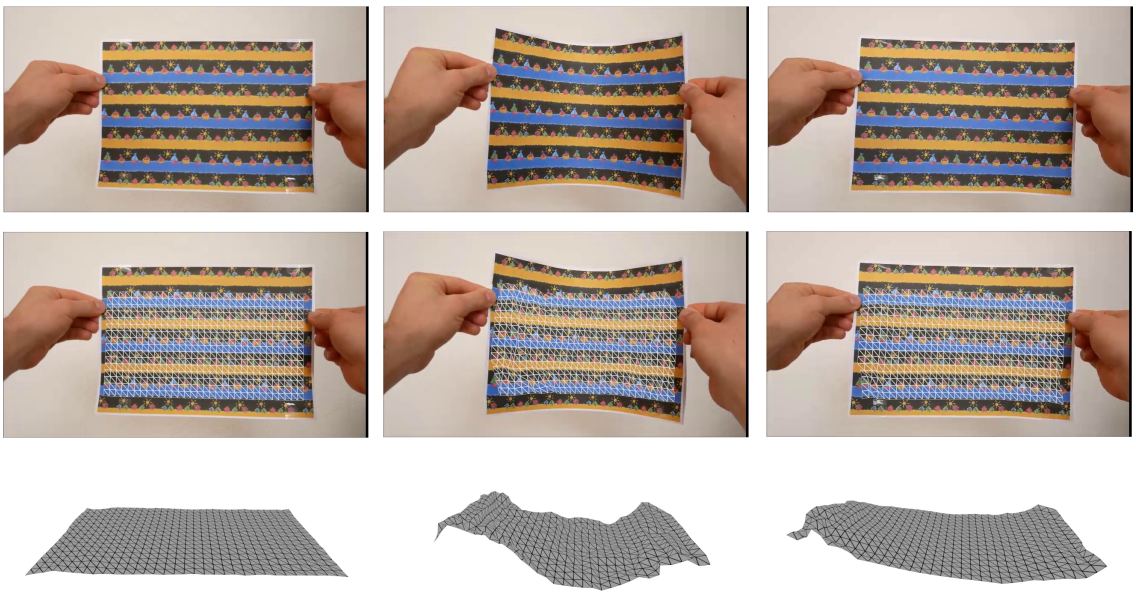

Figure 7.7: Reconstruction of the non-rigid kids painting sequence. Top row. Input frames. Middle row. Corresponding reconstructions projected in the input frame. The last row shows the 3D reconstruction from a different view.

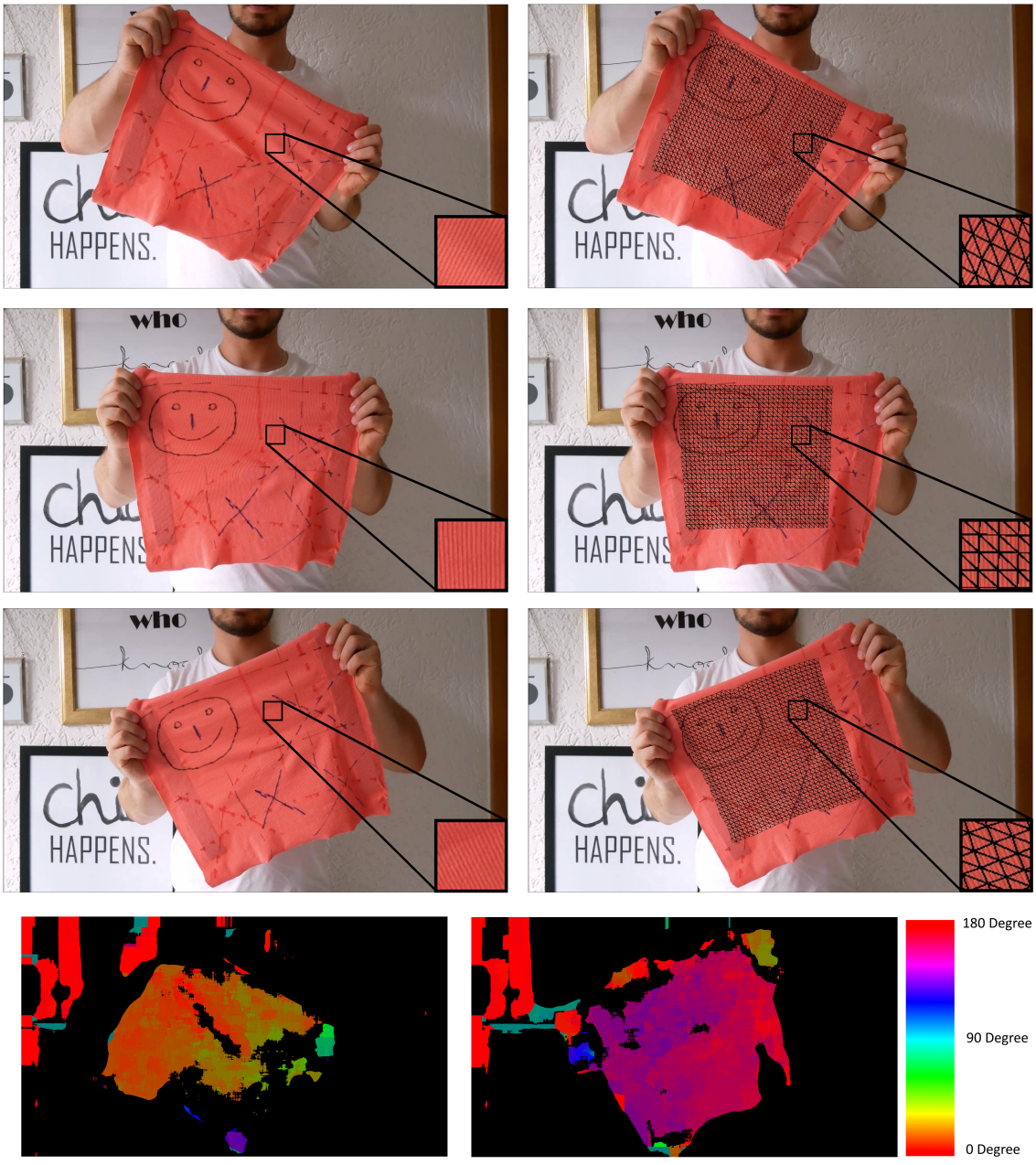

Figure 7.8: Reconstruction of the fabric pattern sequence. The first three rows show pairs of input frame and reconstructed geometry, that is projected into the video frame. The last row illustrates the estimated dominant angles corresponding to the first and last frame above.

Fabric pattern. To evaluate the texture term, a moving piece of cloth that has the typical line patterns as described in the previous chapters, is recorded and reconstructed. Figure 7.8 shows different frames of the sequence and the corresponding reconstructions. Although the scene has a challenging shading and a sparse object texture, RONDA is able to recover the deformations due to the texture term, which accurately tracks the dominant directions of the line pattern.

7.2.2 Comparison to Other Approaches

In the following, RONDA is evaluated on test data from related approaches. All of the examples below provide the geometry of the template mesh that corresponds to the first frame of the sequence. Partially, the vertex colors are given too. If they are not available, the vertices are projected into the first frame as it was the case for the own recordings.

Yu et al. [5]. The thesis tested the face sequence of Yu et al. [5], where different non-rigid deformations are performed due to varying face expressions. Figure 7.9 shows the result of four different frames. One can see that also the untextured reconstructions illustrate the underlying expression. Figure 7.10 shows a direct comparison between RONDA's reconstruction and the one of Yu of the last frame shown in Figure 7.9. It becomes obvious that both capture the facial expression, but the proposed approach is faster than the one of Yu due to the computational power of the GPU. This example with 500 frames indicates that the proposed approach also performs well for longer video sequences.

Varol et al. [6]. The challenging part of this sequence was that the vertex displacement at each time step was larger than for other testing scenes due to fast motions. Some parts of the video are even cut out that is another source of the large displacements. By increasing the Gaussian kernel size, the proposed approach was able to recover the fast motion and produced results which are comparable to existing methods. Some example frames can be found in Figure 7.11.

Salzmann et al. [8]. The proposed approach was also tested against a sequence of Salzmann et al. [8]. Although the background color is similar to the one of the moving object and there is the fact that shadows are present, RONDA was able to reproduce the motion, which visually agrees with the one in the video (see Figure 7.12).

Valgaerts et al. [7]. Next, the face sequence of Valgaerts et al. [7], who proposed a method based on blend shapes and a 3D scan, is evaluated. Figure 7.13 shows the results. Again, facial expressions are reconstructed well.

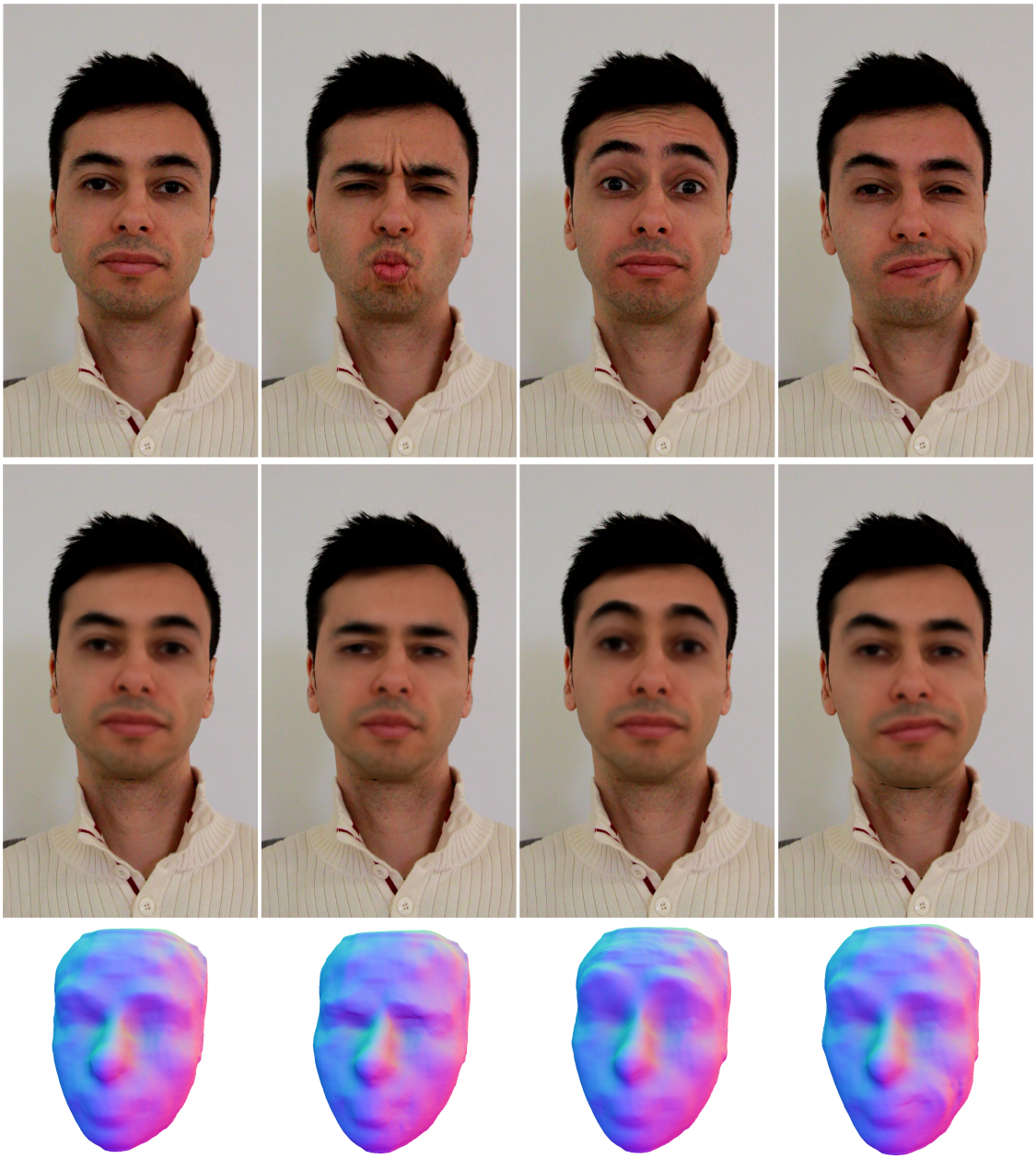

Figure 7.9: Reconstruction of Yu's sequence [5]. Top row. Input frames. Middle row. Corresponding reconstructions projected in the input frame. At the bottom row the reconstructed deformed geometries are visible from a different view. Note that the facial expressions of the person are clearly recognizable. Especially the eyebrow regions move remarkably. But also the mouth deformation for the last frame can be seen.

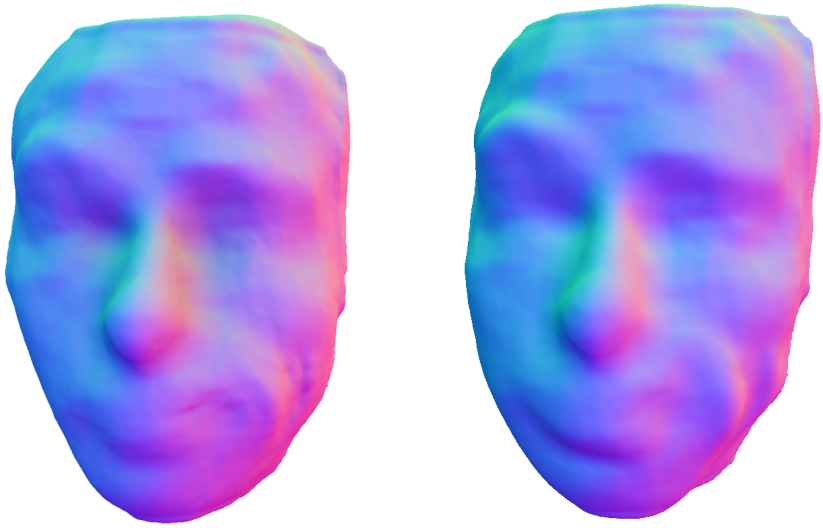

Figure 7.10: Comparison of RONDA's reconstruction (left) and the one of Yu [5] (right). Note that the wrinkles and the facial expression is more visible in the thesis' estimated geometry.

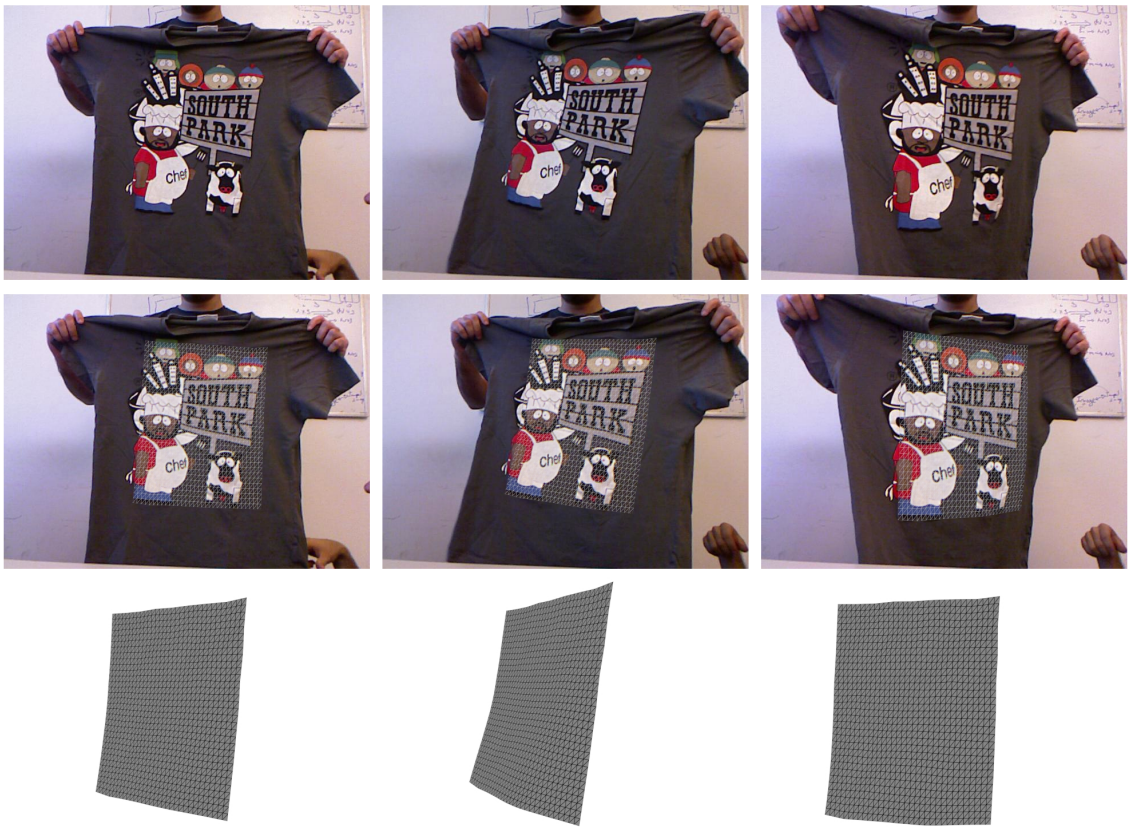

Figure 7.11: Reconstruction of Varol's sequence [6]. Top row. Input frames. Middle row. Corresponding reconstructions projected in the input frame. Bottom row. Estimated deformed geometries from another view.

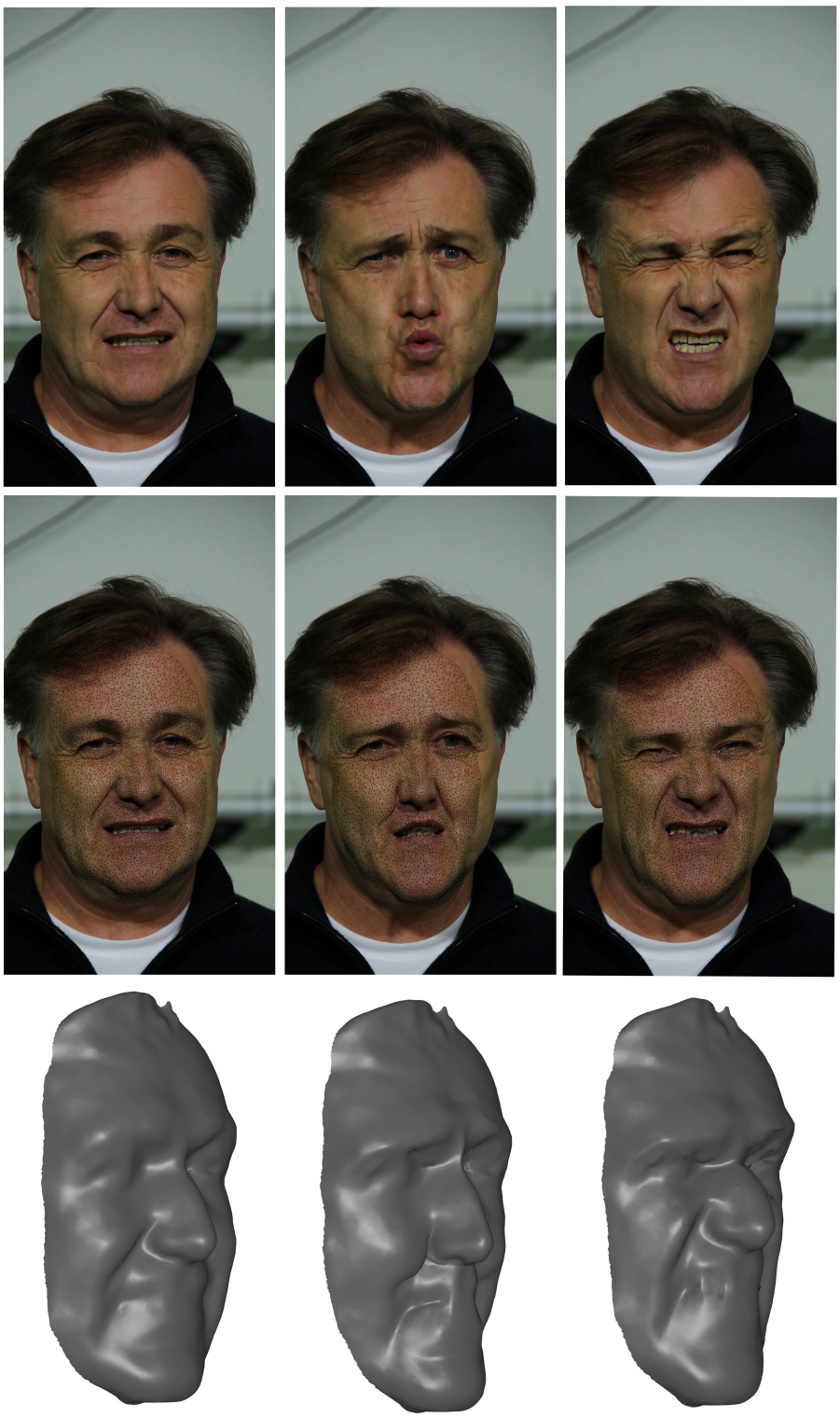

Figure 7.13: Reconstruction of Valgaerts' sequence [7]. Top row. Input frames. Middle row. Corresponding reconstructions projected in the input frame. Bottom row. Estimated deformed geometries from another view. Again, one can see that RONDA is able to reconstruct facial expressions.

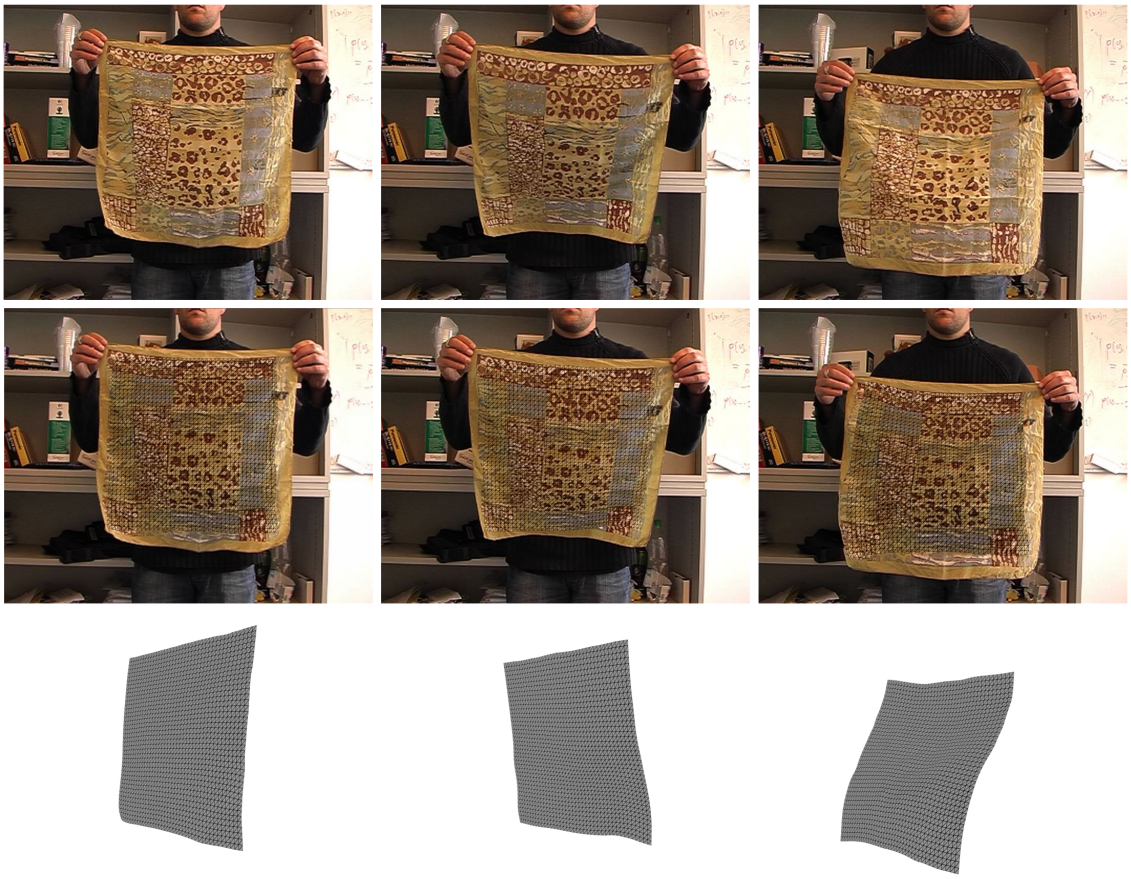

Figure 7.12: Reconstruction of Salzmann's sequence [8]. Top row. Input frames. Middle row. Corresponding reconstructions projected in the input frame. Bottom row. Estimated deformed geometries from another view.

7.3 Ablation Analysis

To validate that the texture term e_{Texture} can help to refine the solution, the thesis tested several of the above scenes with and without it. In case of synthetic recordings, the ground truth data was available such that a quantitative evaluation is possible. Therefore, the measured error is defined as the sum of the Euclidean distances between the ground truth mesh vertices and the corresponding reconstructed points, averaged over the number of frames and the vertex count.

Table 7.1 shows the errors of the MPI logo sequence visualized in Figure 7.2 and the evaluation of a modified version of the scene, but again with the same motion (see Figure 7.14).

Method	Reconstruction error with logo	Reconstruction error of the modified scene
Texture-only	5.627	4.084
Photometric-only	1.321	6.676
Combined	0.825	6.217

Table 7.1: Quantitative comparison of the reconstructions of the MPI logo sequence (see Figure 7.2) and the modified rotation sequence (see Figure 7.14) depending on the weights λ_{Photo} and λ_{Texture}.

Regarding the first videos, the error decreases by 77%, if one uses only the photometric term instead of the texture cost. The best result is obtained by the combination of e_{Texture} and e_{Photo} (decrease of 38% compared to the photometric-only estimate). The reason for this behaviour is that on the one hand, the mesh has certain parts where the line patterns dominate and therefore, the texture term gives a better estimate in these regions. On the other hand, some areas of the surface contain more color variations such that the photometric term is more informative at these locations. If the two penalties are combined, they benefit from each other such that together they deliver the best result for this test setting.

The errors of the modified video are different from the ones above. In this case, the best result was obtained by the texture-only setting. But this makes sense too, since the whole scene does not have a lot of color variations, leading to an error increase by the photometric term. Due to the fact that the line patterns are not interrupted by the logo, the texture term can robustly compute the line directions and they deliver a reliable source of information. The heat maps in Figure 7.14 showing the error per vertex, as well as the ground truth confirm these findings.

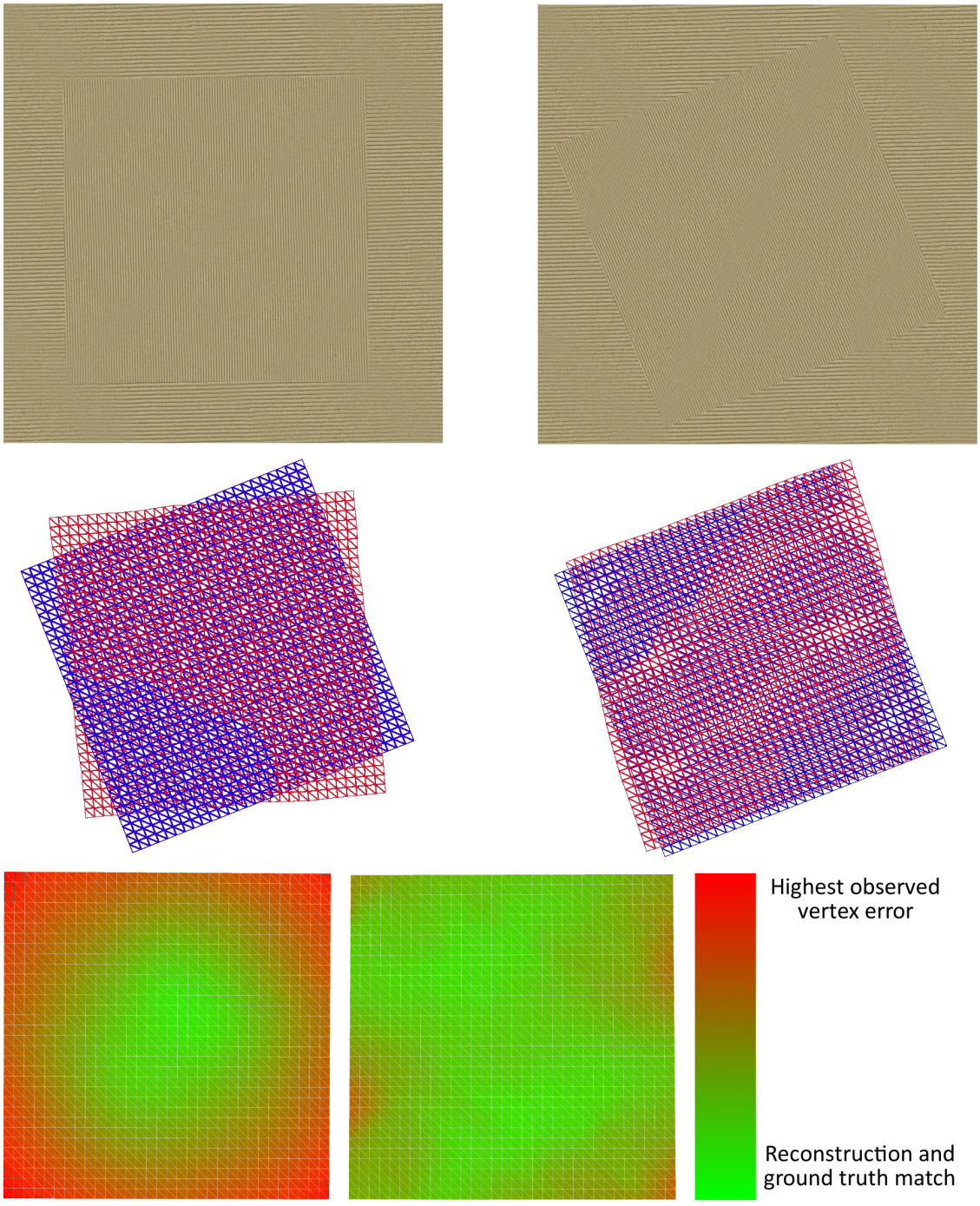

Figure 7.14: Top row. The first and last frame of the modified rotation sequence. Note that the background and foreground colors are very similar so that an observer can hardly distinguish between object and background. Middle row. On the left side, the photometric-only reconstruction is shown in red together with the ground truth in blue. On the right side, the texture-only estimate is visualized. Bottom row from left to right. The per vertex error visualized as a heat map for the photometric-only reconstruction (left) and the one that only uses the texture term (right).

Here one can see that the photometric-only reconstruction is not able to rotate the object completely. Instead, the texture-only estimate is close to the ground truth and captures the motion. The heat maps reflect these observations. The one for the photometric-only reconstruction has overall higher errors that occur mostly at the boundary of the object because the outer vertices perform a larger movement and therefore the error increases faster, if the rotation is not recovered. The heat map for the texture-only estimation has globally smaller vertex errors. The highest ones occur also at the boundary region, but for a different reason. They are projected into a frame area, where the HOG descriptor was not able to find unique dominant gradient directions since the background stripes and the ones of the object have a different orientation. By the definition of the texture term, these areas are regarded as flat regions and e_{Texture} is set to zero. In consequence, these vertices are only moved by the regularizers and this leads to the small observed errors near the boundary.

The synthetic line sequence in Figure 7.4 is also quantitatively evaluated and the errors are shown in Table 7.2. Again, the texture term refines the estimated result.

Method	Reconstruction error of the synthetic stripes sequence
Photometric-only	26.83
Combined	25.54

Table 7.2: Quantitative comparison of the reconstructions of the synthetic stripes sequence (see Figure 7.4) depending on the weights of the photometric and the texture term.

All in all, the texture term can help to recover the deformations for challenging scenes like the modified version of the MPI logo sequence, where the whole video has almost the same color. But also for rather usual setups like the synthetic line sequence, the texture term can improve the estimate, if it is combined with e_{Photo}, since there are also small patches that contain line-like structures.

7.4 Performance

To evaluate the speed of the computation, the MPI Logo sequence from Section 7.1 was used, which consists of 100 frames. If not stated otherwise, the frame size is 800×800 and the mesh has 1,089 vertices. The solver performs 20 Gauss-Newton steps and ten CG iterations.

First, the relation between the vertex count and the performance was tested. Therefore, the number of mesh points is quadrupled at each step. Table 7.3 below shows the counts and the corresponding time that was needed to compute the solution. One can see that for a large number of vertices the performance scales linearly with respect to their count that enables the user to reconstruct also fine details in an adequate amount of time. 50,000 vertices for example are already sufficient to represent small wrinkles in faces without perceivable distortions. Table 7.4 shows how well RONDA can handle large video resolutions, since it can still compute one frame in about 1.62 seconds. That implies that the proposed approach can also cope with the resolutions of current recording devices. The tables (Table 7.5 and Table 7.6) below show the speed of the thesis' optimization for different numbers of CG iterations and Gauss-Newton steps. One can see that the method needs at most 0.53 seconds per frame which is still fast. In most cases, the algorithm converges after 20 Gauss-Newton steps and 40 iterations of the Conjugate Gradient method.

All in all, RONDA needs between 0.13 and 1.62 seconds per frame, which is fast regarding the size of the non-linear system that is solved. As stated before, the performance test was made on a laptop. The speed can still be improved by running the algorithm on a desktop PC.

Number of vertices N	Time (in seconds)
1,089	19
4,225	22
16,641	49
66,049	176

Table 7.3: Performance for different numbers of mesh vertices.

Resolution of the video	Time (in seconds)
400×400	13
800×800	19
1600×1600	44
3200×3200	162

Table 7.4: Performance for different video resolutions.

Number of CG iterations	Time (in seconds)
10	19
20	23
40	33
60	43
80	53

Table 7.5: Performance for different numbers of CG iterations.

Number of Gauss-Newton iterations	Time (in seconds)
10	16
20	19
40	25
60	31
80	37

Table 7.6: Performance for different numbers of Gauss-Newton iterations.

7.5 Applications

The 3D animation data, that is the output of RONDA, can be used for several applications. Two specific ones are shown in Figure 7.15, where in the left column, the original sequence is shown and in the right column, one can see the edited virtual scene containing the deformed model.

Re-texturing the mesh. Since the thesis' approach returns a deforming geometry with UV coordinates, one can edit the corresponding texture map such that one is able to produce realistic deformations with arbitrary textures that can be used as training data in Neural Networks. Figure 7.15 illustrates this field of application.

Re-lighting the scene. Usually, re-lighting a scene without knowing the shape of a moving object is difficult and shadows cannot be determined well. But since RONDA estimates the deforming geometry, one can change the scene lighting for the foreground such that the shading remains realistic. An example is shown at the bottom row of Figure 7.15.

Figure 7.15: Applications. Top row. Example of re-texturing the mesh. One can see that the interior of the signpost contains flowers instead of the logo "SOUTH PARK" after the re-texturing step. Bottom row. This example shows that one can also re-texture (stripes on the face skin) and re-light the scene such that the shadows on the deformed model look realistic. For this particular scene a blue and orange light source were chosen.

Chapter 8

Discussion

Of course, RONDA has some drawbacks and limitations since the problem of reconstructing non-rigid deformations from a monocular video is in general hard and under-constrained. The remaining challenges will be the content of the following paragraphs together with ideas for future work to tackle them.

Initialization. Although Agisoft delivers an accurate and detailed template mesh, the global position still has to be initialized manually. To remove this needed user input, one can try to automatically estimate the six parameters responsible for translation and rotation in 3D. Therefore, one can build an energy-based approach by using a modified version of the thesis' photometric alignment term that takes as parameters a matrix which has six degrees of freedom such that it rigidly transforms all mesh vertices to ensure a correct projection into the first frame.

Real-time performance. Although RONDA is a fast framework, it is still not in real-time. Especially the computation of the dominant gradient directions in the frame takes a lot of time. Therefore, future work can involve to build an efficient GPU-based version of Histogram of Oriented Gradients to speed up the process.

Weight selection. Another practical issue is to find the right choices for λ_i where $i \in \{1, ..., 7\}$. Since the thesis has seven energy terms, one has to determine six weights (one of the seven can be kept constant). This process can be automatized by implementing a meta-optimization. It takes as input a synthetic scene such that the ground truth is known and then a gradient-free method, for example the Downhill-Simplex algorithm [72], can be used to determine the best weights. Although they are tuned to the specific scenario, one still obtains a better estimate for certain kinds of deformations such that these weights can also be applied to real recordings where ground truth data is not available.

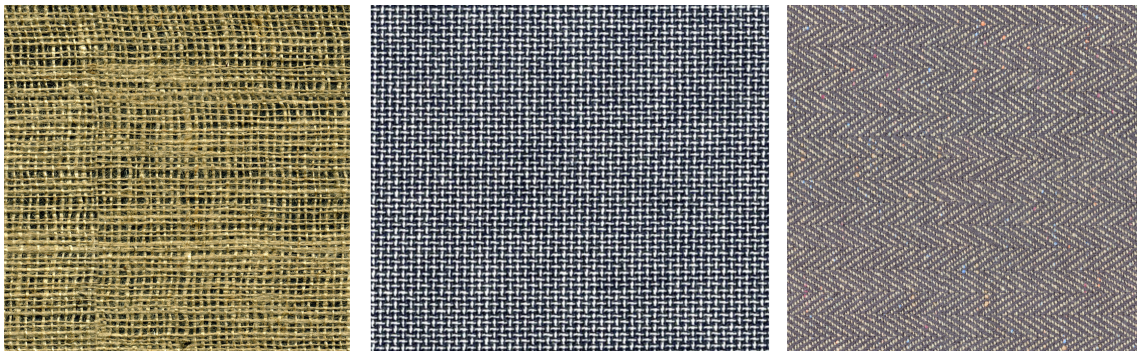

Figure 8.1: Other fabric patterns. Note that they have more than two dominant gradient directions. Graphics were retrieved from [74].

Generalization of the texture pattern. It is clear that other structural cloth patterns exist and that they can have more than two dominant gradient directions. Some examples are shown in Figure 8.1. Therefore, the presented texture term could be extended in a future work to be able to handle these cases. One idea is to apply the Hough Transform [73] to the frame, which results in a better representation of line directions or just to look for more than two peaks in the histogram and according to that to penalize more than one difference of gradient directions.

L_1 **norm.** The L_2 norm for the Laplacian surface constraint introduces a smoothness that is sometimes not desired. If one thinks of sharp folds like the one in Figure 8.2, it is better to use the L_1 norm. The reason is that the residuals are linearly penalized such that the solution introduces a certain sparsity with respect to them. This means most residuals will be zero except a few which is the desired behaviour for sharp deformations. For the ARAP term it can also be beneficial to have a L_1 norm to handle scenarios where only a part of the object changes its location while the rest does not move.

Parametric surface models. One can see that the number of variables linearly scales with the amount of mesh vertices. For small geometries (up to 15,000 vertices) this is not problematic, but if one wants to reconstruct larger objects, it is of advantage to have a parametric surface model such that one does not have this linear dependency.

Coarse-to-fine strategy. To be able to reconstruct larger motions between consecutive frames too and to reduce the number of solver iterations until it converges, one can use a coarse-to-fine strategy. This can be applied to the image such that the process iteratively reduces the size of the smoothing kernel. But also on the mesh level it makes sense to start with a coarser representation and then refine it by inserting additional vertices. Figure 8.3 illustrates the idea.

Figure 8.2: Sharp folds. Left. Input frame. Right. Reconstruction of Salzmann and Fua [56]. Graphics were retrieved from [56].

Shading term. One can see in the results that RONDA is able to reconstruct non-textured regions, but only if features in the surrounding area are given. To make the estimation in such flat parts more robust, one can use an additional shading term. Due to the steadily improving intrinsic image decomposition methods like the one of Meka et al. [75] and the fact that clothes often underlie a Lambertian material model [76], the shading estimation is possible without loosing too much generality. Liu-Yin et al. [77] even showed that specular reflections based on an approximation of the Phong model [78] can be used to reconstruct non-rigid deformations.

Contour-based term. To be more robust with respect to convex-concave ambiguities, one can also try to take the object contour into account. Since there is a huge progress in the field of image segmentation, like the work of Khoreva et al. [79], one can try to segment the objects contour from the background and then enforce that the projected mesh boundary and the segmented one in the frame match.

Combination with a CNN. In the introduction it was said that the thesis can be used to produce data for deep learning methods. But one could also think of the opposite direction. The sensor technologies evolve steadily such that one can use the advanced measurements. For example, there already exist strips that measure the bending and send this information to the computer. If they are combined in a way that a surface is formed, one could infer the position of points on the surface by taking the bending measure into account. If these strips can be developed further such that they are as small as sewing threads, one could weave them into fabrics and directly measure the realistic deformation to obtain annotated 3D motion sequences. By re-texturing them one could also generate additional data. Given this, it can be used to train a CNN which outputs the kind of deformation shown in a frame. This estimate can be integrated as an additional constraint for the proposed energy.

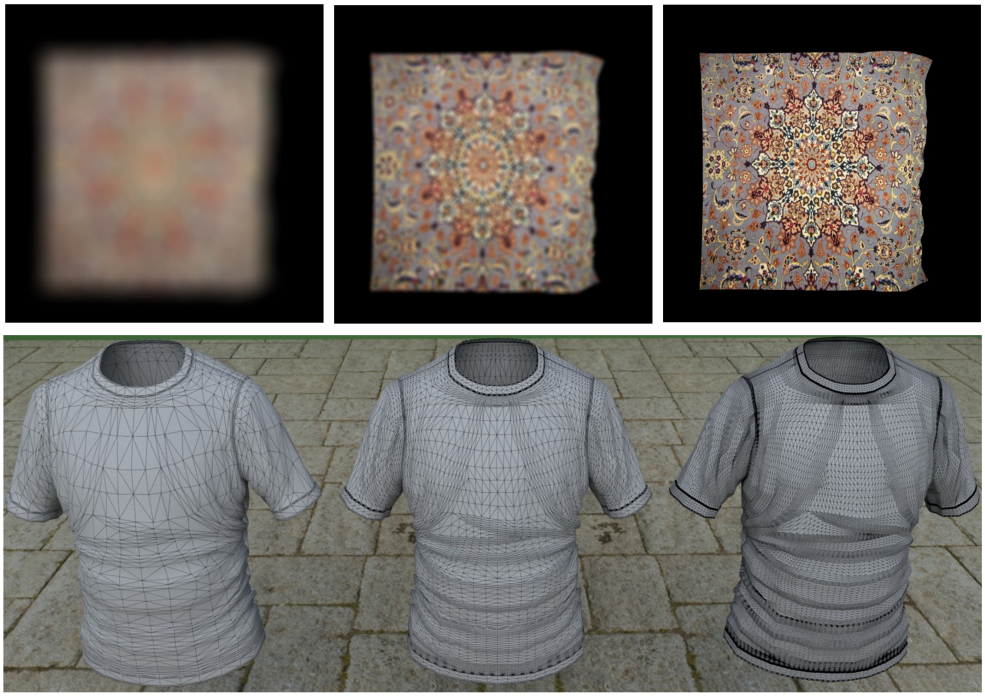

Figure 8.3: Coarse-to-fine strategy. The first row shows the decreasing of the kernel size that results in different smoothing levels. Below one can see the coarsest mesh on the left side, which is then refined from left to right by inserting additional vertices.

Camera motions. At the moment, it is assumed that the camera does not move and all deformations in the video are caused by the mesh itself. To extend the framework to also handle camera motions, like the characteristic shaking of hand-held recordings, one can try to jointly estimate the mesh movement and the one of the recording device. Therefore, one can take the background information in the video into account to discriminate the two types of motion as proposed by Davison et al. [80].

Non-sequential optimization. Instead of optimizing each frame by itself, one can also try to take n frames simultaneously and solve for the corresponding n deformations. This has the advantage, that imperfections like noise and occlusions in a single frame can be neutralized by the other frames and one can impose a more accurate temporal smoothness prior. For example the deformations can be expressed in the trajectory space like Akhter et al. [81] proposed. They use the Discrete Cosine Transform to get basis trajectories such that all motions are linear combinations of them. One the one hand, this reduces the number of variables and on the other hand it also allows to directly control the temporal smoothness by setting the coefficients corresponding to higher frequencies to zero.

Chapter 9

Conclusion

The thesis presented an energy-based method, which is able to solve the challenging task of estimating non-rigid motion, given a single RGB video and a template of the object. The proposed cost function is divided into different constraints where each ensures plausible physical properties, for example photometric alignment. Reformulating the task mathematically leads to a non-linear least squares optimization problem that is tackled by iteratively solving a linear system of equations and subsequent update steps.

One part of the constraints is tuned to the specific case of woven fabrics. It exploits the regular line patterns due to the fabrication process. To precisely detect them, a modified version of Histogram of Oriented Gradients was developed and the difference between the projected texture gradient directions and the ones in the frame are penalized.

The GPU-based optimization framework is fast since it can solve the linear system in parallel. At the same time, the memory usage is low due to the fact that the Jacobian is not fully stored, but instead the computation is performed on demand.

The presented results show that the method delivers accurate reconstructions, which can be used in further applications. One possible practical scenario could be to re-texture the object or to insert the moving mesh into a different environment.

Clearly, RONDA has some limitations caused by the nature of the problem. To overcome these drawbacks, there is a variety of possible directions for future work such that the topic of non-rigid motion estimation from monocular video will also be interesting in the years to come.

Appendix A

Gradients of the Energy Terms

A.1 Photometric Alignment e_{Photo}

It was already mentioned that the entries of the transposed Jacobian have a tridiagonal structure. For a vertex $\mathbf{V}_i$ the rows correspond to the partial derivatives with respect to the three coordinates $\mathbf{V}_{t,x}$, $\mathbf{V}_{t,y}$ and $\mathbf{V}_{t,z}$. The columns correspond to the colors $c \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$. Since the partial derivatives are similar for all color channels the thesis only provides the ones for the red channel's residual

$$r_{\text{Photo,R},i}(\mathbf{V}) = \sigma_p \left(C_{\mathrm{R}} \left(\pi \left(\mathbf{V}_i \right) \right) - \mathbf{c}_{\mathrm{R}_i} \right). \tag{A.1}$$

In the following only the case where $\sigma_p(x) = x$ is considered because otherwise the partial derivatives are zero. This yields

$$r_{\text{Photo,R},i}(\mathbf{V}) = C_{\mathrm{R}} \left(\pi \left(\mathbf{V}_i \right) \right) - \mathbf{c}_{\mathrm{R}_i} \tag{A.2}$$

and the partial derivatives can be derived as

$$\frac{\partial r_{\text{Photo,R},i}(\mathbf{V})}{\partial \mathbf{V}_{t,x}} = \frac{\partial C_{\mathrm{R}} \left(\pi \left(\mathbf{V}_i \right) \right)}{\partial u} \frac{\alpha_u}{\mathbf{V}_{i,z}} \tag{A.3}$$

$$\frac{\partial r_{\text{Photo,R},i}(\mathbf{V})}{\partial \mathbf{V}_{t,y}} = \frac{\partial C_{\mathrm{R}} \left(\pi \left(\mathbf{V}_i \right) \right)}{\partial u} \frac{\gamma}{\mathbf{V}_{i,z}} + \frac{\partial C_{\mathrm{R}} \left(\pi \left(\mathbf{V}_i \right) \right)}{\partial v} \frac{\alpha_v}{\mathbf{V}_{i,z}} \tag{A.4}$$

$$\frac{\partial r_{\text{Photo,R},i}(\mathbf{V})}{\partial \mathbf{V}_{t,z}} = \frac{\partial C_{\mathrm{R}} \left(\pi \left(\mathbf{V}_i \right) \right)}{\partial u} \frac{-\alpha_u \mathbf{V}_{i,x} - \gamma \mathbf{V}_{i,y}}{\left(\mathbf{V}_{i,z} \right)^2} + \frac{\partial C_{\mathrm{R}} \left(\pi \left(\mathbf{V}_i \right) \right)}{\partial v} \frac{-\alpha_v \mathbf{V}_{i,y}}{\left(\mathbf{V}_{i,z} \right)^2}, \tag{A.5}$$

if $t = i$. Otherwise the partial derivatives are zero. $\frac{\partial C_{\mathrm{R}}(\pi(\mathbf{V}_i))}{\partial u}$ and $\frac{\partial C_{\mathrm{R}}(\pi(\mathbf{V}_i))}{\partial v}$ are approximated with finite differences in the smoothed image $I_{\mathrm{R}}^{t+1} * \mathbf{G}_w$ at the location $\lceil \pi \left(\mathbf{V}_i \right) \rceil$.

A.2 Laplacian Surface Deformation $e_{\text{Laplacian}}$

The Laplacian surface deformation constraint can be reformulated as

$$e_{\text{Laplacian}}(\mathbf{V}) = \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \left\| (\mathbf{V}_i - \mathbf{V}_j) - (\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j) \right\|^2 \tag{A.6}$$

$$= \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \sum_{k \in \{x,y,z\}} \left((\mathbf{V}_{i,k} - \mathbf{V}_{j,k}) - (\hat{\mathbf{V}}_{i,k} - \hat{\mathbf{V}}_{j,k}) \right)^2 \tag{A.7}$$

$$= \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \sum_{k \in \{x,y,z\}} \left(r_{\text{Laplacian},i,j,k}(\mathbf{V}) \right)^2, \tag{A.8}$$

where

$$r_{\text{Laplacian},i,j,k}(\mathbf{V}) = (\mathbf{V}_{i,k} - \mathbf{V}_{j,k}) - (\hat{\mathbf{V}}_{i,k} - \hat{\mathbf{V}}_{j,k}). \tag{A.9}$$

The partial derivative with respect to $\mathbf{V}_{t,k}$ is then

$$\frac{\partial r_{\text{Laplacian},i,j,k}(\mathbf{V})}{\partial \mathbf{V}_{t,k}} = \begin{cases} 1 & \text{, if } t = i \\ -1 & \text{, if } t = j \\ 0 & \text{, else.} \end{cases} \tag{A.10}$$

A.3 Preservation of Edge Lengths e_{Edge}

The edge length constraint can be expressed in terms of the residuals as

$$e_{\text{Edge}}(\mathbf{V}) = \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \left(\|\mathbf{V}_i - \mathbf{V}_j\| - \left\| \hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j \right\| \right)^2 \tag{A.11}$$

$$= \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \left(r_{\text{Edge},i,j}(\mathbf{V}) \right)^2, \tag{A.12}$$

where $r_{\text{Edge},i,j}(\mathbf{V})$ is defined as

$$r_{\text{Edge},i,j}(\mathbf{V}) = \|\mathbf{V}_i - \mathbf{V}_j\| - \left\| \hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j \right\| \tag{A.13}$$

$$= \sqrt{(\mathbf{V}_{i,x} - \mathbf{V}_{j,x})^2 + (\mathbf{V}_{i,y} - \mathbf{V}_{j,y})^2 + (\mathbf{V}_{i,z} - \mathbf{V}_{j,z})^2} - \sqrt{\left(\hat{\mathbf{V}}_{i,x} - \hat{\mathbf{V}}_{j,x} \right)^2 + \left(\hat{\mathbf{V}}_{i,y} - \hat{\mathbf{V}}_{j,y} \right)^2 + \left(\hat{\mathbf{V}}_{i,z} - \hat{\mathbf{V}}_{j,z} \right)^2}. \tag{A.14}$$

The partial derivative with respect to $\mathbf{V}_{t,k}$ are then

$$\frac{\partial r_{\text{Edge},i,j}(\mathbf{V})}{\partial \mathbf{V}_{t,k}} = \begin{cases} \frac{1}{\|\mathbf{V}_i - \mathbf{V}_j\|} \left(\mathbf{V}_{i,k} - \mathbf{V}_{j,k}\right) & \text{, if } t = i \\ \frac{1}{\|\mathbf{V}_i - \mathbf{V}_j\|} \left(\mathbf{V}_{i,k} - \mathbf{V}_{j,k}\right)(-1) & \text{, if } t = j \\ 0 & \text{, else,} \end{cases} \tag{A.15}$$

where $k \in \{x, y, z\}$.

A.4 Smooth Motion e_{Velocity}

The smooth motion constraint can be written as

$$e_{\text{Velocti}y}\left(\mathbf{V}\right) = \sum_{i=0}^{N} \left\| \mathbf{V}_i - \mathbf{V}^t{}_i \right\|^2 \tag{A.16}$$

$$= \sum_{i=0}^{N} \sum_{k \in \{x,y,z\}} \left(\mathbf{V}_{i,k} - \mathbf{V}^t{}_{i,k}\right)^2 \tag{A.17}$$

$$= \sum_{i=0}^{N} \sum_{k \in \{x,y,z\}} \left(r_{\text{Velocti}y,i,k}\left(\mathbf{V}\right)\right)^2, \tag{A.18}$$

where the residual is defined as

$$r_{\text{Velocti}y,i,k}\left(\mathbf{V}\right) = \mathbf{V}_{i,k} - \mathbf{V}^t{}_{i,k}. \tag{A.19}$$

The partial derivatives with respect to $\mathbf{V}_{t,k}$ are

$$\frac{\partial r_{\text{Velocity},i,k}(\mathbf{V})}{\partial \mathbf{V}_{t,k}} = \begin{cases} 1 & \text{, if } t = i \\ 0 & \text{, else,} \end{cases} \tag{A.20}$$

where $k \in \{x, y, z\}$.

A.5 Smooth Direction of Motion $e_{\text{Acceleration}}$

The smooth motion direction over time constraint can be written as

$$e_{\text{Acceleration}}\left(\mathbf{V}\right) = \sum_{i=0}^{N} \left\| \left(\mathbf{V}_i - \mathbf{V}^t{}_i\right) - \left(\mathbf{V}^t{}_i - \mathbf{V}_i^{\text{t}-1}\right) \right\|^2 \tag{A.21}$$

$$= \sum_{i=0}^{N} \sum_{k \in \{x,y,z\}} \left(\left(\mathbf{V}_{i,k} - \mathbf{V}^t{}_{i,k}\right) - \left(\mathbf{V}^t{}_{i,k} - \mathbf{V}_{i,k}^{\text{t}-1}\right)\right)^2 \tag{A.22}$$

$$= \sum_{i=0}^{N} \sum_{k \in \{x,y,z\}} \left(r_{\text{Acceleration},i,k}\left(\mathbf{V}\right)\right)^2, \tag{A.23}$$

where the residual is defined as

$$r_{\text{Acceleration},i,k}\left(\mathbf{V}\right) = \left(\mathbf{V}_{i,k} - \mathbf{V}^{t}{}_{i,k}\right) - \left(\mathbf{V}^{t}{}_{i,k} - \mathbf{V}^{\text{t}-1}_{i,k}\right). \tag{A.24}$$

The partial derivatives with respect to $\mathbf{V}_{t,k}$ are

$$\frac{\partial r_{\text{Acceleration},i,k}(\mathbf{V})}{\partial \mathbf{V}_{t,k}} = \begin{cases} 1 & \text{, if } t = i \\ 0 & \text{, else,} \end{cases} \tag{A.25}$$

where $k \in \{x, y, z\}$.

A.6 As-rigid-as-possible Constraint e_{Arap}

The ARAP constraint can be reformulated as

$$e_{\text{Arap}}(\mathbf{V}, \mathbf{\Phi}) = \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \left\| (\mathbf{V}_i - \mathbf{V}_j) - \mathcal{R}_i(\mathbf{\Phi})(\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j) \right\|^2 \tag{A.26}$$

$$= \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \sum_{k \in \{x,y,z\}} \left((\mathbf{V}_{i,k} - \mathbf{V}_{j,k}) - \mathcal{R}_{i,k}(\mathbf{\Phi})(\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j) \right)^2 \tag{A.27}$$

$$= \sum_{i=0}^{N} \sum_{j \in \mathcal{N}(i)} \sum_{k \in \{x,y,z\}} \left(r_{\text{Arap},i,j,k}(\mathbf{V}, \mathbf{\Phi}) \right)^2, \tag{A.28}$$

where

$$r_{\text{Arap},i,j,k}(\mathbf{V}, \mathbf{\Phi}) = (\mathbf{V}_{i,k} - \mathbf{V}_{j,k}) - \mathcal{R}_{i,k}(\mathbf{\Phi})(\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j) \tag{A.29}$$

and $\mathcal{R}_{i,k}(\mathbf{\Phi})$ is the kth row of $\mathcal{R}_i(\mathbf{\Phi})$ where $k = x = 1$, $k = y = 2$ and $k = z = 3$. The partial derivatives with respect to $\mathbf{V}$ and $\mathbf{\Phi}$ are given as

$$\frac{\partial r_{\text{Arap},i,j,k}(\mathbf{V})}{\partial \mathbf{V}_{t,k'}} = \begin{cases} 1 - \frac{\partial}{\partial \mathbf{V}_{t,k'}} \left(\mathcal{R}_{i,k}(\mathbf{\Phi})(\hat{\mathbf{V}}_i) \right) & \text{, if } t = i \wedge k = k' \\ -\frac{\partial}{\partial \mathbf{V}_{t,k'}} \left(\mathcal{R}_{i,k}(\mathbf{\Phi})(\hat{\mathbf{V}}_i) \right) & \text{, if } t = i \wedge k \neq k' \\ -1 + \frac{\partial}{\partial \mathbf{V}_{t,k'}} \left(\mathcal{R}_{i,k}(\mathbf{\Phi})(\hat{\mathbf{V}}_j) \right) & \text{, if } t = j \wedge k = k' \\ \frac{\partial}{\partial \mathbf{V}_{t,k'}} \left(\mathcal{R}_{i,k}(\mathbf{\Phi})(\hat{\mathbf{V}}_j) \right) & \text{, if } t = j \wedge k \neq k' \\ 0 & \text{, else} \end{cases} \tag{A.30}$$

and

$$\frac{\partial r_{\text{Arap},i,j,k}(\mathbf{V})}{\partial \mathbf{\Phi}_{t,k',o}} = \begin{cases} -\frac{\partial}{\partial \mathbf{\Phi}_{t,k',o}} \left(\mathcal{R}_{i,k}(\mathbf{\Phi})(\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j) \right) & \text{, if } t = i \wedge k = k' \\ 0 & \text{, else,} \end{cases} \tag{A.31}$$

where $o \in \{1, 2, 3\}$ and $k' \in \{1, 2, 3\} \vee \{x, y, z\}$.

A.7 Texture-based Constraint e_{Texture}

For now it is assumed that the function $\rho_p(\mathbf{x}, \mathbf{y})$ is reduced to the case $\rho_p(\mathbf{x}, \mathbf{y}) = \mathbf{x} - \mathbf{y}$ because otherwise it returns zero which implies that the partial derivatives are zero. The case $\rho_p(\mathbf{x}, \mathbf{y}) = \mathbf{x} + \mathbf{y}$ can be derived by a simple sign change and is therefore not derived explicitly. The texture energy function boils down to

$$e_{\text{Texture}}(\mathbf{V}) = \sum_{i=0}^{F} \left\| \frac{\pi(b(\mathbf{F}_i)) - \pi(c(\mathbf{F}_i))}{\|\pi(b(\mathbf{F}_i)) - \pi(c(\mathbf{F}_i))\|} - \begin{pmatrix} I^{t+1}_{\text{Dir},u}(\pi(c(\mathbf{F}_i))) \\ I^{t+1}_{\text{Dir},v}(\pi(c(\mathbf{F}_i))) \end{pmatrix} \right\|^2 \tag{A.32}$$

$$\begin{aligned} &= \sum_{i=0}^{F} \left(\frac{\pi_1(b(\mathbf{F}_i)) - \pi_1(c(\mathbf{F}_i))}{\|\pi(b(\mathbf{F}_i)) - \pi(c(\mathbf{F}_i))\|} - I^{t+1}_{\text{Dir},u}(\pi(c(\mathbf{F}_i))) \right)^2 \\ &\quad + \left(\frac{\pi_2(b(\mathbf{F}_i)) - \pi_2(c(\mathbf{F}_i))}{\|\pi(b(\mathbf{F}_i)) - \pi(c(\mathbf{F}_i))\|} - I^{t+1}_{\text{Dir},v}(\pi(c(\mathbf{F}_i))) \right)^2 \end{aligned} \tag{A.33}$$

$$= \sum_{i=0}^{F} \left(r_{\text{Texture},i,1}(\mathbf{V})\right)^2 + \left(r_{\text{Texture},i,2}(\mathbf{V})\right)^2, \tag{A.34}$$

where the residuals are defined as

$$r_{\text{Texture},i,1}(\mathbf{V}) = \frac{\pi_1(b(\mathbf{F}_i)) - \pi_1(c(\mathbf{F}_i))}{\|\pi(b(\mathbf{F}_i)) - \pi(c(\mathbf{F}_i))\|} - I^{t+1}_{\text{Dir},u}(\pi(c(\mathbf{F}_i))) \tag{A.35}$$

$$r_{\text{Texture},i,2}(\mathbf{V}) = \frac{\pi_2(b(\mathbf{F}_i)) - \pi_2(c(\mathbf{F}_i))}{\|\pi(b(\mathbf{F}_i)) - \pi(c(\mathbf{F}_i))\|} - I^{t+1}_{\text{Dir},v}(\pi(c(\mathbf{F}_i))) \tag{A.36}$$

and $\mathbf{F}_i = (l, m, n)$. The partial derivatives with respect to $\mathbf{V}_{t,x}$, $\mathbf{V}_{t,y}$ and $\mathbf{V}_{t,z}$ are

$$\begin{aligned} &\frac{\partial r_{\text{Texture},i,r}(\mathbf{V})}{\partial \mathbf{V}_{t,k}} \\ &= \left(\frac{\partial}{\partial \mathbf{V}_{t,k}} \left(\frac{1}{\|\pi(b(\mathbf{F}_i)) - \pi(c(\mathbf{F}_i))\|} \right) \right) (\pi_r(b(\mathbf{F}_i)) - \pi_r(c(\mathbf{F}_i))) \\ &\quad + \frac{1}{\|\pi(b(\mathbf{F}_i)) - \pi(c(\mathbf{F}_i))\|} \left(\frac{\partial}{\partial \mathbf{V}_{t,k}} (\pi_r(b(\mathbf{F}_i)) - \pi_r(c(\mathbf{F}_i))) \right) \\ &\quad - \frac{\partial}{\partial \mathbf{V}_{t,k}} \left(I^{t+1}_{\text{Dir},r}(\pi(c(\mathbf{F}_i))) \right), \end{aligned} \tag{A.37}$$

where $r = 1 = u \vee r = 2 = v$, $k \in \{x, y, z\}$ and $t \in \{1, ..., N\}$. But only $t \in \{l, m, n\}$ are of interest since other cases will result in a zero partial derivative.

Bibliography

[1] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rhemann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. Real-time Non-rigid Reconstruction using an RGB-D Camera. *Association for Computing Machinery Transactions on Graphics (TOG)*, 2014. Cited on viii, 9, 39, 49

[2] A. Ravishankar Rao. Computing Oriented Texture Fields. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 1991. Cited on viii, 11, 12, 40

[3] Jian Liang, Daniel DeMenthon, and David Doermann. Flattening Curved Documents in Images. *Computer Vision and Pattern Recognition*, 2005. Cited on viii, 11, 12, 40

[4] Pablo Garrido, Levi Valgaerts, Chenglei Wu, and Christian Theobalt. Reconstructing Detailed Dynamic Face Geometry from Monocular Video. *Association for Computing Machinery Transactions on Graphics (TOG)*, 2013. Cited on viii, 13, 14, 17

[5] Rui Yu, Chris Russell, Neill D. F. Campbell, and Lourdes Agapito. Direct, Dense, and Deformable: Template-Based Non-Rigid 3D Reconstruction from RGB Video. *International Conference on Computer Vision*, 2015. Cited on viii, ix, 16, 34, 39, 59, 60, 61

[6] Aydin Varol, Mathieu Salzmann, Pascal Fua, and Raquel Urtasun. A Constrained Latent Variable Model. *Conference on Computer Vision and Pattern Recognition*, 2012. Cited on ix, 16, 59, 61

[7] Levi Valgaerts, Chenglei Wu, Andrés Bruhn, Hans-Peter Seidel, and Christian Theobalt. Lightweight Binocular Facial Performance Capture under Uncontrolled Lighting. *Special Interest Group on Computer Graphics and Interactive Techniques Asia*, 2012. Cited on ix, 59, 62

[8] Mathieu Salzmann, Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Closed-Form Solution to Non-rigid 3D Surface Registration. *European Conference on Computer Vision*, 2008. Cited on ix, 14, 59, 63

[9] Vincent Frei. The Art of VFX. `http://www.artofvfx.com`. Accessed: 2017-04-02. Cited on 2

[10] Blender Foundation. Blender. `https://www.blender.org/`. Accessed: 2017-03-31. Cited on 1, 52

[11] Will Smith and Norman Chan. Adam Savage's: Tested. `http://www.tested.com/tech/robots/2606-understanding-the-uncanny-valley-robotic-motion-breeds-discomfort/`. Accessed: 2017-06-16. Cited on 1

[12] Ayse Pinar Saygin, Thierry Chaminade, Hiroshi Ishiguro, Jon Driver, and Chris Frith. The thing that should not be: predictive coding and the uncanny valley in perceiving human and humanoid robot actions. *Oxford University Press journal Social Cognitive and Affective Neuroscience*, 2011. Cited on 1

[13] Liu Ren, Alton Patrick, Alexei A. Efros, Jessica K. Hodgins, and James M. Rehg. A data-driven approach to quantifying natural human motion. *Special Interest Group on Computer Graphics and Interactive Techniques*, 2005. Cited on 2

[14] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A Skinned Multi-Person Linear Model. *Special Interest Group on Computer Graphics and Interactive Techniques Asia*, 2015. Cited on 2

[15] Digital Photography School. 17 Forced Perspective Technique Examples. `https://digital-photography-school.com`. Accessed: 2017-04-03. Cited on 5

[16] Pixabay. Pixabay. `https://pixabay.com`. Accessed: 2017-04-03. Cited on 5

[17] Francesc Moreno-Noguer, Josep M. Porta, and Pascal Fu. Exploring Ambiguities for Monocular Non-rigid Shape Estimation. *European Conference on Computer Vision*, 2010. Cited on 6, 15

[18] Rodrigo L. Carceroni and Kiriakos N. Kutulakos. Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid 3D Motion, Shape & Reflectance. *International Conference on Computer Vision*, 2001. Cited on 8

[19] Jean-Phillipe Pons, Renaud Keriven, and Olivier Faugeras. Modelling Dynamic Scenes by Registering Multi-View Image Sequence. *Conference on Computer Vision and Pattern Recognition*, 2005. Cited on 8

[20] Mathieu Perriollat and Adrien Bartoli. A Quasi-Minimal Model for Paper-Like Surfaces. *Conference on Computer Vision and Pattern Recognition*, 2007. Cited on 8

[21] Miao Liao, Qing Zhang, Huamin Wang, Ruigang Yang, and Minglun Gong. Modeling Deformable Objects from a Single Depth Camera. *International Conference on Computer Vision*, 2009. Cited on 9

[22] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. *International Symposium on Mixed and Augmented Reality*, 2011. Cited on 10

[23] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. *Conference on Computer Vision and Pattern Recognition*, 2015. Cited on 10

[24] Carlos Hernández, George Vogiatzis, Gabriel J. Brostow, Bjorn Stenger, and Roberto Cipolla. Non-rigid Photometric Stereo with Colored Lights. *International Conference on Computer Vision*, 2007. Cited on 11

[25] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime Faces: High Resolution Capture for Modeling and Animation. *Association for Computing Machinery Annual Conference on Computer Graphics*, 2004. Cited on 11

[26] Thibaut Weise, Bastian Leibe, and Luc Van Gool. Fast 3D Scanning with Automatic Motion Compensation. *Conference on Computer Vision and Pattern Recognition*, 2007. Cited on 11

[27] Jonas Gårding. Shape from Texture for Smooth Curved Surfaces in Perspective Projection. *Journal of Mathematical Imaging and Vision*, 1992. Cited on 11

[28] Angeline M. Loh and Richard Hartley. Shape from non-homogeneous, non-stationary, anisotropic, perspective texture. *British Machine Vision Conference*, 2005. Cited on 11

[29] Ryan White and David A. Forsyth. Combining Cues: Shape from Shading and Texture. *Conference on Computer Vision and Pattern Recognition*, 2006. Cited on 12

[30] Adrien Bartoli, Vincent Gay-Bellile, Umberto Castellani, Julien Peyras, S. Olsen, and Patrick Sayd. Coarse-to-fine low-rank structure-from-motion. *Conference on Computer Vision and Pattern Recognition*, 2008. Cited on 13

[31] Matthew Brand. Morphable 3D models from video. *Conference on Computer Vision and Pattern Recognition*, 2001. Cited on 13

[32] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. Non-Rigid Structure-From-Motion: Estimating Shape and Motion with Hierarchical Priors. *Transactions on Pattern Analysis and Machine Intelligence*, 2008. Cited on 13

[33] Alessio Del Bue, Xavier Lladó, and Lourdes Agapito. Non-Rigid Metric Shape and Motion Recovery from Uncalibrated Images Using Priors. *Conference on Computer Vision and Pattern Recognition*, 2006. Cited on 13

[34] Jing Xiao, Jin xiang Chai, and Takeo Kanade. A Closed-Form Solution to Non-Rigid Shape and Motion Recovery. *European Conference on Computer Vision*, 2004. Cited on 13

[35] Chris Russell, Joao Fayad, and Lourdes Agapito. Energy Based Multiple Model Fitting for Non-Rigid Structure from Motion. *Conference on Computer Vision and Pattern Recognition*, 2011. Cited on 13

[36] Ravi Garg, Anastasios Roussos, and Lourdes Agapito. Dense Variational Reconstruction of Non-Rigid Surfaces from Monocular Video. *Conference on Computer Vision and Pattern Recognition*, 2013. Cited on 13, 16

[37] Yuchao Dai, Hongdong Li, and Mingyi He. A Simple Prior-free Method for Non-Rigid Structure-from-Motion Factorization. *International Journal of Computer Vision*, 2014. Cited on 13

[38] Christoph Bregler, Aaron Hertzmann, and Henning Bierman. Recovering Non-Rigid 3D Shape from Image Streams. *Conference on Computer Vision and Pattern Recognition*, 2000. Cited on 13

[39] Qi Pan, Gerhard Reitmayr, and Tom Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. *British Machine Vision Conference*, 2009. Cited on 14

[40] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts. *International Conference on Computer Vision*, 2007. Cited on 14

[41] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual Modeling with a Hand-Held Camera. *International Journal of Computer Vision*, 2004. Cited on 14

[42] Agisoft. Agisoft. `http://www.agisoft.com/`. Accessed: 2017-04-11. Cited on 14

[43] Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Deformable Surface Tracking Ambiguities. *Conference on Computer Vision and Pattern Recognition*, 2007. Cited on 14

[44] Mathieu Salzmann, Raquel Urtasun, and Pascal Fua. Local Deformation Models for Monocular 3D Shape Recovery. *Conference on Computer Vision and Pattern Recognition*, 2008. Cited on 15

[45] Shuhan Shen, Wenhuan Shi, and Yuncai Liu. Monocular Template-Based Tracking of Inextensible Deformable Surfaces under L2-Norm. *Asian Conference on Computer Vision*, 2009. Cited on 15

[46] Florent Brunet, Richard Hartley, Adrien Bartoli, Nassir Navab, and Remy Malgouyres. Monocular Template-Based Reconstruction of Smooth and Inextensible Surfaces. *Asian Conference on Computer Vision*, 2010. Cited on 16

[47] Mathieu Perriollat, Richard Hartley, and Adrien Bartoli. Monocular Template-based Reconstruction of Inextensible Surfaces. *International Journal of Computer Vision*, 2011. Cited on 16

[48] Abed Malti, Adrien Bartoli, and Toby Collins. A Pixel-Based Approach to Template-Based Monocular 3D Reconstruction of Deformable Surfaces. *International Conference on Computer Vision Workshops*, 2011. Cited on 16

[49] Jonas Östlund, Aydin Varol, Dat Tien Ngo, and Pascal Fua. Laplacian Meshes for Monocular 3D Shape Recovery. *European Conference on Computer Vision*, 2012. Cited on 16

[50] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian Surface Editing. *Symposium on Geometry Processing*, 2004. Cited on 16

[51] Adrien Bartoli, Yan Gérard, François Chadebecq, and Toby Collins. On Template-Based Reconstruction from a Single View: Analytical Solutions and Proofs of Well-Posedness for Developable, Isometric and Conformal Surfaces. *Conference on Computer Vision and Pattern Recognition*, 2012. Cited on 16

[52] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`. Accessed: 2017-04-13. Cited on 17

[53] Francesc Moreno-Noguer, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Capturing 3D Stretchable Surfaces from Single Images in Closed Form. *Conference on Computer Vision and Pattern Recognition*, 2009. Cited on 17

[54] Abed Malti, Richard Hartley, Adrien Bartoli, and Jae-Hak Kim. Monocular Template-Based 3D Reconstruction of Extensible Surfaces with Local Linear Elasticity. *Conference on Computer Vision and Pattern Recognition*, 2013. Cited on 17

[55] Mathieu Salzmann and Pascal Fua. Reconstructing Sharply Folding Surfaces: A Convex Formulation. *Conference on Computer Vision and Pattern Recognition*, 2009. Cited on 17

[56] Mathieu Salzmann and Pascal Fua. Linear Local Models for Monocular Reconstruction of Deformable Surface. *Transactions on Pattern Analysis and Machine Intelligence*, 2011. Cited on 17, 18, 72

[57] Magnus R. Hestenes and Eduard Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 1952. Cited on 23

[58] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Press Syndicate of the University of Cambridge, 1992. Cited on 23

[59] Stephen Boyd and Lieven Vandenberghe. *Convex Optmization*. Cambridge University Press, 2004. Cited on 24

[60] Peter Shirley and Steve Marschner. *Fundamentals of Computer Graphics*. CRC Press Taylor & Francis Group, 2016. Cited on 25

[61] OpenCV Team. OpenCV. `http://opencv.org`. Accessed: 2017-04-09. Cited on 26

[62] Tania Pouli, Erik Reinhard, and Douglas W. Cunningham. *Image Statistics in Visual Computing*. CRC Press Taylor & Francis Group, 2014. Cited on 27

[63] Reinhard Klette. *Concise Computer Vision*. Springer, 2014. Cited on 27

[64] Ujjaval Y. Desai, Marcelo M. Mizuki, Ichiro Masaki, and Berthold K.P. Horn. Edge and mean based image compression. *Technical Report*, 1996. Cited on 29

[65] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973. Cited on 29

[66] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. *Computer Vision and Pattern Recognition*, 2005. Cited on 29

[67] Satya Mallick. Histogram of Oriented Gradients. `http://www.learnopencv.com/histogram-of-oriented-gradients/`. Accessed: 2017-05-05. Cited on 30

[68] Georg Rill and Thomas Schaeffer. *Grundlagen und Methodik der Mehrkörpersimulation*. Springer, 2014. Cited on 39

[69] Jitendra Malik and Ruth Rosenholtz. Computing Local Surface Orientation and Shape from Texture for Curved Surfaces. *International Journal of Computer Vision*, 1997. Cited on 40

[70] Dat Tien Ngo, Sanghyuk Park, Anne Jorstad, Alberto Crivellaro, Chang D. Yoo, and Pascal Fua. Dense Image Registration and Deformable Surface Reconstruction in Presence of Occlusions and Minimal Texture. *International Conference on Computer Vision*, 2015. Cited on 40

[71] NVIDIA Corporation. Parallele Berechnungen mit CUDA. `http://www.nvidia.de/object/cuda-parallel-computing-de.html`. Accessed: 2017-05-08. Cited on 50

[72] John A. Nelder and Roger Mead. A simplex method for function minimization. *The Computer Journal*, 1965. Cited on 70

[73] Paul V. C. Hough. Method and Means for Recognizing Complex Patterns. *Patent*, 1962. Cited on 71

[74] Textures.com Team. Textures.com. `https://www.textures.com/`. Accessed: 2017-05-20. Cited on 71

[75] Abhimitra Meka, Michael Zollhöfer, Christian Richardt, and Christian Theobalt. Live Intrinsic Video. *Association for Computing Machinery Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH*, 2016. Cited on 72

[76] Johann H. Lambert. *Photometria sive de mensura et gradibus luminis, colorum et umbrae*. Klett, 1760. Cited on 72

[77] Qi Liu-Yin, Rui Yu, Lourdes Agapito, Andrew Fitzgibbon, and Chris Russell. Better Together: Joint Reasoning for Non-rigid 3D Reconstruction with Specularities and Shading. *British Machine Vision Conference*, 2016. Cited on 72

[78] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the Association for Computing Machinery*, 1975. Cited on 72

[79] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid Data Dreaming for Object Tracking. *arXiv*, 2017. Cited on 72

[80] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-Time Single Camera SLAM. *Transactions on Pattern Analysis and Machine Intelligence*, 2007. Cited on 73

[81] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Trajectory Space: A Dual Representation for Nonrigid Structure from Motion. *Transactions on Pattern Analysis and Machine Intelligence*, 2010. Cited on 73