Estimating and Simulating Structure and Motion

Jensen, Janus Nørtoft

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

Ph.D. Thesis

DTU Compute
Department of Applied Mathematics and Computer Science

Estimating and Simulating Structure and Motion

Janus Nørtoft Jensen

Kongens Lyngby 2020

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Richard Petersens Plads
Building 324
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

ISSN: 0909-3192

Summary

Estimation and simulation of structure and motion are key topics in computer vision, computer graphics, and augmented reality. There are many connections between these terms. Structure and motion are naturally connected, and knowledge of either implies knowledge about the other. When dealing with images, estimation and simulation are, in some aspects, the opposites of each other. When we estimate, we extract information from images, and when we simulate the imaging process, we create images given information about the appearance of the scene.

In recent years, the interplay between these four terms has grown larger. An example of this is in augmented reality applications where simulation and estimation must work together. Accurate estimation of the surrounding environment is necessary to create realistic, digitally overlaid content and immersive experiences.

In this thesis, we present work that lies in the span of estimation, simulation, structure, and motion. Our goal is to explore synergies between simulation and estimation concerning structure and motion. We investigate how we can use simulations to make better estimations and how we can use estimations to create realistic simulations. The work is split into three areas. Estimating motion to interpolate frames in video sequences, estimating geometry to create illusory motion of physical objects, and simulating the imaging process to estimate geometry.

Our contributions demonstrate that we can use simulations to estimate geometry better and that by estimating structure and motion, we can simulate novel camera viewpoints in time and space that can be used to create both video frame interpolations and immersive augmented reality experiences.

Danish Summary

Estimering og simulering af struktur og bevægelse er nøgleemner i computer vision, computergrafik og augmented reality. Der er mange forbindelser mellem disse termer. Struktur og bevægelse er naturligt forbundede, og viden om den ene medfører viden om den anden. Når man beskæftiger sig med billeder, så er estimering og simulering, i nogle aspekter, hinandens modsætninger. Når vi estimerer, så udtrækker vi information fra billeder, og når vi simulerer billeddannelse, så danner vi billeder givet information om scenens udseende.

I de senere år er samspillet mellem disse fire termer blevet større. Et eksempel på dette er i augmented reality anvendelser, hvor simulering og estimering skal arbejde sammen. At estimere de omkringliggende omgivelser nøjagtigt er nødvendigt for at skabe realistisk, digitalt overlejret indhold og fordybende oplevelser.

I denne afhandling præsenterer vi arbejde der ligger i spændet af estimering, simulering, struktur og bevægelse. Vores mål er at udforske synergier mellem simulering og estimering vedrørende struktur og bevægelse. Vi undersøger hvordan vi kan bruge simuleringer til at lave bedre estimater, og hvordan vi kan bruge estimater til skabe realistiske simuleringer. Arbejdet er opdelt i tre områder. Estimering af bevægelse for at interpolere billeder i videosekvenser, estimering af geometri for at skabe illusorisk bevægelse af fysiske objekter og simulering af billeddannelsesprocessen for at estimere geometri.

Vores bidrag viser, at vi kan bruge simuleringer til at estimere bedre geometri, og at ved at estimere struktur og bevægelse kan vi simulere nye kamerasynspunkter i tid og rum, der kan bruges til at skabe både interpolationer i videosekvenser og fordybende augmented reality-oplevelser.

Preface

This PhD thesis was prepared at the section of Image Analysis and Computer Graphics at the section of Image Analysis and Computer Graphics at the Technical University of Denmark (DTU) in fulfillment of the requirements for acquiring a PhD degree in computer science. The work presented in this was funded by DTU Compute.

The PhD was supervised in the last two years by Professor Anders Bjorholm Dahl and in the first year by Henrik Aanæs. Co-supervisors were Associate Professor Jakob Andreas Bærentzen and Associate Professor Jakob Wilm.

Janus Nørtoft Jensen

Acknowledgements

First and foremost, I would like to thank my main supervisor, Professor Anders Bjorholm Dahl, for his great help and guidance through my PhD's final years. Especially while writing this thesis, Anders has been a tremendous support and always ensuring that I moved in the right direction.

I would also like to thank my other supervisors, Associate Professor Jakob Andreas Bærentzen, Associate Professor Jakob Wilm, and Henrik Aanæs, who was my main supervisor for the first year of my PhD. Thank you to Professor Serge Belongie and his group to host my external research stay at Cornell Tech.

Many thanks to all my great colleagues and friends at the section for Image Analysis and Computer Graphics, DTU Compute. There has always been someone with whom I could have fruitful discussions or just a cup of coffee. Special thanks to Morten Hannemose, with whom I have done most of my research during my PhD, and who is always good at calling me out when I say something wrong or have when I have bad ideas.

List of Contributions

Thesis contributions

A Morten Hannemose, **Janus Nørtoft Jensen**, Gudmundur Einarsson, Jakob Wilm, Anders Bjorholm Dahl, and Jeppe Revall Frisvad. "Video Frame Interpolation via Cyclic Fine-Tuning and Asymmetric Reverse Flow". In: *Scandinavian Conference on Image Analysis*. Springer. 2019, pages 311–323. [Han+19]

B **Janus Nørtoft Jensen**, Morten Hannemose, Jakob Wilm, Anders Bjorholm Dahl, Jeppe Revall Frisvad, and Serge Belongie. "Generating spatial attention cues via illusory motion". In: *Third Workshop on Computer Vision for AR/VR*. 2019. URL: http://people.compute.dtu.dk/jnje/illusory-motion/. [Jen+19]

C **Janus Nørtoft Jensen**, Morten Hannemose, Jakob Andreas Bærentzen, Jakob Wilm, Jeppe Revall Frisvad, and Anders Bjorholm Dahl. "Surface Reconstruction from Structured Light Images using Differentiable Rendering". Submitted to: *Sensors*. 2020. [Jen+20]

Other contributions - not included in thesis

- Gudmundur Einarsson, **Janus Nørtoft Jensen**, Rasmus R Paulsen, Hildur Einarsdottir, Bjarne K Ersbøll, Anders B Dahl, and Lars Bager Christensen. "Foreign Object Detection in Multispectral X-ray Images of Food Items Using Sparse Discriminant Analysis". In: *Scandinavian Conference on Image Analysis*. Springer. 2017, pages 350–361. [Ein+17]

- Jannik Boll Nielsen, Eyþór Rúnar Eiríksson, Rasmus Ahrenkiel Lyngby, Jakob Wilm, **Janus Nørtoft Jensen**, Henrik Aanæs, and David Bue Pedersen. "PicPrint: Embedding pictures in additive manufacturing". In: *euspen and ASPE Special Interest Group Meeting: Additive Manufacturing*. 2017. [Nie+17]

- Rasmus Ahrenkiel Lyngby, Jakob Wilm, Eyþór Rúnar Eiríksson, Jannik Boll Nielsen, **Janus Nørtoft Jensen**, Henrik Aanæs, and David Bue Pedersen. "Inline 3D print failure detection using computer vision". In: *euspen and ASPE Special Interest Group Meeting: Additive Manufacturing*. 2017. [Lyn+17]

- **Janus Nørtoft Jensen**, Rasmus Ahrenkiel Lyngby, Henrik Aanæs, Eyþór Rúnar Eiríksson, Jannik Boll Nielsen, Jakob Wilm, and David Bue Pedersen. "Photogrammetry for Repositioning in Additive Manufacturing". In: *euspen and ASPE Special Interest Group Meeting: Additive Manufacturing*. 2017. [Jen+17]

- Jakob Wilm, Daniel González Madruga, **Janus Nørtoft Jensen**, Søren Kimmer Schou Gregersen, Mads Emil Brix Doest, Maria Grazia Guerra, Henrik Aanæs, and Leonardo De Chiffre. "Effects of subsurface scattering on the accuracy of optical 3D measurements using miniature polymer step gauges". In: *euspen's 18th International Conference and Exhibition*. 2018, pages 449–450. [Wil+18]

- Lars B. Dahlin, Kristian R. Rix, Vedrana A. Dahl, Anders B. Dahl, **Janus Nørtoft Jensen**, Peter Cloetens, Alexandra Pacureanu, Simin Mohseni, Niels O.B. Thomsen, and Martin Bech. "Three-dimensional architecture of human diabetic peripheral nerves revealed by X-ray phase contrast holographic nanotomography". In: *Scientific reports* 10.1 (2020), pages 1–8. [Dah+20]

Contents

CHAPTER 1

Introduction

As suggested by this thesis's title, the work presented here is related to four keywords: structure, motion, estimation, and simulation. The contributions of the thesis primarily lie in these four areas and the interface between them.

Structure and motion in images have been important topics for researchers in computer vision and computer graphics for many decades and continues to be important in many applications today. Sometimes they are studied independently of each other, but they are often also studied together, as either holds much information about the other. Image projections of an object in motion allow for reconstruction of both the 3D geometry, or structure, and the object's motion. This process is known as structure from motion [Ull79]. Reversely, if the structure of a scene in motion is known, it is not very complicated to determine how this translates to motion in images.

Estimation and simulation are two very central concepts in vision and graphics. By estimation, we mean the process of producing approximations. By simulation, we mean the process of imitating a process or behavior. Traditionally, computer vision's main task has been to estimate properties from images such as object geometry or appearance, whereas computer graphics have dealt with simulating the imaging process by synthesizing images given their appearance attributes. Since the late '90s, however, the line between these fields has become less evident. The vision community has seen an increasing interest in generating content, e.g., in computational photography, and the graphics community has seen increased interest in estimating, e.g., photometric properties. The resurgence of virtual- and augmented reality, and the problems that have arisen from this, have also prompted more interaction between vision and graphics to solve those problems.

The interface between the four terms is manifold. As mentioned, structure and motion are inherently linked. Estimating them are classic problems in computer vision, and synthesizing (or simulating) images using known structure and motion is fundamental to graphics. Estimating and simulating can thus, to some extent, be considered the dual of each other. When we estimate, we extract information given images, and when we simulate, we create images given information. Much research focuses on either estimation or simulation. With this thesis's work, we seek to take advantage of this dualism and explore synergies between estimating and simulating concerning structure and motion to achieve better results.

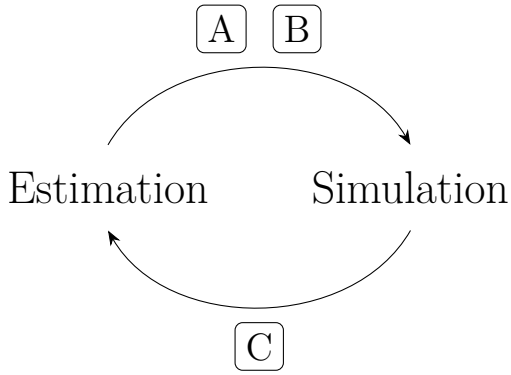


Figure 1.1: Visualizing how the contributions of this thesis uses simulations to create estimations and uses estimations to create simulations.

1.1 Scope & Objectives

This thesis's focus is to investigate applications in the interface between structure, motion, estimation, and simulation, and if interaction between the terms can be used in synergy to achieve better results. More concretely, we want to examine if we can use simulations to create better estimations and use estimations to create better simulations. Specifically, we aim to accomplish the following objectives:

- Use estimated motion to simulate motion and interpolate images.
- Use estimated geometry to simulate physical motion.
- Use simulated imaging process to estimate geometry.

Contributions A to C each deals with one of these objectives. The contributions either use estimations to do simulations or vice versa. This is illustrated in Figure 1.1.

Estimate Motion in Images to Simulate Motion and Interpolate Images We want to investigate if we can estimate motion in a temporal image sequence and use this information to interpolate images at novel time points. Motion between images is often estimated by relating changes in image intensity at a point to movement of the intensity pattern where the points in the pattern are assumed to have constant intensity. Our goal is to let a statistical model learn to estimate motion in image sequences and interpolate images between the originals by showing the model how the interpolated images are supposed to look. In this way, the model learns to predict motion in a self-supervised manner. As part of this goal, an objective is to drop the assumption of constant intensity and let the model gain knowledge about the world by itself and possibly deviate from the assumption. Not enforcing this assumption

enables the model to possibly learn a better motion representation suitable for the task of interpolating images.

Current methods to estimate motion are generic and use the same model to estimate the motion for all image sequences. We aim to transcend those methods by developing a model that can bootstrap itself and learn to be particularly good for the specific sequence that it is interpolating or going even deeper and optimize itself to be better for each individual frame that is being created using the model.

Use Estimated Geometry to Simulate Physical Motion With this objective, the goal is to use geometry estimation to meaningfully interact with the physical world and create the illusion that real-world objects are moving when they, in fact, are not. We aim to do this by projecting specific light patterns onto the objects, tuned in a way to create a very subtle effect.

Movement is a strong attention cue, and making objects appear to move can be used to grab people's attention to specific objects. Some attention cues are better than others for specific tasks. Subtle attention cues, such as the one we aim to create, is possibly preferred to more crude cues in certain situations. Such a situation might be to ensure compliance, e.g. if we want to remind a person to take their pills. Another situation might be that we want to hint at the next clue in an escape room.

This objective is an example of an augmented reality application. To achieve the goal successfully it is imperative that the experience is immersive. The digitally overlaid perceptual information needs to blend in seamlessly with the physical world to make it immersive. To blend seamlessly and create an immersive experience, we estimate accurate geometry and appearance parameters for the object in focus. With these estimations, we want to create an image from the projector's point of view and adapt the light pattern to simulate motion of the object in the physical world realistically.

Use simulated imaging process to estimate geometry Estimating object geometry from images is typically a two-step process. The first step is to produce a set of points sampled from the object's surface, called a point cloud. The second step is to produce a surface from the sampled points. The first step encompasses several sub-steps. When reconstructing an object's surface from structured light images, the structured light is first decoded, the resulting images are undistorted and rectified, point correspondences between cameras (or projectors) are found, and points are triangulated. With each step, we potentially throw away information and add noise that will influence the end result. When a surface is reconstructed from a point cloud, all prior information is usually discarded. This makes tracing the information and error flow from input images to the final result very complicated.

Our goal is to avoid all of the sub-steps as well as the intermediate point cloud representation and estimate surface geometry directly from the input images. To do this, we simulate the imaging process to recreate the captured structured light images.

By comparing rendered images with captured images and measuring the difference we can directly relate the parameters of our reconstructed surface with the original input data in a meaningful way. When iteratively updating the surface geometry, we improve the simulation until the simulated images look like the captured images.

This approach makes use of the duality between estimating parameters from images and producing images given some parameters. Instead of treating surface reconstruction as a backward problem of estimating geometry from images, we recast the problem as a forward problem of simulating the imaging process used to capture the images.

Our aim with the approach is to improve the surface reconstruction. Both in terms of producing a more accurate reconstruction, but also in terms of traceability from input to output. By simulating the imaging process, we can directly measure how each pixel in the images affect the resulting surface geometry.

1.2 Thesis Structure

The thesis is divided into chapters whose content relates to the previous section's objectives.

Chapters 2 and 3 contains background information and related work associated with the objectives and contributions of this thesis. Chapter 2 gives a description of central aspects of the image formation process, how it can be modeled, and about motion in images. Chapter 3 introduces aspects of geometry and shape acquisition in computer vision.

In Chapter 4 we present the contributions of the thesis, and in Chapter 5 we conclude and return to the objectives described in the section above.

CHAPTER 2

Digital Images

Whether we want to extract information from images or render synthetic images, it is necessary to understand how images are formed from photons passing through the lens of a camera and hitting the image sensor, followed by being digitized into the array of pixels we call a digital image.

This chapter will cover the basic theory behind how digital images are formed and how we can model this process. This includes how we mathematically can describe a camera, how we can fit the model's parameters to a real camera, how photons are converted into pixel values, and how noise occurs in that process. We will also briefly cover how motion in images can be described, introduce a method for simulating it and give a short description of convolutional neural networks, a standard tool used to estimate parameters in images.

2.1 Imaging process

In this section, we will first describe the imaging process and how we can model it. Firstly, we will describe how a digital image is formed and how we can describe noise in the image. Secondly, we will describe how we can model a camera as perspective projection and describe how we can use this model for a real camera.

2.1.1 Image formation and noise

Images taken with a digital camera are the results of the image sensor converting photons into an array of digital values through a sequence of steps. Conventional digital cameras do not directly measure the photons hitting the sensor but measure each pixel's irradiance during the exposure time t. Photons are converted into electrons using the photoelectric effect. The electrons form a charge, which is converted into voltage by a capacitor, which subsequently is amplified and digitized [Eur16; Die04]. This process is depicted in Figure 2.1.

The imaging process is not deterministic, and real-world images taken with a digital camera will always contain some amount of noise. How much depends on the quality of the image sensor and on the amount of light in the environment. When photons

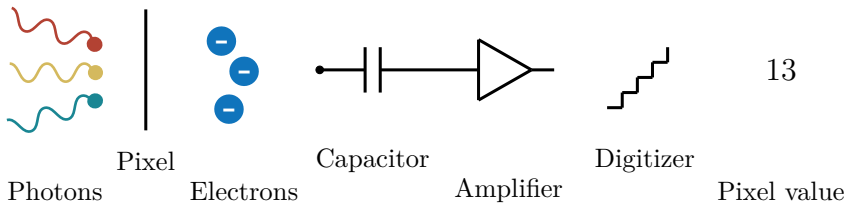


Figure 2.1: From photons to pixel values. Photons hit a pixel and are converted into electrons, which are converted into a voltage, which is amplified and digitized. Figure adapted from [Die04]

arrive at the image sensor, they do so following a Poisson distribution. If μ_e is the mean number of electrons that have been created from photons, then it follows that it has variance

$$\sigma_e^2 = \mu_e. \tag{2.1}$$

This noise term is usually referred to as shot noise or photon noise.

Even if no light reaches the image sensor, there will still be some amount of electrons present. This is called the dark signal and has mean μ_d and variance σ_d. The dark signal comes mainly from thermally induced electrons. Noise arising from the camera electronics, such as the voltage conversion and amplification, is typically included in σ_d, which is called the readout noise and is Gaussian-distributed and independent of the signal. The mean of the digital signal y comes out to

$$\mu_y = K\left(\mu_e + \mu_d\right), \tag{2.2}$$

where the single parameter K is the overall system gain, which covers the process of converting electrons into voltage, amplifying the signal, and digitizing it with an analog-to-digital converter.

Combining the different noise terms, the variance of the digital signal y, or the noise, becomes

$$\sigma_y^2 = K^2\left(\sigma_d^2 + \sigma_e^2\right). \tag{2.3}$$

2.1.2 Modelling a camera

To both understand image formation and how to simulate it, we need a way of mathematically describing a camera. The most common way of doing this is by using the pinhole model, sometimes also referred to as perspective projection [HZ04]. The projection of a 3D world point to an image plane is found by intersecting the image plane and the line that joins the 3D point with an optical center, which is another point in space that defines the camera's position. The distance f from this point to the image plane is called the focal length. If $[X, Y, Z]^\mathsf{T}$ is the 3D point in space and

$[x, y]^\mathsf{T}$ is the corresponding point in the image plane, the latter can be found by means of equal triangles (see Figure 2.2)

$$x = f\frac{X}{Z} \tag{2.4}$$

$$y = f\frac{Y}{Z}. \tag{2.5}$$

With homogeneous coordinates, this projection is a linear mapping which can be described by a 3x4 projection matrix $\mathbf{P}$, i.e., if $\mathbf{q}$ is the 3D point in space in and $\mathbf{p}$ is the projected point in the image plane, both in homogeneous coordinates, then

$$\mathbf{p} = \mathbf{P}\mathbf{q} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{q} \tag{2.6}$$

The focal length f is one of the internal camera parameters that are used to convert image coordinates in world units to pixel coordinates. In the pinhole camera model, the internal parameters are given by the intrinsic matrix A defined as

$$\mathbf{A} = \begin{bmatrix} f & \beta & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1, \end{bmatrix} \tag{2.7}$$

where (c_x, c_y) is the intersection point between the image plane and the line going from the optical center perpendicular to the image plane. This is known as the principal point, and the line is called the optical axis. The parameters α and β are parameters used to handle non-square and sheared pixels. With the quality of modern cameras, these are often not included and $\alpha = 1$ and $\beta = 0$.

Finally, the position of a camera may not coincide with the origin of the world space, and it may have an arbitrary orientation. This is accounted for by transforming the 3D world coordinates with a rotation vector $\mathbf{R}$ and translation vector $\mathbf{t}$. These are known as the extrinsic parameters of the camera.

Putting together the extrinsic parameters and intrinsic parameters the pinhole camera model is defined by

$$\mathbf{p} = \mathbf{P}\mathbf{q} = \mathbf{A}\,[\mathbf{R}\ \mathbf{t}]\,\mathbf{q}. \tag{2.8}$$

Non-linear distortion In some cases, we want a more precise camera model. The linear assumption that straight lines are imaged as straight lines will, in general, not hold for real cameras due to the many lens elements a modern camera lens usually contains. This problem is typically alleviated by extending the pinhole camera model to correct for lens distortion with the Brown-Conrady distortion model [Bro66; Con19]. Two components comprise this model; one accounting for radial distortion and one

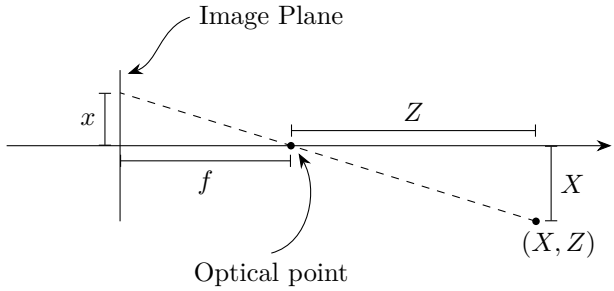


Figure 2.2: Caption

accounting for tangential distortion. Radial distortion is usually the most significant part of this and is in the Brown-Conrady model modeled as a correction term by the polynomial function

$$\mathbf{r}(x, y) = \left(k_1 r^2 + k_2 r^4 + k_3 r^6\right) \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2.9}$$

with $x = (u - c_x)/f$, $y = (v - c_y)/f$ and $r = \sqrt{x^2 + y^2}$ is the radius from the optical axis. The parameters k_1, k_2, and k_3 are internal parameters of the camera called distortion coefficients. The tangential distortion component is added to correct for misalignment between the various optical lens elements in the camera lens and is modeled by the term

$$\mathbf{t}(x, y) = \begin{bmatrix} 2p_1 xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy, \end{bmatrix}, \tag{2.10}$$

with p_1, and p_2 being internal tangential distortion coefficients. For expensive, high-quality lenses, this term is often very insignificant.

Combining the two terms the Brown-Conrady distortion model maps distorted image coordinates (u, v) to undistorted coordinates (u', v') by

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{r}(x, y) + \mathbf{t}(x, y) \tag{2.11}$$

2.1.2.1 Camera Calibration

When a real camera is modelled by the pinhole camera model the camera needs to be calibrated, meaning that the parameters $\mathbf{A}$, $\mathbf{R}$, $\mathbf{t}$, and possibly the distortion parameters $\mathbf{k} = [k_1, k_2, k_3, p_1, p_2]^\mathsf{T}$ needs to be determined. The de facto standard method for doing is the one presented by Zhang [Zha99]. In this method, a planar pattern with discernible features with known relative distances is imaged. From the image locations of these features in a single image, a homography can be established,

determining the relative pose between the camera and the planar object and admits constraints on the intrinsic parameters. With multiple images, where either the camera or the planar object has moved, the intrinsics $\mathbf{A}$ can be determined with a linear closed-form solution. The solution is refined by a non-linear optimization step which minimizes the sum of squared reprojection errors

$$\underset{\mathbf{A},\mathbf{k},\{\mathbf{R}_i\},\{\mathbf{t}_i\}}{\arg\min} \sum_{i=1}^{n} \sum_{j=1}^{m} \|\mathbf{u_{ij}} - \breve{\mathbf{u}}\left(\mathbf{A}, \mathbf{k}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{q}_j\right)\|^2, \tag{2.12}$$

where $\mathbf{u}_{ij}$ is the observed image coordinates of the j^{th} point in the i^{th} image, $\breve{\mathbf{u}}\left(\mathbf{A}, \mathbf{k}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{q}_i\right)$ is the projection of the point $\mathbf{q}_j$ into the i^{th} image as defined by Equations (2.8) and (2.11). The points $\mathbf{q}_j$ is defined in the coordinates of the planar object as $\mathbf{q}_j = [x, y, 0]^\mathsf{T}$, and thus the camera pose given by $\mathbf{R}_i$ and $\mathbf{t}_i$ is relative to the pose of the planar calibration object.

2.1.2.2 Projector Calibration

For some applications, it is necessary to have a calibrated camera and a calibrated projector. Contribution B is an example of such an application. A projector can be considered an inverse camera, as the light rays being emitted from the projector can be described in the same projective way as the camera captures light rays. The only difference is the direction of the light rays, and therefore a projector is commonly modeled as a pinhole camera. However, calibrating a projector is not possible to do in the same fashion as for a camera since the projector cannot observe features on a calibration object.

Some approaches to calibrating a projector are based on precise positioning and movement of a calibration object relative to the projector [Guo+05; Che+09]. Other approaches project feature points onto planes whose pose are determined with a calibrated camera [KMK07; Sad+05; SC08].

Structured light sequences are used in other approaches to create a mapping between camera pixel coordinates and projector pixel coordinates. This mapping allows for creating images of a calibration object from the projector's point of view [ZH06]. One method using this concept is presented by Moreno and Taubin [MT12]. Their approach uses the same method for projector calibration as for camera calibration, after determining the position of feature points on a calibration object in the projector image plane. To determine these, they project structured light sequences onto the calibration object. Decoding the structured light sequences observed by the camera yields a mapping between pixel coordinates in the camera's image plane and the image plane of the projector. This mapping is then used to map the feature points from camera image coordinates to projector image coordinates. However, the mapping is only defined precisely in the camera's pixel locations and cannot be used directly to map from image coordinates, not in exact pixel locations. Therefore, it is necessary to

interpolate between pixel locations. The interpolation is done using local homographies. For each feature point in the camera, a local homography $\mathbf{H}$ is fitted to a small patch of pixels around the point by solving

$$\mathbf{H} = \arg\min_{\widetilde{\mathbf{H}}} \sum_{\mathbf{u}_c} \left\| \mathbf{u}_p - \widetilde{\mathbf{H}}\mathbf{u}_c \right\|^2, \tag{2.13}$$

where $\mathbf{u}_c$ are the pixel coordinates in the image and $\mathbf{u}_p$ are the corresponding projector coordinates. Other methods use a global homography[PP10]. The advantage of using local homographies is that a linear relationship is only assumed close to each feature point. This approach allows for modeling non-linear distortion of the projector, which is significant for many projectors.

2.1.2.3 Stereo Calibration

Determining the relative pose between a number of cameras (or projectors) is called stereo calibration. Each camera's pose is given by a rotation matrix $\mathbf{R}_c$ and a translation vector $\mathbf{t}_c$, which relates it to the world coordinate system. Often this coordinate system is defined by the position and orientation of one of the cameras. The relative pose between two cameras can be found from a series of images of a planar calibration board, similar to camera calibration. It is a necessity that the calibration board is visible on both cameras. As for camera calibration, a linear solution is first found from point correspondences, and this is followed by a non-linear optimization step [HZ04]. In the optimization step, the stereo reprojection error is minimized, defined by

$$\begin{aligned} \arg\min_{\{\mathbf{R}_c\}\{\mathbf{R}_i\},\{\mathbf{t}_i\}} \sum_{i=1}^{n}\sum_{j=1}^{m} & \|\mathbf{u}_{\mathbf{1},\mathbf{ij}} - \breve{\mathbf{u}}\left(\mathbf{A_1}, \mathbf{k_1}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{q}_j\right)\|^2 \\ & + \|\mathbf{u}_{\mathbf{2},\mathbf{ij}} - \breve{\mathbf{u}}\left(\mathbf{A_2}, \mathbf{k_2}, \mathbf{R}_c\mathbf{R}_i, \mathbf{R}_c\mathbf{t}_i + \mathbf{R_c}, \mathbf{q}_j\right)\|^2. \end{aligned} \tag{2.14}$$

Again, $\breve{\mathbf{u}}$ is projection into an image defined by its input parameters, $\mathbf{u}_{1,ij}$ and $\mathbf{u}_{2,ij}$ are the observed image coordinates in camera one and two, respectively, of the j^{th} point in the i^{th} image, $\mathbf{q}_j = [x, y, 0]^{\mathsf{T}}$ are the 3D points defined in the coordinate system of the calibration board, $\mathbf{R}_i$ and $\mathbf{t}_i$ is the pose of the first camera relative to the calibration board, and $\mathbf{R}_c$ and $\mathbf{t}_c$ is pose of the second camera relative to the first, which is what we ultimately are interested in.

2.2 Motion in images

An image captures a single moment in time and is inherently static. However, in sequences of images, such as videos, objects may move from one image to the next or appear to move. Motion in image sequences has been studied extensively for many years. This includes determining and describing the motion and creating new image sequences from a single image, where motion has been induced.

2.2.1 Estimating motion

Estimating motion in images has seen interest in many applications. The motion may arise from either the observer's movement, movement of the observed scene, or both. When we estimate motion in images, distinguishing the cases can be challenging without prior knowledge. What we estimate is thus the relative motion. The most common way of describing motion in images is in terms of optical flow.

2.2.1.1 Optical flow

Optical flow is a very general way of describing motion in images, in which each pixel is assigned an independent motion vector describing the movement of that pixel. Optical flow is typically computed between subsequent images in a sequence under the assumption that a pixel's intensity in the image is constant but merely shifts position with time. This assumption means that optical flow can be determined as the motion of the intensity pattern. The motion is traditionally determined by relating it to the change in image brightness [HS81]. The assumption that the intensity of a pixel (x, y) at time t is constant but moves to a new location with time yields the equation

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t), \tag{2.15}$$

where $I(x, y, t)$ is the intensity at location (x, y) at time t, and $\Delta x, \Delta y, \Delta t$ are the shifts in position and time. Assuming that the motion is small, the right-hand side can be linearized and letting $\Delta t \to 0$ we end up with

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0, \tag{2.16}$$

which is known as the brightness constancy constraint. This is an equation in two unknowns $\frac{dx}{dt}$ and $\frac{dy}{dt}$, which is exactly the optical flow. To compute the flow, an additional constraint is necessary. Various constraints have been proposed, such as constant optical flow in small regions [LK+81] or regularising the flow field with a smoothness term [HS81].

2.2.2 Simulating motion

What humans perceive may not always coincide with reality. Our visual system tries to process sensory information and adapt it to standard viewing situations. Sometimes the adaptions cause us to interpret a scene incorrectly [BP06]. This effect causes rise to what is known as optical, or visual, illusions. An important class of optical illusions is those where we perceive motion even when there is no motion.

2.2.2.1 Motion Without Movement

In Contribution B we make use of a visual illusion by Freeman, Adelson, and Heeger [FAH91] called 'Motion Without Movement'. This method derives patterns from a

single still image, where the patterns appear to be moving. They base their method on the phenomena that our visual system interprets local phase changes as global motion. To create the phase change, the image $\mathbf{I}(x, y)$ is convolved with two filters that are identical except shifted in phase by 90 degrees, and the results are interpolated. Specifically, the second derivative of a Gaussian $\mathbf{G}$ and its Hilbert transform $\mathbf{H}$ are used. The image sequence is computed by

$$\mathbf{D}(x, y, t) = \cos(\omega t) \left(\mathbf{I} * \mathbf{G}\right)(x, y) + \sin(\omega t) \left(\mathbf{I} * \mathbf{H}\right)(x, y), \tag{2.17}$$

where $*$ is the convolution operator and ω controls the motion's temporal frequency.

The result of (2.17) is a sequence of interpolated filter responses. When watched at a constant frame rate, they a clear illusion of motion. However, it is interpolated filter responses, which will look quite different from the original still image. To make the original image appear to be moving, we can simply add some amount of the filter response, controlled by a parameter α to the original image

$$\mathbf{I}_m(x, y, t) = \mathbf{I}(x, y) + \alpha \mathbf{D}(x, y, t). \tag{2.18}$$

2.3 Convolutional neural networks

Convolutional neural networks are a commonly used tool to estimate properties in images. They are very versatile and have shown strengths for many complicated visual tasks, such as image classification [KSH17; He+16], pose estimation [BT16; Bul+20], and image denoising [UVL18; Bro+19].

Feedforward neural networks are function approximators which maps an input $\mathbf{x}$ to an output $\mathbf{y} = f(\mathbf{y}; \theta)$ where θ are parameters that the model learns by seeing examples of $\mathbf{x}$ and $\mathbf{y}$ [Goo+16].

The function f is typically composed of a number of layers $f_1, f_2, \ldots, f_n$

$$f(x) = f_n \circ f_{n-1} \circ \cdots \circ f_2 \circ f_1(x). \tag{2.19}$$

Learning the layers' parameters is usually done using the back-propagation algorithm [RHW86], also called reverse-mode automatic differentiation.

In convolutional neural networks, the layers are made up of a set of image filters that are convolved with the output of the previous layer (or the input for the first layer), and the weights in the filter are the learned parameters.

In Contribution A we use a convolutional neural network to predict motion from an unknown frame to its two neighboring frames in a video sequence.

CHAPTER 3

Geometry in Computer Vision

The initial objective of the field computer vision was to mimic human vision as part of an attempt to replicate human intelligence in computers. One of the earliest tasks was that of recovering the 3D structure of a scene from images thereof. This task is still an active area of research. In Contribution C, we present a novel method for reconstructing surfaces in a structured light setup.

Typically, reconstructing a continuous surface is done in two steps, which may include several substeps. The first step is to sample points from the boundary of the object. These point samples are usually referred to as a point cloud. The second step is then to produce a continuous surface from the point cloud.

To generate a point cloud, we need a way of computing the distance, or depth, between points on the object and the cameras observing it. Different computer vision algorithms attempt to mimic various depth cues that the human visual system uses to achieve this. One of the cues human perception uses to infer depth, and the one most algorithms replicates, is stereopsis. Stereopsis is the depth perception that arises from binocular disparity, i.e., the difference in position of a point between the two different retinal images, that our eyes see from their different positions [HR95].

3.1 Point triangulation

Using binocular disparity in two or more images of the same scene, acquired from different positions, to determine the distances from a point on an object to the cameras is called point triangulation.

Point triangulation is the process used to sample a point cloud from the surface of an object, which subsequently can be used to recover the full surface. Light from a 3D point travels in straight lines onto the image sensors of the cameras observing it. These lines can be determined from the observed image coordinates that the 3D point projects to and known camera positions. The 3D position of the point is where the lines intersect. Figure 3.1 illustrates this concept. To mathematically describe

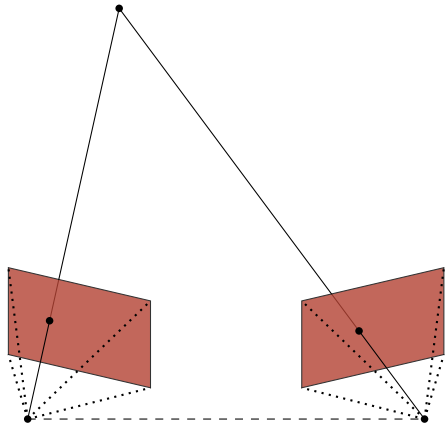


Figure 3.1: Estimation of 3D point by triangulation.

this, suppose that $\mathbf{p}_1 = [u_1, v_1, 1]^\mathsf{T}$ and $\mathbf{p}_2 = [u_2, v_2, 1]^\mathsf{T}$ are projections of a 3D point $\mathbf{q} = [X, Y, Z, 1]^\mathsf{T}$ into two calibrated cameras defined by projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$. The task is then to recover $\mathbf{q}$. If the i^{th} row of $\mathbf{P}_j$ is denoted $\mathbf{P}_j^i$ then a simple linear method arises from rearranging the equations given by the projections

$$s_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P_1^1} \\ \mathbf{P_1^2} \\ \mathbf{P_1^3} \end{bmatrix} \mathbf{q}, \text{ and } s_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P_2^1} \\ \mathbf{P_2^2} \\ \mathbf{P_2^3} \end{bmatrix} \mathbf{q}, \tag{3.1}$$

to form the linear system

$$\underbrace{\begin{bmatrix} \mathbf{P}_1^3 u_1 - \mathbf{P}_1^1 \\ \mathbf{P}_1^3 v_1 - \mathbf{P}_1^2 \\ \mathbf{P}_2^3 u_2 - \mathbf{P}_2^1 \\ \mathbf{P}_2^3 v_2 - \mathbf{P}_2^2 \end{bmatrix}}_{\mathbf{B}} \mathbf{q} = \mathbf{0}, \text{ s.t. } \mathbf{q} \neq 0, \tag{3.2}$$

In real situations, all parts used to construct $\mathbf{B}$ stem from noisy measurements and thus will contain errors. Instead of finding the null space of $\mathbf{B}$ it is therefore common to determine $\mathbf{q}$ by

$$\arg\min_{\mathbf{q}} \|\mathbf{B}\mathbf{q}\|^2, \qquad \text{s.t. } \|\mathbf{q}\| = 1. \tag{3.3}$$

One way of solving this is by choosing the unit singular vector corresponding to the smallest singular value of $\mathbf{B}$ [HZ04]. This linear method often produces good results. Analysis, however, reveals that when solving the problem in Equation (3.3), observations from cameras further away are weighted more [HZ04]. To obtain a better estimate of $\mathbf{q}$, the reprojection errors can be minimized, similarly to camera calibration

in Section 2.1.2.1. The result of Equation (3.3) is then used as a linear initializer for this minimization problem.

Before we can start triangulating points, we need to observe the same point in multiple cameras. Determining image points coming from the same 3D point is known as the correspondence problem.

3.2 Structured Light Scanning

We typically distinguish between images acquired with passive lighting and images acquired with active lighting when determining image correspondences between two or several images. Structured light scanning is an imaging technique that falls in the latter category and is used to acquire the shape of objects in the form of a point cloud. As opposed to passive methods, such as multi-view stereo, where correspondences typically are determined as points with similar neighborhoods in the image space [Sei+06], structured light actively illuminates the scene with coded patterns. The coding of the patterns makes finding correspondences between images a less complicated task.

The main principle in structured light scanning is that one or more cameras acquire images of a scene illuminated by several patterns from an active light source, typically a projector. The patterns encode the scene with information from which dense correspondences from the projector to the cameras or between the cameras can easily be determined and subsequently triangulated. The result is a dense 2.5D depth image, which easily can be converted into a point-cloud or a triangle mesh.

3.2.1 Phase Shifting

There exists an abundance of coding strategies for structured light. In this section, we will focus on the commonly used phase-shifting patterns. For an overview of various structured light coding strategies, see [Sal+10]. Phase-shifting patterns uniquely encodes every point along the coding axis by a phase value. The coding axis is typically the projector's horizontal axis. Objects will deform the recorded pattern observed by a camera. Suppose the phase value in each camera pixel is determined. This yields correspondences between the camera pixels and projector pixels. From these correspondences, points can be triangulated to recover a point cloud from the object's surface. If using more than one camera, and the triangulation is done between the cameras, the patterns can be considered a way of adding a unique texture to the object. This texture makes it significantly easier to determine dense correspondences between the cameras.

Phase-shifting patterns encode the scene with N sinusoidal patterns of the form

$$\mathbf{I}_{p,n}(u_p) = \frac{1}{2} + \frac{1}{2}\sin\left(2\pi\left(f\frac{u_p}{N_p} - \frac{n}{N}\right)\right), \tag{3.4}$$

where f is the frequency (or the number of periods within the projector), u_p is the projector column, N_p is the total number of projector columns, and $n \in 1, \ldots, N$ is the index of the pattern. Note here that the pattern only changes in the horizontal direction of the projector. Patterns can similarly be defined in the vertical direction. The intensities observed by the camera is

$$\mathbf{I}_{c,n}(u_c, v_c) = \mathbf{A}_c(u_c, v_c) \sin\left(2\pi\left(f\frac{u_p}{N_p} - \frac{n}{N}\right)\right) + \mathbf{B}_c(u_c, v_c), \tag{3.5}$$

where $\frac{u_p}{N_p}$ is the corresponding normalized projector coordinate for the camera pixel in question, $\mathbf{A}_c$ is the intensity of the projector light reflected into the camera, and $\mathbf{B}_c$ is the intensity of light corresponding to other light sources in the scene. With these, we can separate the light in the scene into direct and global light, which is described in Section 3.2.2. Both $\mathbf{A}_c$ and $\mathbf{B}_c$ as well as the phase, or projector column, can easily be computed using the discrete Fourier transform.

With phase-shifting patterns, a scene can be encoded using only three shifts. Increasing the number of shifts, or patterns, increases the precision at which the terms above can be determined. In practice, different sequences with different frequencies are often used.

3.2.1.1 Phase Unwrapping

The phases that we estimate are wrapped. That means that we only know them modulo 2π [JB94]. This causes ambiguities when determining correspondences. Unwrapping the phases, and thereby resolving the ambiguities, is therefore necessary. In principle this can be achieved by adding a multiple of 2π when a sharp shift occurs in the phases along the coding axis. Due to several factors, mainly noise, this will usually not give a good solution.

One way of unwrapping the phase values is by utilizing that we typically project more than one sequence of patterns. By choosing the frequencies of the sequences carefully, we can find the unwrapped phase with the heterodyne principle [RRT97]. If two phases with wavelength λ_1 and λ_2 are subtracted, the result is a beat phase function with wavelength λ_b given by

$$\lambda_b = \frac{\lambda_1\lambda_2}{\lambda_1 - \lambda_2}. \tag{3.6}$$

This is illustrated in Figure 3.2. By choosing λ_1 and λ_2 such that $\lambda_b = 1$ the result is a phase function which spans the entire projector. This is achieved when the frequencies are chosen such that $f_2 = f_1 + 1$. The beat function can be used to determine the multiples of 2π that we need to add to the original unwrapped phases in order to unwrap them.

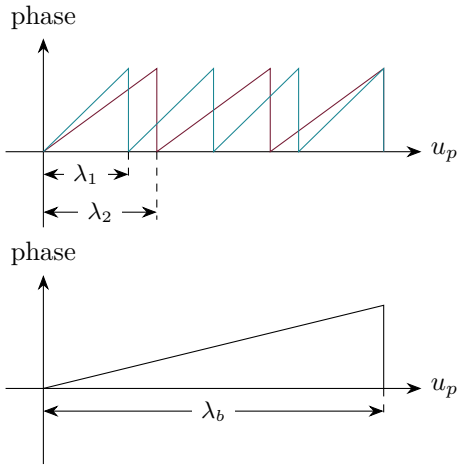


Figure 3.2: Figured adapted from [RRT97], illustrating how a beat function with wavelength λ_b can be determined by subtracting two phase functions with wavelength λ_1 and λ_2.

3.2.2 Direct and Global Illumination

When a camera takes a picture of a scene, it measures the irradiance in each pixel. In other terms, the camera measures the amount of light that bounces on the objects in the scene and hits the camera sensor. In structured light imaging, the light can be separated into two parts. The first part is the projector light reflected by the objects, and the second part is everything else. This separation is also referred to as direct and global illumination. The latter contains ambient light and light due to effects such as subsurface scatting and interreflections. Global illumination may lead to unwanted reconstruction artifacts in the reconstruction, and therefore we often want to separate the direct light and the global light [CSL08].

When patterns of the form (3.5) are observed, we automatically gain a way of separating direct light from global light [Nay+06], given by

$$\mathbf{L}_{\text{direct}} = 2\mathbf{A}_c, \qquad \mathbf{L}_{\text{global}} = \mathbf{B}_c - \mathbf{A}_c.$$

In Contribution C we utilize the global and direct light estimates to create realistic looking renderings, trying to replicate images captured by a camera. We do this by using the estimated $\mathbf{A}_c$ and $\mathbf{B}_c$ to render images on the form (3.5).

3.3 Surface Reconstruction

After a point cloud has been obtained, the second step is to reconstruct a continuous surface. Reconstructing a physical object's surface has been an active topic since the earliest days of computer vision in the early 1970s. Back then, reconstructing 3D structure from images was merely seen as a step towards full scene understanding and artificial intelligence. Since then, however, recovering digital versions of objects has become valuable in, e.g., many engineering applications or for cultural heritage preservation.

3.3.1 Surface representations

We can represent surfaces in several different ways. When choosing a representation to use for a specific application, it is vital to consider different representations' strengths and weaknesses. Suppose the application, e.g., is to measure geometric distances on the object. Then it is sufficient to represent the surface with point samples from the object boundary since it is fast and easy to compute distances between points. Point samples, however, do not contain any neighborhood information about which points are close to each other geodesically. This property may be required for other applications. Most applications require a representation that specifies the entire surface continuously. Surfaces representations are commonly split into either explicit representations or implicit representations. Figure 3.3 shows a 2D example of an explicit and an implicit representation of the same curve.

3.3.1.1 Explicit surfaces

Explicit representations define, explicitly, the boundary of an object. One way of doing this is by parameterizing the surface by a function $f : X \to \mathbb{R}^3$, where $X \subset \mathbb{R}^2$.

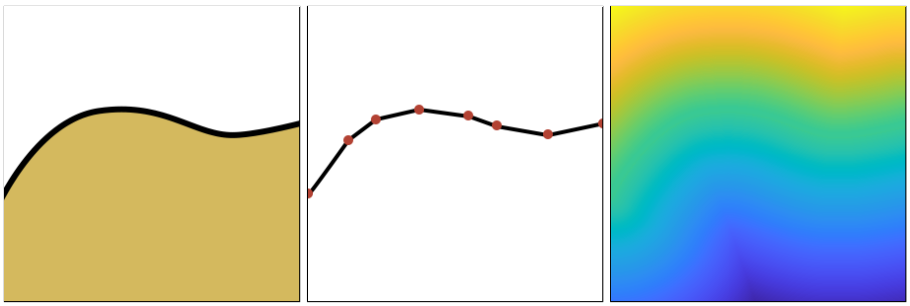


Figure 3.3: Left: A curve representing the boundary of an object with the inside in gold. Middle: Explicit representation using piecewise linear segments. Right: Implicit representation with deep blue meaning large negative values and bright yellow meaning high positive values.

The surface $\mathcal{S}$ is then given by $\mathcal{S} = f(X)$.

Most freeform surfaces are impractical or even impossible to represent exactly, and therefore various approximations are used.

Spline surfaces Producing a single parameterization to represent a complex shape is not an easy task. Instead, the surface is often represented by a set of surface patches, with each patch having its own parametrization. A common way of parameterizing the patches is by defining control points $\mathbf{c}_i \in \mathbb{R}^3$ and a set of basis functions $N_i(u, v)$ and combining them in the form

$$f(u, v) = \sum_{i=1}^{n} \mathbf{c}_i N_i(u, v). \tag{3.7}$$

The most typically used basis functions are B-splines, which are piecewise polynomials with minimal support. The minimal support implies that each control point $\mathbf{c}_i$ only has local influence on the geometry. Furthermore, B-splines has the property that $\sum_{i=1}^{n} N_i(u, v) = 1$, which means that the surface is completely contained in the convex hull of the control points. Together with the local influence this means that controlling the geometry of the surface can be done by simply moving the control points.

To represent a complex, continuous surface, we need a large number of pathces linked together smoothly. Ensuring smoothness in the transition between the patches is however, not trivial.

Subdivision surfaces Subdivision surfaces are akin to spline surfaces in that a set of control points defines them. However, instead of combining these with a set of basis functions, they form a control mesh that is repeatedly subdivided. In contrast to spline surfaces, we can represent objects of arbitrary topology without the need for linking together patches. They cannot, however, be used to represent objects with arbitrary geometry.

Triangle meshes Triangle meshes are a special case of polygonal meshes. Polygonal meshes represent the surface by a set vertices $\mathcal{V}$, a set of edges $\mathcal{E}$, and a set of faces $\mathcal{F}$. Polygonal meshes are thus not a parameterized surface, albeit still an explicit surface representation. In a triangle mesh, each face $f \in \mathcal{F}$ connects exactly three vertices making each face a triangle, hence the name. Since triangles are planar, triangle meshes are piecewise linear. The approximation error when using them to represent smooth surfaces is $O(h^2)$, where h is the maximum edge length [Bot+06]. This implies that the error from using a triangle mesh representation is inversely proportional to the number of faces, which means that we can represent any smooth surface as precisely as needed by simply adjusting the number of faces.

Moreover, triangle meshes allow for arbitrary topology and for changes to the surface geometry by merely moving the vertices. Together these features render triangle meshes as a powerful and versatile means of representing surfaces.

3.3.1.2 Implicit surfaces

Another way of specifying a surface is to implicitly represent it as the kernel of some function $F : \mathbb{R}^3 \to \mathbb{R}$, or in other terms, the surface is given by the level set $S = \{(x, y, z) \in \mathbb{R}^3 \mid F(x, y, z) = 0\}$. Since F is defined over an entire volume, this is also referred to as a volumetric representation. This representation is very often used in medical imaging for various modalities such as CT or MRI scans. Normally F is defined to be negative inside an object and positive outside. Implicit representations have several advantages. Some of the most important are:

- With one simple function evaluation, we can determine if a point is inside or outside an object.
- Boolean operations on implicit surfaces amount to simply looking at the sign of F. Using Boolean operations, it is easy and efficient to make changes to the topology of the surface.

Contrary to explicit surfaces, such as a triangle mesh, however, we cannot easily change the geometry of an implicit surface.

Signed distance function The most widely used implicit surface representation is the signed distance function. This specifies the distance to the object surface, with negative values corresponding to points inside the object and positive values to points outside. This makes distance calculations to the surface trivial.

3.3.2 Surface reconstruction from point cloud

Throughout the years, a wealth of methods for reconstructing surfaces from point clouds have been proposed. For an overview of many of these methods we refer the reader to Berger et al. [Ber+17]. In general, there are two approaches for reconstructing a surface from a point cloud. A surface can either be fitted to the points, or the points can be interpolated. To make the first approach tractable, certain prior assumptions must be made, e.g., smoothness of the resulting surface. Most methods using this approach are volumetric, where the surface is implicitly defined as a level set of a function defined in a volume.

In the second approach, the point clouds are assumed to be noise free, and reconstructing a surface is treated as a combinatorial problem, where a neighborhood structure needs to be imposed on the points in the point cloud.

Another class of methods for surface reconstruction, are methods that bypass the intermediate point cloud representation, and reconstructs a surface directly from images. The distinction between the two types of methods is visualized in Figure 3.4.

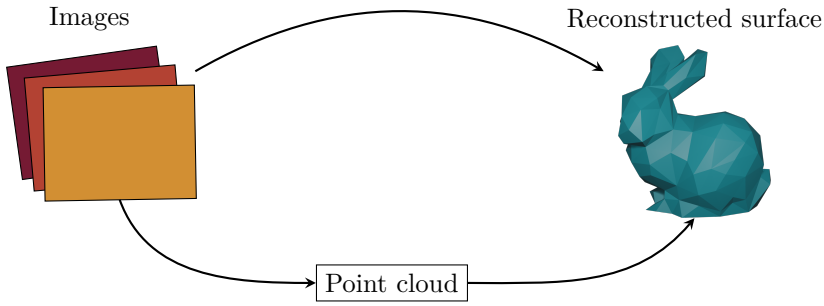


Figure 3.4: Surface reconstruction from images is most often are two-step approach, where an intermediate point cloud representation is first determined, followed by a surface reconstruction step (route below). Other methods use a one-step approach where the surface is reconstructed directly from the images (route above).

3.3.2.1 Volumetric Surface Reconstruction

Volumetric methods for surface reconstruction represent surfaces, as the name implies, in a volume. The surface is implicitly defined as a level set of a function, which is often defined on a regular grid of voxels. An early and defining method of this kind for reconstructing surfaces of arbitrary geometry is that of Hoppe et al. [Hop+92]. In this work, a signed distance field is computed as the distance to locally fitted planes. The zero-level set of the distance field defines the surface.

The arguably most well-known method is Poisson Reconstruction [KBH06]. This method estimates an indicator function, which is one inside the surface and zero outside the surface. Normals are computed at each point in the point cloud, and the oriented points are treated as samples of the gradient field of the indicator function. The problem then amounts to finding an indicator function whose gradient field best fits these samples. Solving this problem in the least-squares sense is formulated as a Poisson equation, hence the name. Screened Poisson Reconstruction is an extension of this work, where interpolation constraints are incorporated for each point in the point cloud, penalizing the indicator function deviating from zero at these points [KH13]. In Contribution C we use Screened Poisson Reconstruction to create starting guesses for a minimization problem.

A variety of other methods reconstruct surfaces using an implicit function. These include determining an indicator function using a wavelet basis [MPS08], determining a signed distance field and regularizing it using a Markov Random Field [PBL09], computing a signed distance field using a moving least squares approach [SOS04], or implicitly defining the surface as the decision boundary of a learned classifier given by a neural network [Mes+19].

If an explicit representation, e.g., a triangle mesh, is required, this can be extracted using isosurface polygonization algorithms such as Marching Cubes [LC87].

3.3.2.2 Combinatorial Surface Reconstruction

Surface reconstruction from point clouds can also be considered a combinatorial problem, where the original points in the point cloud, or a subset of them, are interpolated. Usually, a triangle mesh is created from this set of points [CG06]. This contrasts with the surface fitting methods described in the previous section, where the resulting surface does not necessarily pass through the original points of the point cloud. Typically, combinatorial methods are based on extracting the surface as a subset of triangles from a Delaunay triangulation [Boi84; EM94; ABK98].

3.3.3 Surface reconstruction directly from images

While most surface reconstruction algorithms take a point cloud of samples from the object boundary as input, some algorithms attempt to recreate the surface directly from images, thereby avoiding the process of producing a point cloud from the images.

Kutulakos and Seitz [KS00] introduces Space Carving. This algorithm computes the surface by considering the set of all shapes consistent with the input images and determining a special shape of this set, called the photo hull, which contains all of the other shapes in the set within its volume.

Some methods reconstruct the surface by deforming a surface representation such that it is consistent with the input images. One such method is the one presented by Zhang and Seitz [ZS00], and improvements to this method [IS02; IS03; YXA04; YXA07].

More recent methods are based on formulations of differentiable rendering, through either raytracing [LHJ19], or smoothed rasterization [Liu+19].

In Contribution C, we present a method in this category of reconstructing the surface directly from images. Our method, however, is specifically designed for images acquired using structured light.

CHAPTER 4
Contributions

In this chapter, we present three contributions. In Contribution A, we present a method for video frame interpolation, which can be used to create slow-motion videos from regular ones. In Contribution B we create illusory motion of physical objects by projecting light patterns onto them. Making objects appear to be moving can be used to attract visual attention to the object. In Contribution C, we present a method for mesh-based surface reconstruction from structured light images, which can be used to recreate accurate digital 3D versions of physical objects.

4.1 Synthesizing spatiotemporal viewpoints

Common to Contributions A to C is that they all deal with synthesizing images from different viewpoints. The goal in Contribution A is to synthesize novel viewpoints, mainly temporally but to some extent also spatially. In Contribution B we synthesize novel viewpoints spatially as a tool to simulate motion of physical objects, and in Contribution C we recreate the viewpoints of captured images by simulating the entire structured light imaging process and use it to reconstruct surfaces.

4.1.1 Synthesizing novel viewpoints in time from estimated motion

In Contribution A we represent a method for video frame interpolation. Video frame interpolation is mainly view synthesis in the temporal domain, and to a lesser extent in the spatial domain. To achieve frame interpolations of high visual quality, it is important to retain naturally occuring motion in the images, as simply interpolating based on image intensities rarely will give realistic results. An example of this kind of frame blending can be seen in Figure 4.1a.

Our method synthesizes a novel frame halfway between two input frames in time. To preserve the scene's natural movement, we explicitly estimate the motion and use the estimate to create the middle frame. We do this by computing weighted motion fields, akin to optical flow, from the unknown middle frame to the two input frames. We use the weighted motion fields to linearly combine pixel values from the input frames to obtain the middle frame. Determining motion from an unknown frame to two known frames is a complicated visual task, which requires translating pixel values to scene understanding, such as how objects move relative to the camera and how they

interact with light in the scene. Convolutional neural networks have been shown to excel in many complicated visual tasks and thus seems natural to utilize to estimate the motion fields.

Optical flow is one of the most used ways of describing motion in images. As stated, the motion fields that we determine are closely related to optical flow. We also determine a vector for each pixel in the unknown frame describing the motion of that point. In optical flow, the brightness constancy constraint is enforced to reduce the set of equations' underdetermination. We choose not to employ this constraint for the motion model to be more general and to allow the scene to become darker or brighter. In practice, we do this by also predicting weights with the motion field. By relaxing the brightness constraint, our motion model becomes more general but also more underdetermined, which complicates the process further. This also supports the use of a convolutional neural network to learn the motion.

We have chosen not to train the neural network explicitly to produce accurate motion fields. Although a strong motion model is needed to retain the videos' natural motion, the main goal is to produce interpolated images of high visual quality. Therefore, the network is instead trained to produce motion fields that are good for synthesizing an accurate unknown frame. To do this, the network is trained from triplets of images. The two outer frames are input and, the middle frame is used for evaluating the loss function. As a consequence of this formulation, there is no need for labeled data. The video sequences, from which the network is trained self-supervises the training.

The results presented in Contribution A are better than or comparable to the publically available state-of-the-art algorithms at the time of publication. Frame interpolation is a very active research topic, though, and many approaches have been presented since, most notably Softmax Splatting [NL20].

(a) Overlaid input frames (b) Ground truth (c) Ours without CFT (d) Ours

Figure 4.1: Representative example of how the cyclic fine-tuning improves the interpolated frame. It can be seen that the small misalignment of the tire and yellow "41" is corrected by the cyclic fine-tuning. Figure from Contribution A

4.1.1.1 Cyclic fine-tuning

The concept of cyclic fine-tuning is presented in Contribution A and is a novel approach to fine-tuning a convolutional neural network at inference time. It uses the fact that the output modality is the same as the input modality, and the output from our model can therefore also be used as input. By taking advantage of this fact, we can fine-tune the network for each specific frame that we want to interpolate. The process shown in Figure 4.2. The predicted frame $\hat{\mathbf{I}}_{1.5}$, can be used to predict two new frames $\hat{\mathbf{I}}_1$ and $\hat{\mathbf{I}}_2$. Notice that we have ground truth images $\mathbf{I}_1$ and $\mathbf{I}_2$ corresponding to these. That means that we can compare the synthesized images and the real images and quantify how well they resemble each other. The quantified resemblance is a measure of how useful the predicted frame $\hat{\mathbf{I}}_{1.5}$ is for predicting. This can be considered a proxy of how well the frame has been synthesized. In other words, if the frames that are predicted based on this frame look nothing like the real images, then this frame is probably not very well synthesized, and contrary if the predicted frames based on this frame look a lot like the ground truth images, then the predicted frame $\hat{\mathbf{I}}_{1.5}$ is likely of high quality. From Figures 4.1c and 4.1d it can be seen that cyclic fine-tuning greatly improves the visual quality.

The concept is not limited to our video frame interpolation network but can be used with any video frame interpolation network or even neural networks for other tasks where the input and output have the same modality.

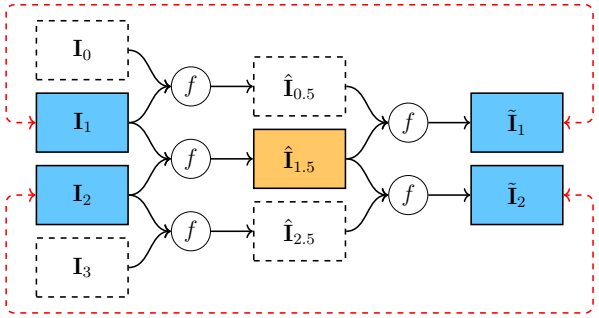


Figure 4.2: Diagram illustrating the cyclic fine-tuning process when predicting frame $\hat{\mathbf{I}}_{1.5}$. The model is first applied in a pairwise manner on the four input frames $\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2$, and $\mathbf{I}_3$, then on the results $\hat{\mathbf{I}}_{0.5}$,$\hat{\mathbf{I}}_{1.5}$, and $\hat{\mathbf{I}}_{2.5}$. The results of the second iteration, $\tilde{\mathbf{I}}_1$ and $\tilde{\mathbf{I}}_2$, are then compared with the input frames and the weights of the network are updated. This process optimizes our model specifically to be good at interpolating frame $\hat{\mathbf{I}}_{1.5}$.

4.1.2 Synthesizing novel viewpoints in space from estimated structure

In order to simulate motion of physical objects in Contribution B, a key component is the ability to synthesize novel viewpoints. Specifically, we synthesize an image from the projector's point of view. This section describes how this is done in greater detail than in the original paper, where many details were left out due to length restrictions.

Synthesizing an image from the perspective of the projector is done through a number of steps. These are:

(i) Calibrate a structured light setup with a projector and two cameras using the methods described in Sections 2.1.2.1 to 2.1.2.3.

(ii) Compute colored point clouds from each camera.

(iii) Project the point clouds to the projector image plane.

(iv) Combine the projector images and clean the result to obtain a synthesized image.

Computing colored point clouds We compute colored point clouds using a structured light setup. By colored, we mean that we associate the color in the camera image with the 3D point. The two cameras in the setup are placed on either side of the projector. This placement allows for better coverage of the object in focus, compared with only using one camera. Using phase-shifting structured light sequences outlined in Section 3.2, we create two separate 3D scans - one for each camera-projector pair.

Projecting point clouds to projector image plane If $\mathbf{p} = [X, Y, Z, 1]^\mathsf{T}$ is a point in the point cloud corresponding to the first camera in homogeneous coordinates, and $\mathbf{P}_p$ and $\mathbf{P}_c^1$ are the projection matrices of the projector and the first camera respectively then the projections

$$s_p \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \mathbf{P}_p \mathbf{p} \qquad \text{and} \qquad s_c \begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} = \mathbf{P}_c^1 \mathbf{p}, \tag{4.1}$$

gives us the projector and camera pixel coordinates (u_p, v_p) and (u_c, v_c). Note that due to the construction (u_c, v_c) are integer coordinates while (u_p, v_p) in general will not be. If we denote the image from the first camera by $\mathbf{I}_c^1$, then the unknown projector image $\mathbf{I}_p^1$ is constructed by defining by

$$\mathbf{I}_p^1 \left(\lfloor u_p \rceil, \lfloor v_p \rceil \right) := \mathbf{I}_c^1 \left(u_c, v_c \right), \tag{4.2}$$

with $\lfloor \cdot \rceil$ is the rounding-operator. Using the same process, we can also create a projector image $\mathbf{I}_p^1$ with the point cloud corresponding to the second camera. In many cases, several points will hit the same pixel in the projector image. This is handled

by defining the value in the projector image as the mean of the camera intensities of these points. Some of the points in the 3D scans will have miscalculated depths due to errors in the phase estimation. Due to this, they intersect the wrong pixels in the projector image, resulting in individual pixels with wrong values. For this reason, the images are filtered with a 3×3 median filter. Examples of the images created like this are shown left in Figures 4.3b and 4.3d.

Combining the projector images After the two images are created, one corresponding to each camera, they are combined by a weighted average. The two images may differ slightly in brightness. Therefore a brightness scale factor k is computed by

$$k = \text{median} \left\{ \frac{\mathbf{I}_p^1}{\mathbf{I}_p^2} \right\}, \tag{4.3}$$

where the division is pixel-wise, and the median is taken over all pixels. We use this scale factor as a measure of how the two images differ in brightness and use it to create the combined projector image

$$\mathbf{I}_p = \frac{1}{2} \left(\mathbf{I}_p^1 + k\mathbf{I}_p^2 \right) \tag{4.4}$$

The result from this is seen in Figure 4.3e, where the red pixels indicate missing values in the projector image.

Since we are not interested in making the whole scene, appear to be moving, but only a specific object, we create a segmentation mask of the object. We get this by thresholding the depth map corresponding to the image followed by morphological operations. An example of a mask created with this procedure is shown in Figure 4.3f.

Finally, the image's missing values are inpainted using an algorithm based on biharmonic functions, which is available in the widely used image processing library scikit-image [Wal+14; DH18]. Figure 4.3g shows the final result.

4.1.2.1 Synthesizing with uncalibrated setup

Even in an uncalibrated setup, we can still synthesize an image from the projector's perspective. When we use phase-shifting structured light sequences, the coding axis is typically in the projector's horizontal direction. This gives us a mapping from columns in the camera to columns in the projector. If we also include a sequence in the vertical direction, we similarly obtain a mapping from camera rows to projector rows. If we combine the two, we get a mapping f from camera pixel coordinates to projector coordinates

$$(u_p, v_p) = f(u_c, v_c). \tag{4.5}$$

Using this, we can construct the projector image described in the previous section according to (4.2) and on.

(a) Image captured by left camera.

(b) Synthesized projector image using left camera.

(c) Image captured by right camera.

(d) Synthesized projector image using right camera.

(e) Combined projector images.

(f) Mask of object created by thresholding depth.

(g) Final result with object object segmented with mask and missing values inpainted.

Figure 4.3: Creating an image from the projector's point of view.

Although this approach has a significant advantage that we do not need to calibrate the system, it has the drawback of giving us no depth information about the object and the scene. First of all, this makes the task of segmenting the object from the rest of the scene much harder as we can only only rely on 2D information. Secondly, without depth information we cannot adapt 2D 'Motion Without Movement' filters to the 3D geometry of the object.

4.2 Simulating motion of physical objects

Making static objects appear to move without touching them may, to some, sound like an impossible task. This task is, nevertheless, what Contribution B attempts to solve. As mentioned in Section 2.2.2, the human visual perception system may perceive motion where there is no motion. This can be due to, e.g., changes in luminance or color. This effect has been known for many years, and optical illusions utilizing this are numerous. In Contribution B, we adapt an illusion aptly called 'Motion Without Movement' [FAH91] to a projector-camera setup to make a static physical object appear to be moving through continuous changes to the luminance of the object. The steps taken to achieve the effect are as follows:

(i) Synthesize image from projector's point of view according to Section 4.1.2.

(ii) Use the filters from [FAH91] on the synthesized image.

(iii) Project the filter response back onto the object.

For step (ii) the filters can either be used as described in the original paper or be adapted to the geometry of the object. In the next section we present a method for doing the latter.

4.2.1 Geometry Aware Filtering

The filters described by Freeman, Adelson, and Heeger [FAH91] are 2D image filters, and thus the filtering is done in image space, without accounting for the geometry of the object. Therefore, the projected pattern is, to a great extent, view-dependent, and the perceived effect may decrease as we deviate from the viewpoint of the projector.

When we synthesize the projector image as described in Section 4.1.2, we get an intensity in all pixels and the 3D position for each pixel. With this information, we can compute a normal vector at each point using the 3D position of neighboring pixels. The 3D position of the pixel and the normal vector defines the tangent plane. By filtering in the tangent plane, instead of in the image plane, we approximate filtering along the surface, under the assumption that the object is locally planar. We determine a homography between the image plane and the tangent plane to do the filtering in the tangent plane. A few considerations are necessary when determining

the homography. The first is that the area in the tangent plane that we apply the filters to should be constant for all points. The second is that slightly changing the pixel position should also only alter the filter response slightly unless there is a large change in the object's geometry. We satisfy both of these by choosing an orthonormal basis that changes smoothly over the surface [Fri12]. Figure 4.4 shows an example of which image pixels are used when filtering in the image plane and when filtering in the tangent plane.

4.2.2 Simulating physical motion by projection of filter response

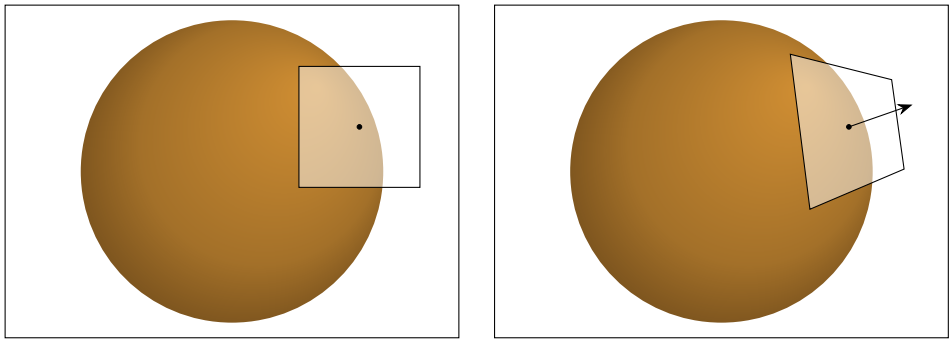
To create the effect of illusory motion, the filter responses from step (ii) in Section 4.2 are projected onto the object. This creates a visual effect that the object is moving sideways or rotating around itself.

From demonstrating the setup, it is clear that the effect is indeed perceived as motion. It is also clear that the effect is very subtle, as was initially intended. Due to the subtlety of the effect, some things will ruin the effect of motion. If the projector light is too bright, the illusion of motion can be ruined, as the luminance changes are too large. An inaccurate calibration of projector and cameras also ruins the effect, as the synthesized projector image will be blurry.

Simulating physical motion can be used to attract people's attention. Our attention is captured when there are sudden changes, such as motion, in our environment. Creating a subtle motion effect can catch people's attention without the effect appearing intrusive. However, this was not explored further, so it remains to be determined if the effect can, in fact, be used to attract and guide the attention of humans.

4.3 Estimating geometry by simulating imaging process

In Contribution C, we represent a novel method for reconstructing an object's surface from structured light image sequences. The method is based on simulating the image acquisition process, using a parameterized surface, and comparing the rendered images with recorded images. The surface parameters are then iteratively changed to make the rendered images look like the recorded images. We frame this as an optimization problem. This approach differs significantly from how surface reconstruction typically is done, where multiple steps are needed to recover the object's surface. Figure 4.5 shows a comparison between the traditional steps and the steps in our approach. We show that our method can reconstruct various objects with very high accuracy. In particular, our method is very good at reconstructing sharp features such as edges or corners. We also show that our methods are very robust with regard to image noise. An example of a reconstruction is shown in Figure 4.6.



(a) Filtering in image plane.

(b) Filtering in tangent plane.

Figure 4.4: Filtering with 'Motion Without Movement' filters can either be done in the image plane or the tangent plane. The figures above show which image pixels both of these approaches correspond to.

As our method is based on iteratively updating an object's geometry to fit captured images best, we choose to parameterize the surface using a triangle mesh. The geometry of a triangle mesh can be changed by only moving the vertices. We can, however, not easily change the topology of a triangle mesh. Therefore, our optimization's initial surface needs to have the same topology as the object whose surface we aim to reconstruct. Using a triangle mesh also has the advantage, that computing the derivatives of the parameters (the vertex positions) with respect to the pixel differences is relatively simple. This is because any pixel only influences the derivatives of three vertices. All pixels, stemming from the parameterized surface's projection into the cameras, belong to exactly one face each. The three vertices whose derivatives are affected by a pixel are precisely those that define the face to which the pixel belongs.

4.3.1 Advantages of simulating imaging process

There are several advantages to replacing the steps in the traditional reconstruction pipeline with a simulated imaging process. The first advantage is traceability from the input images to the final reconstructed surface. With each step in the typical pipeline, we throw the information from the previous step away. Moreover, with each step, additional noise will be added and propagated to the final result. The complex error propagation and lack of information flow severely hinder the evaluation of how well the reconstructed surface corresponds to the original image data. In contrast, in our approach, we can directly compare the output, in the form of rendered images, with the captured images and compute how the differences affect the reconstructed surface.

Another advantage of simulating the imaging process and formulating surface reconstruction as an optimization problem is that we can simultaneously use all recorded image data. Usually, a point cloud is created for each camera position separately, and

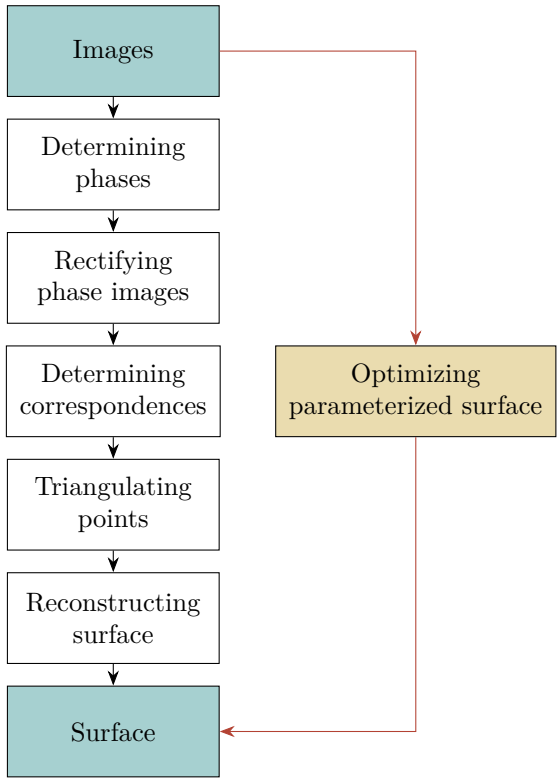


Figure 4.5: The steps involved in estimate a surface from structured light images. The black route on the left shows the steps needed in the typical approach. The red route on the right shows the step needed in the approach in Contribution C.

Figure 4.6: The method from Contribution C used to reconstruct the Stanford Bunny starting from a sphere. From left to right: Initial mesh (sphere), after 50 iterations, after 750 iterations, and the converged result. The final reconstruction has 13 780 vertices and a symmetric volume difference error of 0.30%, when compared to the ground truth. Figures is from Contribution C.

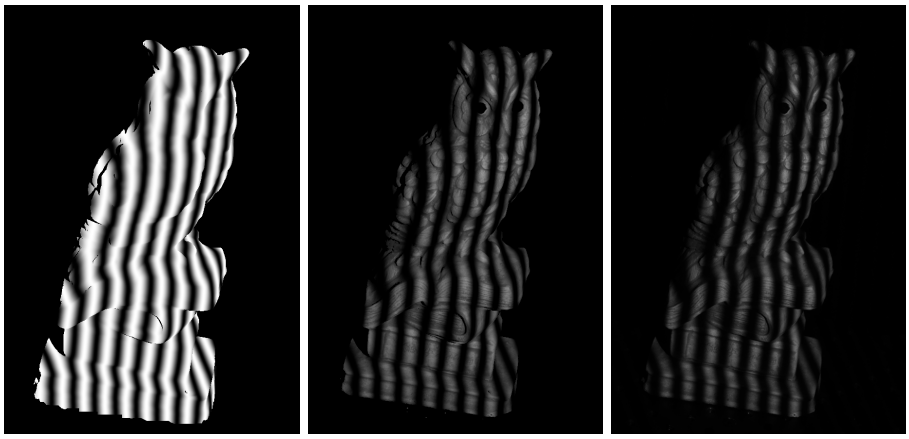
these sub-scans are then subsequently registered to each other using the calibrated camera positions or with a registration step using, e.g., the Iterative Closest Point algorithm [BM92; AHB87], or sometimes both. There is no need for a registration step in our approach since all images are handled simultaneously and included in the optimization. Note that for the results in Contribution C, we know the exact camera position, and the same knowledge would trivialize merging sub-scans in the traditional pipeline. In real scenarios, we only know the camera positions to some degree of accuracy. For that reason, both the relative camera positions and a registration algorithm are used in succession to register sub-scans. If the individual point clouds are very noisy or have little overlap, the accuracy of algorithmic registrations will be negatively affected. With our method, there is no need for a separate registration step. As we optimize using all image data simultaneously, we can incorporate the camera poses as the optimization parameters. By including the camera poses as parameters, the registration is part of the optimization. In this way, the registration is done to fit the original image data best and is unaffected by missing overlap between point clouds or images.

Our method is able to handle many camera poses and structured light patterns simultaneously. The results presented in Contribution C uses 60 simultaneous camera positions, each with 24 structured light patterns at 1920×1080 resolution, and can easily handle even more. That is approximately three billion squared error terms that are summed in each optimization step. With so many squared error terms, there is a lot of redundancy, meaning that our method is particularly good, e.g., in situations with low signal-to-noise ratios. This is evident from experiments with varying camera noise.

As our method is based on simulating the image acquisition process, there are cases to which it can easily be extended and possibly have an advantage over the traditional approach. One such example is surface reconstruction of specular objects. Very often, high-dynamic-range imaging is used when capturing images of objects with specularities. A good exposure level can then be determined for each pixel from which the structured light signal can be decoded. Our method is easily extended to a high-dynamic-range setting by simply rendering images corresponding to all the exposure levels in the high-dynamic-range stack. Thus, our method can use all the available information in the high-dynamic-range stack instead of simply choosing one exposure level at each pixel. Overexposed pixels will not influence the result as the renderings are clipped to the max value corresponding to bright white, which means that the squared error will be zero in those pixels.

The rendering technique used in Contribution C is rather crude, and it may be surprising how such good results can be achieved with relatively simple renderings. The method can indeed be extended using more sophisticated rendering techniques. This is, however, not necessary. The trick is that we estimate the direct and global light from the captured images and use these for our renderings. This yields very convincing renderings, which is apparent from Figure 4.7. In Figure 4.7a a virtual

projection of a structured light pattern is shown. Figure 4.7b shows the result when we compensate for the observed radiometry with the estimated direct and global light. There is an excellent likeness with the actual image, shown in Figure 4.7c. The estimated direct and global parameters may be prone to errors, especially when images are overexposed or underexposed. As in the case of the camera poses, the parameters may easily be incorporated into the optimization problem to achieve better estimates. In that case, the estimated values serve as excellent starting guesses.



(a) Rendering without radiometric compensation.

(b) Rendering with radiometric compensation.

(c) Captured image.

Figure 4.7: Illustration of how estimated global and direct light can be used to create more realistic renderings.

CHAPTER 5

Conclusion

In this thesis, we have presented research in the span of structure, motion, estimation, and simulation. We have explored synergies between these terms. We have both estimated structure and motion, and we have simulated structure and motion. In Contributions A and B we estimated to simulate, and in Contribution C we simulated to estimate.

Creating better estimations using simulations and creating better simulations using estimations has been the goal of this thesis. This is a very broad goal, and more specifically, the thesis has had three objectives.

The first objective was to estimate motion in image sequences and to use this motion to interpolate temporally new images. This objective was the focus of Contribution A. In this contribution, we introduced the cyclic fine-tuning concept where a neural network is improved at inference time, specifically for the prediction it is making. We utilize that the model's output also can be used as input to re-predict the original input and that the usefulness as an input is a proxy measure of how well it has been predicted in the first place. This concept is not limited to our model and can be used for other methods where the input and output modalities are the same. In Contribution A, a convolutional neural network was trained in a self-supervised manner to predict motion, and subsequently, the motion was used to create new images in-between the original frames in a movie. To obtain interpolated frames of good visual quality, we needed to retain the natural motion in the sequence, and therefore it was essential with a model that could estimate the motion in the sequence. Using this model and cyclic fine-tuning, we achieved better results than the publically available state-of-the-art methods at the time of publishing.

The second objective was to use estimated geometry to simulate physical motion, or in other terms, create the illusion that a real-world physical object is moving. This objective is the topic of Contribution B. With this contribution, we show that we can bring a 2D optical illusion to the physical realm and use it to simulate motion in an augmented reality setting, within the limits of what is achievable with a projector. This work indicates that it is feasible to use a projector to create immersive augmented reality experiences when we adapt the projected light to the scene's geometry.

We achieved the motion effect by projecting specific light patterns onto the object on which we wanted to create the illusion. We created the light patterns by adapting a

2D image optical illusion where filters are used on the image, and the filter responses are interpolated. To adapt this effect to a physical 3D setting, we created an image of the object from the projector's point of view. This was done with a structured light setup, where the geometry of the object was estimated and projected back into the projector. The filters were used on this projector image, and the filter responses were projected back onto the object to create the illusion of motion.

Estimating the geometry was not strictly necessary to create a projector image. However, estimating it enables us to adapt the 2D filters to the 3D setting. This adaption makes the effect less view-dependent and thus makes the simulation more convincing. Potential future work includes creating other motion effects and determining if the effect can be a useful attention cue.

The third and final objective was to estimate geometry by simulating the imaging process. Contribution C dealt with this objective. In contrast to typical surface reconstruction approaches, our formulation is an end-to-end model going directly from input images to the output surface. With this formulation, we skip the many steps usually needed to reconstruct surfaces from structured light images and gain full traceability from the input images to the resulting final surface. The results showed that our method produces very accurate reconstructions, and is especially good at reconstructing sharp features such as edges or corners.

Instead of treating surface reconstruction as a backward problem, we reformulated it as a forward problem where we simulated the imaging process and iterated the simulation with updated parameters until the rendered images matched the captured images. The parameters of the simulation were the vertex position of a triangle mesh. When the process converged, the final set of parameters gave us the resulting surface mesh.

By formulating the problem in this way, we gain several advantages over the traditional approach. As mentioned, we skip the many steps usually needed in surface reconstruction to allow for better information and noise flow. The formulation and implementation also allow us to use all input image data simultaneously. This alleviates the issue of having to combine several subscans. Future work includes using this method on real-world data.

To shortly summarize, we have met the three objectives set out at the beginning of this thesis.

Bibliography

[ABK98] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. "A new Voronoi-based surface reconstruction algorithm". In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998, pages 415–421.

[AHB87] K Somani Arun, Thomas S Huang, and Steven D Blostein. "Least-squares fitting of two 3-D point sets". In: *IEEE Transactions on pattern analysis and machine intelligence* 5 (1987), pages 698–700.

[Ber+17] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. "A survey of surface reconstruction from point clouds". In: *Computer Graphics Forum*. Volume 36. 1. Wiley Online Library. 2017, pages 301–329.

[BM92] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes". In: *Sensor fusion IV: control paradigms and data structures*. Volume 1611. International Society for Optics and Photonics. 1992, pages 586–606.

[Boi84] Jean-Daniel Boissonnat. "Geometric structures for three-dimensional shape representation". In: *ACM Transactions on Graphics (TOG)* 3.4 (1984), pages 266–286.

[Bot+06] Mario Botsch, Mark Pauly, Christian Rossl, Stephan Bischoff, and Leif Kobbelt. "Geometric Modeling Based on Triangle Meshes". In: *ACM SIGGRAPH 2006 Courses*. SIGGRAPH '06. Boston, Massachusetts: Association for Computing Machinery, 2006, 1–es. ISBN: 1595933646. DOI: 10.1145/1185657.1185839. URL: https://doi.org/10.1145/1185657.1185839.

[BP06] Michael Bach and Ch M Poloschek. "Optical illusions". In: *Adv Clin Neurosci Rehabil* 6.2 (2006), pages 20–21.

[Bro+19] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. "Unprocessing images for learned raw denoising". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pages 11036–11045.

[Bro66] Duane C Brown. "Decentering distortion of lenses". In: *Photogrammetric Engineering and Remote Sensing* (1966).

[BT16] Adrian Bulat and Georgios Tzimiropoulos. “Human pose estimation via convolutional part heatmap regression”. In: *European Conference on Computer Vision*. Springer. 2016, pages 717–732.

[Bul+20] Adrian Bulat, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. “Toward fast and accurate human pose estimation via soft-gated skip connections”. In: *arXiv preprint arXiv:2002.11098* (2020).

[CG06] Frédéric Cazals and Joachim Giesen. “Delaunay triangulation based surface reconstruction”. In: *Effective computational geometry for curves and surfaces*. Springer, 2006, pages 231–276.

[Che+09] Xiaobo Chen, Juntong Xi, Ye Jin, and Jin Sun. “Accurate calibration for a camera–projector measurement system based on structured light projection”. In: *Optics and Lasers in Engineering* 47.3-4 (2009), pages 310–319.

[Con19] A. E. Conrady. “Decentred Lens-Systems”. In: *Monthly Notices of the Royal Astronomical Society* 79.5 (March 1919), pages 384–390. ISSN: 0035-8711. DOI: 10.1093/mnras/79.5.384. eprint: https://academic.oup.com/mnras/article-pdf/79/5/384/18250798/mnras79-0384.pdf. URL: https://doi.org/10.1093/mnras/79.5.384.

[CSL08] Tongbo Chen, Hans-Peter Seidel, and Hendrik PA Lensch. “Modulated phase-shifting for 3D scanning”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pages 1–8.

[Dah+20] Lars B. Dahlin, Kristian R. Rix, Vedrana A. Dahl, Anders B. Dahl, Janus Nørtoft Jensen, Peter Cloetens, Alexandra Pacureanu, Simin Mohseni, Niels O.B. Thomsen, and Martin Bech. “Three-dimensional architecture of human diabetic peripheral nerves revealed by X-ray phase contrast holographic nanotomography”. In: *Scientific reports* 10.1 (2020), pages 1–8.

[DH18] Steven B Damelin and NS Hoang. “On surface completion and image inpainting by biharmonic functions: Numerical aspects”. In: *International Journal of Mathematics and Mathematical Sciences* 2018 (2018).

[Die04] F. Dierks. *Sensitivity and Image Quality of Digital Cameras*. Technical report. Basler AG, 2004.

[Ein+17] Gudmundur Einarsson, Janus Nørtoft Jensen, Rasmus R Paulsen, Hildur Einarsdottir, Bjarne K Ersbøll, Anders B Dahl, and Lars Bager Christensen. “Foreign Object Detection in Multispectral X-ray Images of Food Items Using Sparse Discriminant Analysis”. In: *Scandinavian Conference on Image Analysis*. Springer. 2017, pages 350–361.

[EM94] Herbert Edelsbrunner and Ernst P Mücke. “Three-dimensional alpha shapes”. In: *ACM Transactions on Graphics (TOG)* 13.1 (1994), pages 43–72.

[Eur16] European Machine Vision Association. “EMVA standard 1288, standard for characterization of image sensors and cameras”. In: *Release* 3.1 (2016), page 39.

[FAH91] William T Freeman, Edward H Adelson, and David J Heeger. “Motion without movement”. In: *ACM Siggraph Computer Graphics* 25.4 (1991), pages 27–30.

[Fri12] Jeppe Revall Frisvad. “Building an orthonormal basis from a 3d unit vector without normalization”. In: *Journal of Graphics Tools* 16.3 (2012), pages 151–159.

[Goo+16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Volume 1. 2. MIT press Cambridge, 2016.

[Guo+05] Hongwei Guo, Haitao He, Yingjie Yu, and Mingyi Chen. “Least-squares calibration method for fringe projection profilometry”. In: *Optical Engineering* 44.3 (2005), page 033603.

[Han+19] Morten Hannemose, Janus Nørtoft Jensen, Gudmundur Einarsson, Jakob Wilm, Anders Bjorholm Dahl, and Jeppe Revall Frisvad. “Video Frame Interpolation via Cyclic Fine-Tuning and Asymmetric Reverse Flow”. In: *Scandinavian Conference on Image Analysis*. Springer. 2019, pages 311–323.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pages 770–778.

[Hop+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. “Surface reconstruction from unorganized points”. In: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 1992, pages 71–78.

[HR95] I.P. Howard and B.J. Rogers. *Binocular Vision and Stereopsis*. Oxford psychology series. Oxford University Press, 1995. ISBN: 9780195084764.

[HS81] Berthold KP Horn and Brian G Schunck. “Determining optical flow”. In: *Techniques and Applications of Image Understanding*. Volume 281. International Society for Optics and Photonics. 1981, pages 319–331.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[IS02] John Isidoro and Stan Sclaroff. “Stochastic mesh-based multiview reconstruction”. In: *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*. IEEE. 2002, pages 568–577.

[IS03] John Isidoro and Stan Sclaroff. “Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints”. In: *null*. IEEE. 2003, page 1335.

[JB94] Thomas R Judge and PJ Bryanston-Cross. “A review of phase unwrapping techniques in fringe analysis”. In: *Optics and Lasers in Engineering* 21.4 (1994), pages 199–239.

[Jen+17] Janus Nørtoft Jensen, Rasmus Ahrenkiel Lyngby, Henrik Aanæs, Eyþór Rúnar Eiríksson, Jannik Boll Nielsen, Jakob Wilm, and David Bue Pedersen. “Photogrammetry for Repositioning in Additive Manufacturing”. In: *euspen and ASPE Special Interest Group Meeting: Additive Manufacturing*. 2017.

[Jen+19] Janus Nørtoft Jensen, Morten Hannemose, Jakob Wilm, Anders Bjorholm Dahl, Jeppe Revall Frisvad, and Serge Belongie. “Generating spatial attention cues via illusory motion”. In: *Third Workshop on Computer Vision for AR/VR*. 2019. URL: http://people.compute.dtu.dk/jnje/illusory-motion/.

[Jen+20] Janus Nørtoft Jensen, Morten Hannemose, Jakob Andreas Bærentzen, Jakob Wilm, Jeppe Revall Frisvad, and Anders Bjorholm Dahl. “Surface Reconstruction from Structured Light Images using Differentiable Rendering”. Submitted to: *Sensors*. 2020.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. “Poisson surface reconstruction”. In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Volume 7. 2006.

[KH13] Michael Kazhdan and Hugues Hoppe. “Screened poisson surface reconstruction”. In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), pages 1–13.

[KMK07] Makoto Kimura, Masaaki Mochimaru, and Takeo Kanade. “Projector calibration using arbitrary planes and calibrated camera”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pages 1–2.

[KS00] Kiriakos N Kutulakos and Steven M Seitz. “A theory of shape by space carving”. In: *International journal of computer vision* 38.3 (2000), pages 199–218.

[KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pages 84–90.

[LC87] William E Lorensen and Harvey E Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *ACM siggraph computer graphics* 21.4 (1987), pages 163–169.

[LHJ19] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. “Reparameterizing discontinuous integrands for differentiable rendering”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pages 1–14.

[Liu+19] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. "Soft rasterizer: A differentiable renderer for image-based 3d reasoning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pages 7708–7717.

[LK+81] Bruce D Lucas, Takeo Kanade, et al. "An iterative image registration technique with an application to stereo vision". In: (1981).

[Lyn+17] Rasmus Ahrenkiel Lyngby, Jakob Wilm, Eyþór Rúnar Eiríksson, Jannik Boll Nielsen, Janus Nørtoft Jensen, Henrik Aanæs, and David Bue Pedersen. "In-line 3D print failure detection using computer vision". In: *euspen and ASPE Special Interest Group Meeting: Additive Manufacturing*. 2017.

[Mes+19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pages 4460–4470.

[MPS08] Josiah Manson, Guergana Petrova, and Scott Schaefer. "Streaming surface reconstruction using wavelets". In: *Computer Graphics Forum*. Volume 27. 5. Wiley Online Library. 2008, pages 1411–1420.

[MT12] Daniel Moreno and Gabriel Taubin. "Simple, accurate, and robust projector-camera calibration". In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE. 2012, pages 464–471.

[Nay+06] Shree K Nayar, Gurunandan Krishnan, Michael D Grossberg, and Ramesh Raskar. "Fast separation of direct and global components of a scene using high frequency illumination". In: *ACM SIGGRAPH 2006 Papers*. 2006, pages 935–944.

[Nie+17] Jannik Boll Nielsen, Eyþór Rúnar Eiríksson, Rasmus Ahrenkiel Lyngby, Jakob Wilm, Janus Nørtoft Jensen, Henrik Aanæs, and David Bue Pedersen. "PicPrint: Embedding pictures in additive manufacturing". In: *euspen and ASPE Special Interest Group Meeting: Additive Manufacturing*. 2017.

[NL20] Simon Niklaus and Feng Liu. "Softmax Splatting for Video Frame Interpolation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pages 5437–5446.

[PBL09] Rasmus R Paulsen, Jakob Andreas Baerentzen, and Rasmus Larsen. "Markov random field surface reconstruction". In: *IEEE transactions on visualization and computer graphics* 16.4 (2009), pages 636–646.

[PP10] Soon-Yong Park and Go Gwang Park. "Active calibration of camera-projector systems based on planar homography". In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pages 320–323.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pages 533–536.

[RRT97] Carsten Reich, Reinhold Ritter, and Jan Thesing. “White light heterodyne principle for 3D-measurement”. In: *Sensors, Sensor Systems, and Sensor Data Processing*. Volume 3100. International Society for Optics and Photonics. 1997, pages 236–244.

[Sad+05] Filip Sadlo, Tim Weyrich, Ronald Peikert, and Markus Gross. “A practical structured light acquisition system for point-based geometry and texture”. In: *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* IEEE. 2005, pages 89–145.

[Sal+10] Joaquim Salvi, Sergio Fernandez, Tomislav Pribanic, and Xavier Llado. “A state of the art in structured light patterns for surface profilometry”. In: *Pattern recognition* 43.8 (2010), pages 2666–2680.

[SC08] Zhan Song and Ronald Chung. “Use of LCD panel for calibrating structured-light-based range sensing system”. In: *IEEE Transactions on Instrumentation and Measurement* 57.11 (2008), pages 2623–2630.

[Sei+06] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. “A comparison and evaluation of multi-view stereo reconstruction algorithms”. In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR’06)*. Volume 1. IEEE. 2006, pages 519–528.

[SOS04] Chen Shen, James F O’Brien, and Jonathan R Shewchuk. “Interpolating and approximating implicit surfaces from polygon soup”. In: *ACM SIGGRAPH 2004 Papers*. 2004, pages 896–904.

[Ull79] Shimon Ullman. “The interpretation of structure from motion”. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 203.1153 (1979), pages 405–426.

[UVL18] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Deep image prior”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pages 9446–9454.

[Wal+14] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. “scikit-image: image processing in Python”. In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.

[Wil+18] Jakob Wilm, Daniel González Madruga, Janus Nørtoft Jensen, Søren Kimmer Schou Gregersen, Mads Emil Brix Doest, Maria Grazia Guerra, Henrik Aanæs, and Leonardo De Chiffre. "Effects of subsurface scattering on the accuracy of optical 3D measurements using miniature polymer step gauges". In: *euspen's 18th International Conference and Exhibition*. 2018, pages 449–450.

[YXA04] Tianli Yu, Ning Xu, and Narendra Ahuja. "Shape and view independent reflectance map from multiple views". In: *European Conference on Computer Vision*. Springer. 2004, pages 602–615.

[YXA07] Tianli Yu, Ning Xu, and Narendra Ahuja. "Shape and view independent reflectance map from multiple views". In: *International journal of computer vision* 73.2 (2007), pages 123–138.

[ZH06] Song Zhang and Peisen S Huang. "Novel method for structured light system calibration". In: *Optical Engineering* 45.8 (2006), page 083601.

[Zha99] Zhengyou Zhang. "Flexible camera calibration by viewing a plane from unknown orientations". In: *Proceedings of the seventh ieee international conference on computer vision*. Volume 1. Ieee. 1999, pages 666–673.

[ZS00] Li Zhang and Steven M Seitz. "Image-based multiresolution shape recovery by surface deformation". In: *Videometrics and Optical Methods for 3D Shape Measurement*. Volume 4309. International Society for Optics and Photonics. 2000, pages 51–61.

CONTRIBUTION A

Video Frame Interpolation via Cyclic Fine-Tuning and Asymmetric Reverse Flow

Video Frame Interpolation via Cyclic Fine-Tuning and Asymmetric Reverse Flow

Morten Hannemose[1], Janus Nørtoft Jensen[1], Gudmundur Einarsson[1], Jakob Wilm[2], Anders Bjorholm Dahl[1], and Jeppe Revall Frisvad[1]

[1] DTU Compute, Technical University of Denmark
[2] SDU Robotics, University of Southern Denmark

Abstract. The objective in video frame interpolation is to predict additional in-between frames in a video while retaining natural motion and good visual quality. In this work, we use a convolutional neural network (CNN) that takes two frames as input and predicts two optical flows with pixelwise weights. The flows are from an unknown in-between frame to the input frames. The input frames are warped with the predicted flows, multiplied by the predicted weights, and added to form the in-between frame. We also propose a new strategy to improve the performance of video frame interpolation models: we reconstruct the original frames using the learned model by reusing the predicted frames as input for the model. This is used during inference to fine-tune the model so that it predicts the best possible frames. Our model outperforms the publicly available state-of-the-art methods on multiple datasets.

Keywords: slow motion · video frame interpolation · convolutional neural networks.

1 Introduction

Video frame interpolation, also known as inbetweening, is the process of generating intermediate frames between two consecutive frames in a video sequence. This is an important technique in computer animation [19], where artists draw keyframes and lets software interpolate between them. With the advent of high frame rate displays that need to display videos recorded at lower frame rates, inbetweening has become important in order to perform frame rate up-conversion [2]. Computer animation research [9, 19] indicates that good inbetweening cannot be obtained based on linear motion, as objects often deform and follow nonlinear paths between frames. In an early paper, Catmull [3] interestingly argues that inbetweening is "akin to difficult artificial intelligence problems" in that it must be able understand the content of the images in order to accurately handle e.g. occlusions. Applying learning-based methods to the problem of inbetweening thus seems an interesting line of investigation.

Some of the first work on video frame interpolation using CNNs was presented by Niklaus et al. [17, 18]. Their approach relies on estimating kernels to jointly represent motion and interpolate intermediate frames. Concurrently, Liu

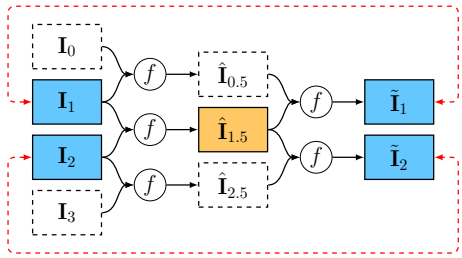


Fig. 1. Diagram illustrating the cyclic fine-tuning process when predicting frame $\hat{\mathbf{I}}_{1.5}$. The model is first applied in a pairwise manner on the four input frames $\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2$, and $\mathbf{I}_3$, then on the results $\hat{\mathbf{I}}_{0.5}$,$\hat{\mathbf{I}}_{1.5}$, and $\hat{\mathbf{I}}_{2.5}$. The results of the second iteration, $\tilde{\mathbf{I}}_1$ and $\tilde{\mathbf{I}}_2$, are then compared with the input frames and the weights of the network are updated. This process optimizes our model specifically to be good at interpolating frame $\hat{\mathbf{I}}_{1.5}$.

et al. [11] and Jiang et al. [6] used neural networks to predict optical flow and used it to warp the input images followed by a linear blending.

Our contribution is twofold. Firstly, we propose a CNN architecture that directly estimates asymmetric optical flows and weights from an unknown intermediate frame to two input frames. We use this to interpolate the frame in-between. Existing techniques either assume that this flow is symmetric or use a symmetric approximation followed by a refinement step [6, 11, 16]. For non-linear motion, this assumption does not hold, and we document the effect of relaxing it. Secondly, we propose a new strategy for fine-tuning a network for each specific frame in a video. We rely on the fact that interpolated frames can be used to estimate the original frames by applying the method again with the in-between frames as input. The similarity of reconstructed and original frames can be considered a proxy for the quality of the interpolated frames. For each frame we predict, the model is fine-tuned in this manner using the surrounding frames in the video, see Figure 1. This concept is not restricted to our method and could be applied to other methods as well.

2 Related work

Video frame interpolation is usually done in two steps: motion estimation followed by frame synthesis. Motion estimation is often performed using optical flow [1, 4, 25], and optical flow algorithms have used interpolation error as an error metric [1, 12, 23]. Frame synthesis can then be done via e.g. bilinear interpolation and occlusion reasoning using simple hole filling. Other methods use phase decompositions of the input frames to predict the phase decomposition of the intermediate frame and invert this for frame generation [14, 15], or they use local per pixel convolution kernels on the input frames to both represent motion and synthesize new frames [17, 18]. Mahajan et al. [13] determine where each pixel in an intermediate frame comes from in the surrounding input frames by

solving an expensive optimization problem. Our method is similar but replaces the optimization step with a learned neural network.

The advent of CNNs has prompted several new learning based approaches. Liu et al. [11] train a CNN to predict a symmetrical optical flow from the intermediate frame to the surrounding frames. They synthesize the target frame by interpolating the values in the input frames. Niklaus et al. [17] train a network to output local 38x38 convolution kernels for each pixel to be applied on the input images. In [18], they are able to improve this to 51x51 kernels. However, their representation is still limited to motions within this range. Jiang et al. [6] first predict bidirectional optical flows between two input frames. They combine these to get a symmetric approximation of the flows from an intermediate frame to the input frames, which is then refined in a separate step. Our method, in contrast, directly predicts the final flows to the input frames without the need for an intermediate step. Niklaus et al. [16] also initially predict bidirectional flows between the input frames and extract context maps for the images. They warp the input images and context maps to the intermediate time step using the predicted flows. Another network blends these to get the intermediate frame.

Liu et al. [10] propose a new loss term, which they call cycle consistency loss. This is a loss based on how well the output frames of a model can reconstruct the input frames. They retrain the model from [11] with this and show state-of-the-art results. We use this loss term and show how it can be used during inference to improve results. Meyer et al. [14] estimate the phase of an intermediate frame from the phases of two input frames represented by steerable pyramid filters. They invert the decomposition to reconstruct the image. This method alleviates some of the limitations of optical flow, which are also limitations of our method: sudden light changes, transparency and motion blur, for example. However, their results have a lower level of detail.

3 Method

Given a video containing the image sequence $\mathbf{I}_0, \mathbf{I}_1, \cdots, \mathbf{I}_n$, we are interested in computing additional images that can be inserted in the original sequence to increase the frame rate, while keeping good visual quality in the video. Our method doubles the frame rate, which allows for the retrieval of approximately any in-between frame by recursive application of the method. This means that we need to compute estimates of $\mathbf{I}_{0.5}, \mathbf{I}_{1.5}, \cdots, \mathbf{I}_{n-0.5}$, such that the final sequence would be:

$$\mathbf{I}_0, \mathbf{I}_{0.5}, \mathbf{I}_1, \cdots, \mathbf{I}_{n-0.5}, \mathbf{I}_n.$$

We simplify the problem by only looking at interpolating a single frame $\mathbf{I}_1$, that is located temporally between two neighboring frames $\mathbf{I}_0$ and $\mathbf{I}_2$. If we know the optical flows from the missing frame to each of these and denote them as $\mathbf{F}_{1\to0}$ and $\mathbf{F}_{1\to2}$, we can compute an estimate of the missing frame by

$$\hat{\mathbf{I}}_1 = \mathbf{W}_0 \mathcal{W}(\mathbf{F}_{1\to0}, \mathbf{I}_0) + \mathbf{W}_2 \mathcal{W}(\mathbf{F}_{1\to2}, \mathbf{I}_2), \tag{1}$$

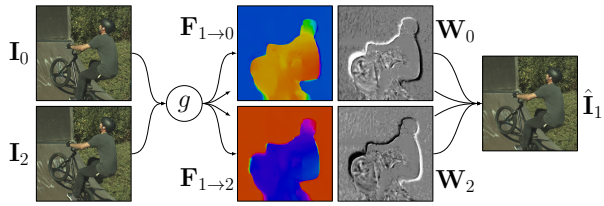


Fig. 2. Illustration of the frame interpolation process with g from Equation (2). From left to right: Input frames, predicted flows, weights and final interpolated frame.

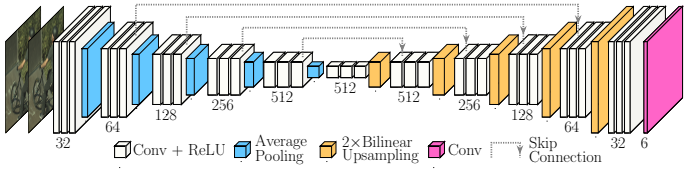


Fig. 3. The architecture of our network. Input is two color images $\mathbf{I}_0$ and $\mathbf{I}_2$ and output is optical flows $\mathbf{F}_{1\to0}, \mathbf{F}_{1\to2}$, and weights $\mathbf{W}_0, \mathbf{W}_2$. Convolutions are 3×3 and average pooling is 2×2 with a stride of 2. Skip connections are implemented by adding the output of the layer that arrows emerge from to the output of the layers they point to.

where $\mathcal{W}(\cdot, \cdot)$ is the backward warping function that follows the vector to the input frame and samples a value with bilinear interpolation. $\mathbf{W}_0$ and $\mathbf{W}_2$ are weights for each pixel describing how much of each of the neighboring frames should contribute to the middle frame. The weights are used for handling occlusions. Examples of flows and weights can be seen in Figure 2. We train a CNN g with a U-Net [20] style architecture, illustrated in Figure 3. The network takes two images as input and predicts the flows and pixel-wise weights

$$g(\mathbf{I}_0, \mathbf{I}_2) \to \mathbf{F}_{1\to0}, \mathbf{F}_{1\to2}, \mathbf{W}_0, \mathbf{W}_2. \tag{2}$$

Our architecture uses five 2×2 average pooling layers with stride 2 for the encoding and five bilinear upsampling layers to upscale the layers with a factor 2 in the decoding. We use four skip connections (addition) between layers in the encoder and decoder. It should be noted that our network is fully convolutional, which implies that it works on images of any size, where both dimensions are a multiple of 32. If this is not the case, we pad the image with boundary reflections.

Our model for frame interpolation is obtained by combining Equations (1) and (2) into

$$f(\mathbf{I}_0, \mathbf{I}_2) = \hat{\mathbf{I}}_1, \tag{3}$$

where $\hat{\mathbf{I}}_1$ is the estimated image. The model is depicted in Figure 2. All components of f are differentiable, which means that our model is end-to-end trainable. It is easy to get data in the form of triplets $(\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2)$ by taking frames from videos that we use as training data for our model.

3.1 Loss-functions

We employ a number of loss functions to train our network. All of our loss functions are given for a single triplet $(\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2)$, and the loss for a minibatch of triplets is simply the mean of the loss for each triplet. In the following paragraphs, we define the different loss functions that we employ.

Reconstruction loss models how well the network has reconstructed the missing frame:

$$\mathcal{L}_1 = \left|\left|\mathbf{I}_1 - \hat{\mathbf{I}}\right|\right|_1 . \tag{4}$$

Bidirectional reconstruction loss models how well each of our predicted optical flows is able to reconstruct the missing frame on its own:

$$\mathcal{L}_b = ||\mathbf{I}_1 - \mathcal{W}(\mathbf{F}_{1\to 0}, \mathbf{I}_0)||_1 + ||\mathbf{I}_1 - \mathcal{W}(\mathbf{F}_{1\to 2}, \mathbf{I}_2)||_1 . \tag{5}$$

This has similarities to the work of Jiang et al. [6] but differs since the flow is estimated from the missing frame to the existing frames, and not between the existing frames.

Feature loss is introduced as an approximation of the perceptual similarity by comparing feature representation of the images from a pre-trained deep neural network [7]. Let ϕ be the output of `relu4_4` from VGG19 [21], then

$$\mathcal{L}_f = \left|\left|\phi(\mathbf{I}_1) - \phi(\hat{\mathbf{I}}_1)\right|\right|_2^2 . \tag{6}$$

Smoothness loss is a penalty on the absolute difference between neighboring pixels in the flow field. This encourages a smoother optical flow [6,11]:

$$\mathcal{L}_s = ||\nabla \mathbf{F}_{1\to 0}||_1 + ||\nabla \mathbf{F}_{1\to 2}||_1 , \tag{7}$$

where $||\nabla \mathbf{F}||_1$ is the sum of the anisotropic total variation for each (x, y) component in the optical flow $\mathbf{F}$. For ease of notation, we introduce a linear combination of Equations (4) to (7):

$$\mathcal{L}_r(\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2) = \lambda_1 \mathcal{L}_1 + \lambda_b \mathcal{L}_b + \lambda_f \mathcal{L}_f + \lambda_s \mathcal{L}_s . \tag{8}$$

Note that we explicitly express this as a function of a triplet. When this triplet is the three input images, we define

$$\mathcal{L}_\alpha = \mathcal{L}_r(\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2) . \tag{9}$$

Similarly, for ease of notation, let the bidirectional loss from Equation (5) be a function

$$\mathcal{L}_B(\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2, \mathbf{F}_{1\to 0}, \mathbf{F}_{1\to 2}) = \mathcal{L}_b \tag{10}$$

where, in this case, $\mathbf{F}_{1\to 0}$ and $\mathbf{F}_{1\to 2}$ are the flows predicted by the network.

Pyramid loss is a sum of bidirectional losses for downscaled versions of images and flow maps:

$$\mathcal{L}_p = \sum_{l=1}^{l=4} 4^l \mathcal{L}_B\Big(A_l(\mathbf{I}_0), A_l(\mathbf{I}_1), A_l(\mathbf{I}_2), A_l(\mathbf{F}_{1\to 0}), A_l(\mathbf{F}_{1\to 2})\Big), \tag{11}$$

where A_l is the $2^l \times 2^l$ average pooling operator with stride 2^l.

Cyclic loss functions. We can apply our model recursively to get another estimate of $\mathbf{I}_1$, namely

$$\tilde{\mathbf{I}}_1 = f\left(\hat{\mathbf{I}}_{0.5}, \hat{\mathbf{I}}_{1.5}\right) = f\Big(f\left(\mathbf{I}_0, \mathbf{I}_1\right), f\left(\mathbf{I}_1, \mathbf{I}_2\right)\Big). \tag{12}$$

Cyclic loss is introduced to ensure that outputs from the model work well as inputs to the model [10]. It is defined by

$$\mathcal{L}_c = \mathcal{L}_r\left(\hat{\mathbf{I}}_{0.5}, \mathbf{I}_1, \hat{\mathbf{I}}_{1.5}\right). \tag{13}$$

Motion loss is introduced in order to get extra supervision on the optical flow and utilizes the recursive nature of our network.

$$\mathcal{L}_m = ||\mathbf{F}_{1\to 0} - 2\mathbf{F}_{1\to 0.5}||_2^2 + ||\mathbf{F}_{1\to 2} - 2\mathbf{F}_{1\to 1.5}||_2^2 \tag{14}$$

This is introduced as self-supervision of the optical flow, under the assumption that the flow $\mathbf{F}_{1\to 0}$ is approximately twice that of $\mathbf{F}_{1\to 0.5}$ and similarly for $\mathbf{F}_{1\to 2}$ and $\mathbf{F}_{1\to 1.5}$, and assuming that the flow is easier to learn for shorter time steps.

3.2 Training

We train our network using the assembled loss function

$$\mathcal{L} = \mathcal{L}_\alpha + \mathcal{L}_c + \lambda_m \mathcal{L}_m, \tag{15}$$

where $\mathcal{L}_\alpha$, $\mathcal{L}_c$ and $\mathcal{L}_m$ are as defined in Equations (9), (13) and (14) with $\lambda_r = 1$, $\lambda_b = 1$, $\lambda_f = 8/3$, $\lambda_s = 10/3$ and $\lambda_m = 1/192$. The values have been selected based on the performance on a validation set.

We train our network using the Adam optimizer [8] with default values $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and with a minibatch size of 64.

Inspired by Liu et al. [10], we first train the network using only $\mathcal{L}_\alpha$. This is done on patches of size 128×128 for 150 epochs with a learning rate of 10^{-5}, followed by 50 epochs with a learning rate of 10^{-6}. We then train with the full loss function $\mathcal{L}$ on patches of size 256×256 for 35 epochs with a learning rate of 10^{-5} followed by 30 epochs with a learning rate of 10^{-6}. We did not use batch-normalization, as it decreased performance on our validation set. Due to the presence of the cyclic loss functions, four forward and backward passes are needed for each minibatch during the training with the full loss function.

Training data. We train our network on triplets of patches extracted from consecutive video frames. For our training data, we downloaded 1500 videos in 4k from youtube.com/4k and resized them to 1920×1080. For every four frames in the video not containing a scene cut, we chose a random 320×320 patch and cropped it from the first three frames. If any of these patches were too similar or if the mean absolute differences from the middle patch to the previous and following patches were too big, or too dissimilar, they were discarded to avoid patches that either had little motion or did not include the same object. Our final training set consists of 476,160 triplets.

Data augmentation. The data is augmented while we train the network by cropping a random patch from the 320×320 data with size as specified in the training details in Section 3.2. In this way, we use the training data more effectively. We also add a random translation to the flow between the patches, by offsetting the crop to the first and third patch while not moving the center patch [18]. This offset is ± 5 pixels in each direction. Furthermore, we also performed random horizontal flips and swapped the temporal order of the triplet.

3.3 Cyclic fine-tuning (CFT)

We introduce the concept of fine-tuning the model during inference for each frame that we want to interpolate and refer to this as cyclic fine-tuning (CFT). Recall that the cyclic loss $\mathcal{L}_c$ measures how well the predicted frames are able to reconstruct the original frames. This gives an indication of the quality of the interpolated frames. The idea of CFT is to exploit this property at inference time to improve interpolation quality. We do this by extending the cyclic loss to ± 2 frames around the desired frame and fine-tuning the network using these images only.

When interpolating frame $\mathbf{I}_{1.5}$, we would use surrounding frames $\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2$, and $\mathbf{I}_3$ to compute $\hat{\mathbf{I}}_{0.5}, \hat{\mathbf{I}}_{1.5}$, and $\hat{\mathbf{I}}_{2.5}$, which are then used to compute $\tilde{\mathbf{I}}_1$ and $\tilde{\mathbf{I}}_2$ as illustrated in Figure 1 on page 2. Note that the desired interpolated frame $\hat{\mathbf{I}}_{1.5}$ is used in the computation of both of the original frames. Therefore by fine-tuning of the network to improve the quality of the reconstructed original frames, we are improving the quality of the desired intermediate frame indirectly.

Specifically, we minimize the loss for each of the two triplets $(\hat{\mathbf{I}}_{0.5}, \mathbf{I}_1, \hat{\mathbf{I}}_{1.5})$ and $(\hat{\mathbf{I}}_{1.5}, \mathbf{I}_2, \hat{\mathbf{I}}_{2.5})$ with the loss for each triplet given by

$$\mathcal{L}_{CFT} = \mathcal{L}_c + \lambda_p \mathcal{L}_p, \tag{16}$$

where $\lambda_p = 10$, and $\mathcal{L}_p$ is the pyramid loss described in Section 3.1. In order for the model to be presented with slightly different samples, we only do this fine-tuning on patches of 256×256 with flow augmentation as described in the previous section. For computational efficiency, we only do this for 50 patch-triplets for each interpolated frame.

More than ± 2 frames can be applied for fine-tuning, however, we found that this did not increase performance. This fine-tuning process is not limited to our model and can be applied to any frame interpolation model taking two images as input and outputting one image.

4 Experiments

We evaluate variations of our method on three diverse datasets: UCF101 [22], SlowFlow [5] and See You Again [18]. These have previously been used for frame interpolation [6,11,18]. UCF101 contains different image sequences of a variety of actions, and we evaluate our method on the same frames as Liu et al. [11], but

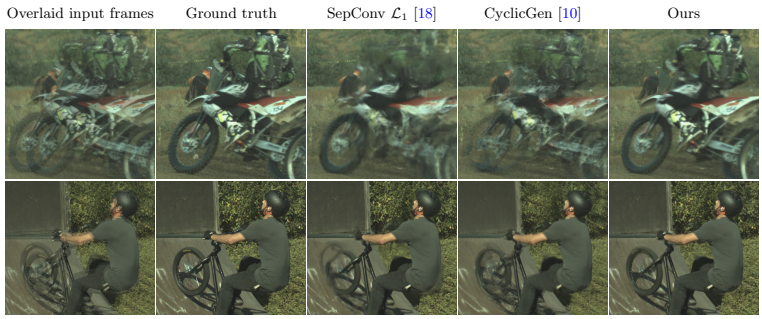


Fig. 4. Qualitative examples on the SlowFlow dataset. Images shown are representative crops taken from the full images. Note that our method performs much better for large motions. The motion of the dirt bike is approximately 53 pixels, and the bike tire has a motion of 34 pixels.

Table 1. Overview of the datasets we used for evaluation.

Dataset	Number of sequences	Resolution	Number of frames		Avg. sequence length
			Interpolated	Total	
SlowFlow [5]	34	$1280 \times \{1024, 720\}$	17,871	20,458	602
See You Again	117	1920×1080	2,503	5,355	46
UCF101 [22]	379	256×256	379	1,137	3

did not use any motion masks as we are interested in performing equally well over the entire frame. SlowFlow is a high-fps dataset that we include to showcase our performance when predicting multiple in-between frames. For this dataset, we have only used every eighth frame as input and predicted the remaining seven in-between in a recursive manner. All frames in the dataset have been debayered, resized to 1280 pixels on the long edge and gamma corrected with a gamma value of 2.2. See You Again is a high-resolution music video, where we predict the even-numbered frames using the odd-numbered frames. Furthermore, we have divided it into sequences by removing scene changes. A summary of the datasets is shown in Table 1.

We have compared our method with multiple state-of-the-art methods [6,10, 11,18] which either have publicly available code and/or published their predicted frames. For the comparison with SepConv [18], we use the $\mathcal{L}_1$ version of their network for which they report their best quantitative results. For each evaluation, we report the Peak Signal to Noise Ratio (PSNR) and the Structural Similarity Index (SSIM) [24].

Table 2. Interpolation results on SlowFlow, See You Again and UCF101. PSNR, SSIM: higher is better. Our baseline is our model trained only with $\mathcal{L}_1 + \mathcal{L}_b + \mathcal{L}_s$ and constrained to symmetric flow. Elements are added to the model cumulatively. Larger patches means training on 256×256 patches. Bold numbers signify that a method performs significantly better than the rest on that task with $p < 0.02$.

	SlowFlow		See You Again		UCF101	
Method	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DVF [11]	-	-	-	-	34.12	0.941
SuperSloMo [6]	-	-	-	-	34.75	0.947
SepConv $\mathcal{L}_1$ [18]	34.03	0.899	42.49	0.983	34.78	0.947
CyclicGen [10]	31.33	0.839	41.28	0.975	**35.11**	**0.949**
Our baseline	34.28	0.903	42.50	0.984	34.39	0.946
+ asymmetric flow	34.33	0.904	42.54	0.985	34.40	0.946
+ feature loss	34.29	0.901	42.62	0.984	34.60	0.948
+ cyclic loss	34.33	0.900	42.73	0.984	34.62	0.947
+ motion loss	34.31	0.900	42.74	0.984	34.61	0.948
+ larger patches	34.60	0.907	43.14	0.986	34.69	0.948
+ CFT	**34.91**	**0.912**	**43.21**	**0.986**	34.94	**0.949**

Comparison with state-of-the-art. Table 2 shows that our best method, with or without CFT, clearly outperforms the other methods on SlowFlow and See you Again, which is also reflected in Figure 4. On UCF101 our best method performs better than all other methods except CyclicGen, where our best method has the same SSIM but lower PSNR. We suspect this is partly due to the fact that our CFT does not have access to ± 2 frames in all sequences. For some of the sequences, we had to use $-1, +3$ as the intermediate frame was at the beginning of the sequence. Visually, our method produces better results as seen in Figure 5. We note that CyclicGen is trained on UCF101, and their much worse performance on the two other datasets could indicate overfitting.

Effect of various model configurations. Table 2 reveals that an asymmetric flow around the interpolated frame slightly improves performance on all three datasets as compared with enforcing a symmetric flow. There is no clear change in performance when we add feature loss, cyclic loss and motion loss.

For all three datasets, performance improves when we train on larger image patches. Using larger patches allows for the network to learn larger flows and the performance improvement is correspondingly seen most clearly in SlowFlow and See You Again which, as compared with UCF101, have much larger images with larger motions. We see a big performance improvement when cyclic fine-tuning is added, which is also clearly visible in Figure 6.

Discussion. Adding CFT to our model increases the run-time of our method by approximately 6.5 seconds per frame pair. This is not dependent on image

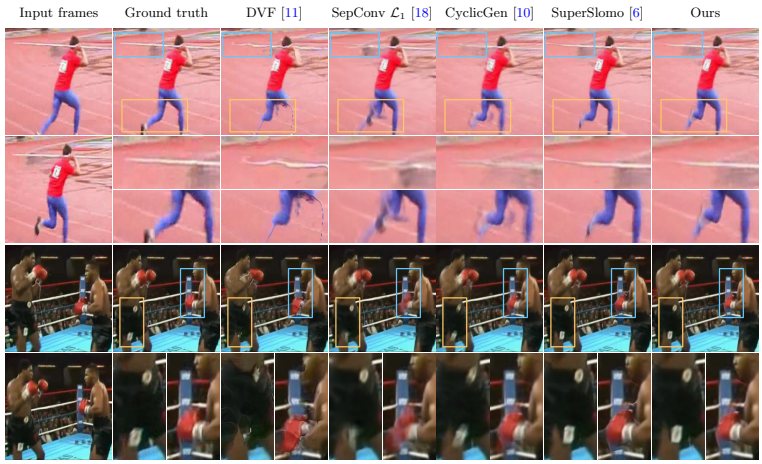


Fig. 5. Qualitative comparison for two sequences from UCF101. Top: Our method produces the least distorted javelin and retains the detailed lines on the track. All methods perform inaccurately on the leg, however, SuperSlomo and our method are the most visually plausible. Bottom: Our method and SuperSlomo create accurate white squares on the shorts (left box). Our method also produces the least distorted ropes and white squares on the corner post, while creating foreground similar to the ground truth (right box).

size, as we only fine-tune on 256×256 patches for 50 iterations per frame pair. For reference, our method takes 0.08 seconds without CFT to interpolate a 1920×1080 image on an NVIDIA GTX 1080 TI. It should be noted that CFT is only necessary to do once per frame pair in the original video, and thus there is no extra overhead when computing multiple in-between frames.

Training for more than 50 iterations does not necessarily ensure better results, as we can only optimize a proxy of the interpolation quality. The best number of iterations remains to be determined, but it is certainly dependent on the quality of the pre-training, the training parameters, and the specific video.

As of now, CFT should only be used if the target is purely interpolation quality. Improving the speed of CFT is a topic worthy of further investigation. Possible solutions of achieving similar results include training a network to learn the result of CFT, or training a network to predict the necessary weight changes.

5 Conclusion

We have proposed a CNN for video frame interpolation that predicts two optical flows with pixelwise weights from an unknown intermediate frame to the frames

Fig. 6. Representative example of how the cyclic fine-tuning improves the interpolated frame. It can be seen that the small misalignment of the tire and yellow "41" is corrected by the cyclic fine-tuning.

before and after. The flows are used to warp the input frames to the intermediate time step. These warped frames are then linearly combined using the weights to obtain the intermediate frame. We have trained our CNN using 1500 high-quality videos and shown that it performs better than or comparably to state-of-the-art methods across three different datasets. Furthermore, we have proposed a new strategy for fine-tuning frame interpolation methods for each specific frame at evaluation time. When used with our model, we have shown that it improves both the quantitative and visual results.

Acknowledgements. We would like to thank Joel Janai for providing us with the SlowFlow data [5].

References

1. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. International Journal of Computer Vision **92**(1), 1–31 (2011)
2. Castagno, R., Haavisto, P., Ramponi, G.: A method for motion adaptive frame rate up-conversion. IEEE Transactions on Circuits and Systems for Video Technology **6**(5), 436–446 (October 1996)
3. Catmull, E.: The problems of computer-assisted animation. Computer Graphics (SIGGRAPH '78) **12**(3), 348–353 (August 1978)
4. Herbst, E., Seitz, S., Baker, S.: Occlusion reasoning for temporal interpolation using optical flow. Tech. rep., Microsoft Research (August 2009)
5. Janai, J., Güney, F., Wulff, J., Black, M., Geiger, A.: Slow Flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1406–1416 (2017)
6. Jiang, H., Sun, D., Jampani, V., Yang, M.H., Learned-Miller, E., Kautz, J.: Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In: Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9000–9008 (2018)

7. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision (ECCV). pp. 694–711 (2016)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
9. Lasseter, J.: Principles of traditional animation applied to 3D computer animation. Compututer Graphics (SIGGRAPH '87) **21**(4), 35–44 (August 1987)
10. Liu, Y.L., Liao, Y.T., Lin, Y.Y., Chuang, Y.Y.: Deep video frame interpolation using cyclic frame generation. In: AAAI Conference on Artificial Intelligence (2019)
11. Liu, Z., Yeh, R.A., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: International Conference on Computer Vision. pp. 4463–4471 (2017)
12. Long, G., Kneip, L., Alvarez, J.M., Li, H., Zhang, X., Yu, Q.: Learning image matching by simply watching video. In: ECCV. pp. 434–450. Springer, Cham (2016)
13. Mahajan, D., Huang, F.C., Matusik, W., Ramamoorthi, R., Belhumeur, P.: Moving gradients: A path-based method for plausible image interpolation. ACM Transactions on Graphics **28**(3), 42:1–42:11 (July 2009)
14. Meyer, S., Djelouah, A., McWilliams, B., Sorkine-Hornung, A., Gross, M., Schroers, C.: Phasenet for video frame interpolation. In: Conference on Computer Vision and Pattern Recognition (CVPR). pp. 498–507 (2018)
15. Meyer, S., Wang, O., Zimmer, H., Grosse, M., Sorkine-Hornung, A.: Phase-based frame interpolation for video. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1410–1418 (2015)
16. Niklaus, S., Liu, F.: Context-aware synthesis for video frame interpolation. In: Conference on Computer Vision and Pattern Recognition. pp. 1701–1710 (2018)
17. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive convolution. In: Conference on Computer Vision and Pattern Recognition. pp. 670–679 (2017)
18. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: International Conference on Computer Vision. pp. 261–270 (2017)
19. Reeves, W.T.: Inbetweening for computer animation utilizing moving point constraints. Computer Graphics (SIGGRAPH '81) **15**(3), 263–269 (August 1981)
20. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
22. Soomro, K., Zamir, A.R., Shah, M., Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. CoRR **abs/1212.0402** (2012)
23. Szeliski, R.: Prediction error as a quality metric for motion and stereo. In: IEEE International Conference on Computer Vision (ICCV). vol. 2, pp. 781–788 (1999)
24. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004)
25. Werlberger, M., Pock, T., Unger, M., Bischof, H.: Optical flow guided TV-L^1 video interpolation and restoration. In: Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR). pp. 273–286. Springer (2011)

CONTRIBUTION B

Generating Spatial Attention Cues via Illusory Motion

Generating Spatial Attention Cues via Illusory Motion

Janus Nørtoft Jensen[1*] jnje@dtu.dk
Morten Hannemose[1*] mohan@dtu.dk
Jakob Wilm[2] jaw@mmmi.sdu.dk
Anders Bjorholm Dahl[1] abda@dtu.dk
Jeppe Revall Frisvad[1] jerf@dtu.dk
Serge Belongie[3] sjb344@cornell.edu

[1]Technical University of Denmark [2]University of Southern Denmark [3]Cornell Tech

Abstract

For many applications in augmented reality (AR), the user has a much more enjoyable experience if the AR system is able to properly guide the user's attention. In this extended abstract, we explain how to create patterns of light that when projected onto an object are perceived as if the object itself is moving. This can be used as a spatial attention cue. We accomplish this with a calibrated projector-camera setup to synthesize an image from the projector's point of view. This image is filtered to create local phase changes that are then projected back onto the object and perceived as motion. Our method will be shown as a live demonstration at the CV4AR/VR workshop at CVPR 2019.

(a) Left camera (b) Projector (c) Right camera

Figure 1. Example of a synthesized image from the projector's point of view (b), along with the camera images the intensities are sampled from (a, c).

1. Introduction

Visual attention is important as it affects our performance in many visual tasks [2]. To successfully accomplish tasks such as visual search or tele-assistance we need effective spatial attention cues to attract the attention of a user. What constitutes an effective cue is in part determined by the task at hand. A very obvious cue, such as a big bouncing red arrow, might be the best, if the purpose is to alert the user of possible danger. In many other situations, however, more subtle cues might be preferred. This could be in the setting of an escape room game where we would guide players towards the next clue if they are stuck. A very obvious cue could possibly ruin the fun of the game, however a subtle one, which would take longer to notice, could be useful.

As described by Carrasco and Barbot [2], our attention is involuntarily captured by sudden changes in the environment. We seek to exploit this effect by creating apparent motion through light projection. While the human ability to detect motion is not better in the peripheral vision [10], the speed of visual processing does increase in the peripheral vision [3]. When something moves differently compared to the movement of the observer, it becomes a powerful cue to attract attention, especially in the periphery of the visual field [12]. This is referred to as a relative-motion cue. Our idea is to guide attention by creating apparent relative-motion cues in a scene by means of a light projector that modifies the appearance of a physical object to make it look as if it were moving, thereby creating illusory motion.

2. Projecting Illusory Motion

Prior work in guiding visual attention has mainly been focused on head-mounted displays and images displayed on a screen. The spatial attention cues used include flickering [16, 17], blurring [8], and color manipulation [9]. Spatial projection has been shown to be more effective than head-mounted displays in providing spatial instructions in assembly tasks [1]. Research in attention cues for spatial projection is however limited. Some use laser projection [14, 7] as a simple cue. Similar to our approach Taki-

*These authors contributed equally to this work.

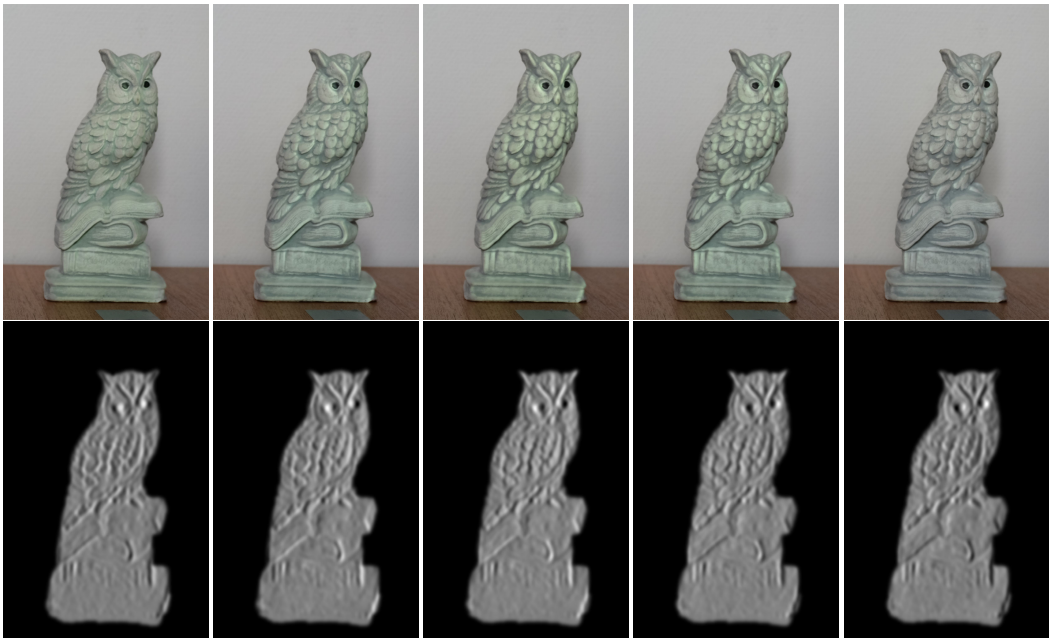


Figure 2. Five frames sampled from the continuous loop of motion our method produces. Top row: Picture of object taken with camera. Bottom row: Image projected by projector to create corresponding picture. Videos showing the effect can be seen at http://people.compute.dtu.dk/jnje/illusory-motion.

moto et al. [15] uses a calibrated projector-camera setup. They modulate color information recorded with the camera and project the result back onto the object. For humans, the sensitivity to color variations declines faster compared with the sensitivity to luminance [6]. It thus seems natural to investigate modifying the luminance instead.

Our approach to create the illusion of motion is based on adapting the work by Freeman et al. [4] to a projector-camera setup. In short, they apply local filters with continuously varying phase over time to the image. This is based on the observation that local phase changes are interpreted as global motion. We synthesize an image from the viewpoint of the projector (Figure 1) and use it as input for their method. The filter response is then projected back onto the object. In Figure 2, examples of the object with the filter response projected onto it are shown along with the images projected.

2.1. Projector-camera calibration

We use two cameras and a projector mounted in a fixed setup, and we model the projector and cameras as pinhole cameras with radial distortion. To calibrate the system we encode the projector's pixel coordinates using structured light [13], detect corners in images of a checkerboard and convert these to the projector's pixel space by local homographies [11]. The projector and cameras are then calibrated with Zhang's method [18].

2.2. Synthesizing image from projector's point of view

To synthesize an image from the projector's point of view we use structured light to create two 3D-scans of the object - one based on each camera. The resulting point clouds and the pixel intensities associated with the points are then projected back to the projector to form the desired image (Figure 1). Creating two separate 3D-scans enables us to scan all points visible in the projector and at least one camera, thereby getting better coverage of the object. After the points from each scan have been projected to the projector's pixel space they are rounded to the nearest integer pixel. Because the projector has a lower resolution than the cameras, each pixel in the projector contains multiple measurements. We compute medians of these to obtain a single value per pixel.

2.3. Creating Illusory Motion

With the image from the projector's point of view, we can directly apply the method of Freeman et al. [4] to this

image. As the filter responses contain both positive and negative values, and the projective setting has the constraint of only adding light, we add a constant value to the filter response to make all values positive. The filtered image is multiplied with a mask to restrict light to the object, and the resulting image is projected onto the object. Examples of the projected image and the resulting effect are in Figure 2.

3. Future work

We can perhaps utilize the 3D information obtained through our process to make the projected patterns less view dependent. Because we know the 3D position of each pixel in addition to its intensity, we can compute a normal at each point. If we, based on these normals, choose a consistent orthonormal basis at each point, such that the basis changes smoothly over the object [5], and assuming that the object is locally planar, we can project the filter onto this plane, and thereby approximate convolution along the surface of the object instead of in image space.

Our current work has been focused on creating the illusion of motion. Further work is needed to determine under which circumstances it is perceived as motion and to determine its effectiveness as a spatial attention cue. Furthermore, examining how it affects the user's experience of interacting with the AR system, and how our cue compares with using alternative spatial attention cues is important to examine. Reasonable variables to look into would be how much light is necessary to make the effect noticeable and how this varies across the visual field.

References

[1] Sebastian Büttner, Markus Funk, Oliver Sand, and Carsten Röcker. Using head-mounted displays and in-situ projection for assistive systems: A comparison. In *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '16, pages 44:1–44:8. ACM, 2016. 1

[2] Marisa Carrasco and Antoine Barbot. Spatial attention alters visual appearance. *Current opinion in psychology*, 29:56–64, 2019. 1

[3] Marisa Carrasco, Brian McElree, Kristina Denisova, and Anna Marie Giordano. Speed of visual processing increases with eccentricity. *Nature Neuroscience*, 6(7):699, 2003. 1

[4] William T Freeman, Edward H Adelson, and David J Heeger. *Motion without movement*, volume 25. Citeseer, 1991. 2

[5] Jeppe Revall Frisvad. Building an orthonormal basis from a 3d unit vector without normalization. *Journal of Graphics Tools*, 16(3):151–159, 2012. 3

[6] Thorsten Hansen, Lars Pracejus, and Karl R Gegenfurtner. Color perception in the intermediate periphery of the visual field. *Journal of Vision*, 9(4):26–26, 2009. 2

[7] Harald Haraldsson, Doron Tal, Karla Polo-Garcia, and Serge Belongie. Pointar: Augmented reality for tele-assistance. In *CVPR Workshop on Embedded Computer Vision*, Salt Lake City, UT, 2018. 1

[8] Hajime Hata, Hideki Koike, and Yoichi Sato. Visual guidance with unnoticed blur effect. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '16, pages 28–35. ACM, 2016. 1

[9] Victor A. Mateescu and Ivan V. Bajić. Attention retargeting by color manipulation in images. In *Proceedings of the 1st International Workshop on Perception Inspired Video Processing*, PIVP '14, pages 15–20. ACM, 2014. 1

[10] Suzanne P McKee and Ken Nakayama. The detection of motion in the peripheral visual field. *Vision research*, 24(1):25–32, 1984. 1

[11] Daniel Moreno and Gabriel Taubin. Simple, accurate, and robust projector-camera calibration. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 464–471. IEEE, 2012. 2

[12] Dorothe A Poggel, Hans Strasburger, and Manfred MacKeben. Cueing attention by relative motion in the periphery of the visual field. *Perception*, 36(7):955–970, 2007. 1

[13] Carsten Reich, Reinhold Ritter, and Jan Thesing. White light heterodyne principle for 3d-measurement. In *Sensors, Sensor Systems, and Sensor Data Processing*, volume 3100, pages 236–245. International Society for Optics and Photonics, 1997. 2

[14] Björn Schwerdtfeger, Daniel Pustka, Andreas Hofhauser, and Gudrun Klinker. Using laser projectors for augmented reality. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, pages 134–137. ACM, 2008. 1

[15] Hironori Takimoto, Katsumi Yamamoto, Akihiro Kanagawa, Mitsuyoshi Kishihara, and Kensuke Okubo. Guiding visual attention based on visual saliency map with projector-camera system. In *HCI International 2017 – Posters' Extended Abstracts*, pages 383–390. Springer International Publishing, 2017. 2

[16] N. Waldin, M. Waldner, and I. Viola. Flicker observer effect: Guiding attention through high frequency flicker in images. *Computer Graphics Forum*, 36(2):467–476, 2017. 1

[17] M. Waldner, M. Le Muzic, M. Bernhard, W. Purgathofer, and I. Viola. Attractive flicker guiding attention in dynamic narrative visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2456–2465, Dec 2014. 1

[18] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000. 2

CONTRIBUTION C

Surface Reconstruction from Structured Light Images using Differentiable Rendering

MDPI

Article

Surface Reconstruction from Structured Light Images Using Differentiable Rendering

Janus N. Jensen [1,†,*], Morten Hannemose [1,†], J. Andreas Bærentzen [1], Jakob Wilm [2], Jeppe R. Frisvad [1] and Anders B. Dahl [1]

[1] DTU Compute, Technical University of Denmark, Kgs. Lyngby, Denmark
[2] SDU Robotics, University of Southern Denmark, Odense, Denmark
* Correspondence: jnje@dtu.dk
† These authors contributed equally to this work.

Version December 20, 2020 submitted to Sensors

Abstract: When 3D scanning objects, the objective is usually to obtain a continuous surface. However, most surface scanning methods, such as structured light scanning, yield a point cloud. Obtaining a continuous surface from a point cloud requires a subsequent surface reconstruction step , which is directly affected by any error from the computation of the point cloud. Here, we investigate the possibility of using a one-step approach, where we compute the surface directly from structured light images. To do so, we propose a new method based on minimizing the least-squares error between real images and renderings of a triangle mesh, where the vertex positions of the mesh are the parameters of the minimization problem. We present simulation experiments demonstrating that our one-step method for computing a triangle mesh has several advantages over the two-step approach that relies on an intermediate point cloud. Our method can produce accurate reconstructions when initializing the optimization from a simple sphere. We also show that our method is good at reconstructing sharp edges and robust with respect to image noise. In addition, our method is useful for improving the output from other reconstruction algorithms if we use these as initialization.

Keywords: 3D surface reconstruction, 3D scanning, structured light, differentiable rendering

1. Introduction

Structured light 3D scanning of an object can be used to produce a point cloud from which we can reconstruct a triangle mesh. The resulting mesh is a digital representation of the surface of the scanned object. This has many applications including cultural heritage preservation and industrial quality control [1,2]. For most applications, the accuracy of a recovered surface of the reconstruction is of great import. Typically, producing point clouds from phase-shifting structured light images is rather cumbersome. It involves determining the phases, unwrapping these, re-sampling the unwrapped phases due to image distortion and rectification, finding point correspondences, and finally triangulating these. Afterwards, the point clouds from different sub-scans need to be merged before the final reconstruction of a triangle mesh. During this process of producing point clouds and subsequently a triangle mesh, the image noise propagates non-linearly to affect vertex positions in the reconstructed triangle mesh. We therefore propose to skip the point cloud, and the process of creating it, and instead reconstruct surfaces directly from image intensities to investigate how that affects the accuracy of the reconstructions. Using vertex positions as model parameters, we minimize the least-squares error between rendered and recorded images to obtain a triangle mesh directly. An example is in Figure 1.

Figure 1. Starting from a sphere, our method reconstructs the Stanford Bunny [3] from images with low levels of noise. From left to right: Initial mesh (sphere), after 50 iterations, after 750 iterations, and the converged result (1 952 iterations). The final reconstruction has 13 780 vertices and a volume error (Δ_V) of 0.30%, when compared to the ground truth (depicted in Figure 4j).

Our method has three major implications: (1) explicit point triangulation from image correspondences is no longer needed; (2) we understand how noise in the image data affects the final reconstruction; (3) the reconstruction is done for all image data simultaneously.

Similar approaches have been used for multi-view passive stereo but to the best of our knowledge not for structured light reconstructions. In structured light, dense correspondences can be established, usually yielding much higher accuracy in weakly textured areas.

In many reconstruction methods, the raw image data is not considered during the reconstruction [4]. The reconstruction is instead done using unstructured point clouds that have been constructed from the image data. This means that noise due to imaging and point matching processes is not modeled in the reconstruction at all, and this noise is then non-linearly propagated through to the end result. With our approach, the only step in the method where an error is minimized is in the image intensity domain. This enables us to minimize a more meaningful error, which is especially important in applications with a low signal-to-noise ratio of the reflected light, e.g., when scanning highly specular objects or when using short exposure times for fast acquisition.

2. Related work

Suppose we know the configuration of light and camera in a vision setup and the reflectance properties of the imaged object. Obtaining the shape of the object based on its shading in an acquired image is then referred to as the shape-from-shading problem [5]. The original shape-from-shading method by Horn [5,6] used so-called characteristic curves to describe the observed shape. The method was based on illumination from point-like sources and the shading would then only allow estimation of the gradient along a path. This was the reason for using a collection of curves to describe the object shape.

Curves are inconvenient in the sense that they require stitching to become a full surface description. One way to fit a mesh instead of curves is to use an optimization technique that does not require gradients. This has been done for a rectangular mesh using simulated annealing and simplex search [7]. Gradients are however preferable to ease the optimization problem. For triangle meshes, an approach has been developed based on image gradients [8]. Unfortunately, the shape can be hard to recover from image gradients due to color variance caused by normal variations in the surface. To keep gradient-based optimization while having a method more robust to surface reflectance deviating from an assumption of a specific shading model, we use structured light with a differentiable pattern.

Combination of shape from shading with a structured light approach like phase-shifting improves the performance of the shape estimation [9,10] and enables simultaneous acquisition of shape and object color (diffuse reflectance) [11]. We do not include estimation of spatially varying reflectance in this study, but we note that this is an option. Only a height field was reconstructed in this previous work. We take the concept one step further and reconstruct a closed 3D object as in the work of Zhang and Seitz [8] but also exploiting structured light.

The mesh-based reconstruction method of Zhang and Seitz [8] was improved by Isidoro and Sclaroff [12,13] through the creation of a better initial mesh and by Yu *et al.* [14,15] using a model that more accurately models the physical reflectance. Another option is to use multiview stereo to acquire a good initial guess and then refine the mesh using a shape-from-shading approach [16]. Our method can be used similarly with the innovation of using structured light to improve the robustness of the mesh refinement.

The use of structured light has become a strong technique for point-based 3D reconstruction [17] but has to the best of our knowledge not previously been tested in mesh-based 3D reconstruction. Our motivation is to have the benefits of a mesh-based technique. An important benefit is that the connectivity between points (vertices) is retained throughout the geometric refinement process.

The recent differentiable renderer from Loubet *et al.* [18] uses ray tracing and reverse mode automatic differentiation. The use of such a framework for mesh refinement is an option. Every rendering is however quite computationally demanding, so the optimization would have a significant run time. Liu *et al.* [19] introduced the soft rasterizer, which is a faster differentiable renderer based on a smoothed version of rasterization. This has shown promising results in other mesh reconstruction tasks, but to make it able to render the structured light of a projector is nontrivial.

3. Method

Our method fits a surface to a set of structured light images by minimizing the squared differences between rendered images and real images. We parameterize the object surface by a triangular mesh, and the parameters we optimize are thus the vertex positions $\mathbf{v}$. The real images are captured with structured light phase-shifting from multiple camera-projector positions. We denote these images $\mathbf{I}_{c,p}$, where $c \in \mathcal{C}$ is the index of the camera-projector pair and $p \in \mathcal{P}$ is the index of the projected pattern. We render images $\widetilde{\mathbf{I}}_{c,p}(\mathbf{v})$ of our parameterized surface from the same camera-projector positions and find the optimal vertex positions by solving the following minimization problem

$$\underset{\mathbf{v}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{v}) = \underset{\mathbf{v}}{\operatorname{argmin}} \sum_{c\in\mathcal{C}} \sum_{p\in\mathcal{P}} \left\| \mathbf{I}_{c,p} - \widetilde{\mathbf{I}}_{c,p}(\mathbf{v}) \right\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm, i.e., we minimize the sum of squared differences over all pixels for all patterns and all camera-projector pairs.

We generate our structured light images by projecting sinusoidal patterns of the form

$$\frac{1}{2} + \frac{1}{2}\sin\left(2\pi n_p x + \phi_p\right), \tag{2}$$

where x is the x-coordinate of the projector normalized to $[0, 1]$, n_p is the frequency (number of periods) of the pattern, and ϕ_p is a phase-shift. With structured light images, such as phase-shifted images made from Equation 2, the goal is usually to find the x-coordinate of the projector, which subsequently can be used for triangulation.

Our method is not specific to phase-shifting patterns, however, we use differentiable patterns to make the minimization problem tractable.

3.1. Rendering images

To solve the minimization problem in Equation 1 we need to render the images $\widetilde{\mathbf{I}}_{c,p}(\mathbf{v})$ in each iteration. We want to adequately reproduce the structured light images that would have been obtained if our current parameterized surface was a real object. We achieve this by simulating the structured light process as seen from the viewpoint of the camera. We render the images using the formula

$$\widetilde{\mathbf{I}}_{c,p}(\mathbf{v}) = \mathbf{A}_c \sin\left(2\pi n_p \widetilde{\mathbf{X}}_c(\mathbf{v}) + \phi_p\right) + \mathbf{B}_c. \tag{3}$$

Here, $\sin(\cdot)$ is elementwise application of the sine function, n_p is the number of periods as in Equation 2, ϕ_p is the phase-shift for the p^{th} pattern, and $\widetilde{\mathbf{X}}_c(\mathbf{v})$ contains the x-coordinates of the projector in the $[0, 1]$ range for each pixel. We use the x-coordinate of the projector as the projector is offset from the camera along its x-axis. The matrices $\mathbf{A}_c$ and $\mathbf{B}_c$ are amplitudes and biases that are estimated from the ground truth images by fitting sinusoids at each pixel location. We find the elements of $\widetilde{\mathbf{X}}_c(\mathbf{v})$ by tracing a ray from the camera through the center of each pixel and projecting the point where it intersects the surface back to the projector. As we model the projector as a pinhole camera, the point is projected to the projector as follows

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix}, \tag{4}$$

where $\mathbf{P}$ is the projection matrix of the projector, $\mathbf{p}_i$ is the i^{th} column of $\mathbf{P}$, and $\mathbf{r}$ is the 3D point where the ray intersects the triangle face. The i, j^{th} element of $\widetilde{X}_c$ is then given by

$$\widetilde{X}_c^{i,j} = \frac{q_1}{q_3}. \tag{5}$$

If the ray does not intersect the surface, we treat the pixel as background and set $\widetilde{X}_c^{i,j}(\mathbf{v}) := X_c^{i,j}$, such that the corresponding term in the loss $\mathcal{L}(\mathbf{v})$ will be zero.

Note that $2\mathbf{A}_c$ is the proportion of projector light that is reflected into the camera and $\mathbf{B}_c - \mathbf{A}_c$ is the amount of global light. We estimate these to have the renderings better resemble the true images. Fortunately, we need only estimate them once for each viewpoint, as we can then use these estimates repeatedly in each iteration of the optimization.

3.2. Optimizing the surface

We use gradient descent to solve the optimization problem in Equation 1. In order to do this, we need the gradient of $\mathcal{L}(\mathbf{v})$, which in turn depends on the gradient of the elements in $\widetilde{\mathbf{X}}_c(\mathbf{v})$. Recall that each of these elements is computed by tracing a single ray from the camera to the surface of the object. The gradient of $\widetilde{X}_c^{i,j}$ will therefore only have contributions from the vertices spanning the triangle face that intersects the ray. We can compute the derivative for one of these three vertices ($\mathbf{p}_a$) as follows

$$\frac{\partial \widetilde{X}_c^{i,j}}{\partial \mathbf{p}_a} = \left(\frac{\mathbf{p}_1 - \widetilde{X}_c^{i,j} \mathbf{p}_3}{q_3} \cdot \mathbf{d} \right) \frac{\lambda_a \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}, \tag{6}$$

where $\mathbf{d}$ is the ray direction, $\mathbf{n}$ is the normal of the face, and λ_a is the barycentric coordinate corresponding to $\mathbf{p}_a$. The equations for the remaining two vertices, $\mathbf{p}_b$ and $\mathbf{p}_c$, use λ_b and λ_c but are otherwise identical. For a derivation of Equation 6, see Appendix A.

As mentioned, we use gradient descent to update the vertex positions, that is

$$\mathbf{v}_{i+1} = \mathbf{v}_i - \alpha_i \nabla \mathcal{L}(\mathbf{v_i}), \tag{7}$$

where $\mathbf{v}_i$, and $\mathbf{v}_{i+1}$ are the vertex positions in the i^{th} and $(i+1)^{\text{th}}$ iterations, respectively. To choose the step-length α_i we use a simple backtracking line-search strategy to choose

$$\alpha_i := \frac{1}{2^n} \alpha, \tag{8}$$

where α is a fixed constant and n is the smallest non-negative integer such that $\mathcal{L}(\mathbf{v}_{i+1}) < \mathcal{L}(\mathbf{v}_i)$ when doing the update.

3.3. Initializing the optimization

Up until this point, we have described how the iterative part of the optimization problem works, but this is only one half of the problem. A good initial guess is extremely important to ensure convergence. In the next two sections, we will introduce two possible ways to obtain an initial guess for the minimization problem in Equation 1.

3.3.1. Using other reconstruction methods

One way of getting a good initial guess is by using a reconstruction found via another reconstruction method. In this way, our method can be seen as a post-processing step that tries to adjust the reconstruction to better fit to the original image data. In some of our experiments, we have used Screened Poisson Reconstructions [20] at various depths as initialization. When using Poisson reconstructions, we experience it often being beneficial to remesh the mesh before starting the optimization.

3.3.2. Using simple shape

Another way of getting a good initial guess is by solving a related, but simpler, minimization problem, and use the result as initialization for the minimization problem in Equation 1. If we denote the x-coordinates of the projector from the ground truth images by $\mathbf{X}_c$, we can solve the simpler problem given by

$$\underset{\mathbf{v}}{\operatorname{argmin}} \sum_{c \in \mathcal{C}} \left\| \mathbf{X}_c - \widetilde{\mathbf{X}}_c(\mathbf{v}) \right\|_F^2, \tag{9}$$

and use the solution as an initial guess for our problem in Equation 1. The method used to recover $\mathbf{X}_c$ depends on the patterns displayed, but for two sets of phase-shifted patterns, the heterodyne principle can be used [21].

To make the initial problem simpler, we start with a mesh that has few vertices and gradually increase the number of vertices by remeshing. This enables us to use a simple shape, e.g., a sphere as our initial mesh.

3.3.3. Remeshing

As described in Sections 3.3.1 and 3.3.2 we use a remeshing algorithm. The algorithm we use is adapted from the Python geometry processing library Pymesh [22]. The outline of the algorithm is shown in Algorithm 1.

Algorithm 1: Remeshing algorithm adapted from [22]

Input: Mesh, target edge length ℓ
Result: Mesh
Remove degenerate triangles
Split edges longer than ℓ
while *mesh is changed* **do**
    Collapse edges shorter than ℓ
    Split obtuse triangles (angle bigger than 150°)
Remove self-intersections
Remove duplicate faces
Replace mesh with outer hull of mesh
Remove duplicated vertices
Split obtuse triangles (angle bigger than 179°)
Remove isolated vertices

Figure 2. Crop of image shown with varying levels of noise. From left to right: $k = 1$, $k = 10^2$, $k = 10^3$.

4. Experiments

To demonstrate the usefulness of our method, we have carried out a few experiments, that we will briefly introduce. First, we show that our method can reconstruct an object starting from a sphere, see Figure 1. Secondly, we reconstruct three different objects at multiple levels of noise, see Figure 2. Finally, we compare against a Poisson reconstruction at two levels of noise and show that our method can reconstruct sharp edges.

4.1. Generating ground truth images

We did all our experiments using synthetic ground truth images, to have access to the ground truth shape of the mesh to compare against. Our ground truth images are made by projecting two sets of phase-shifted patterns with 15 and 16 periods respectively, with 16 shifts of the first pattern and 8 shifts of the second, such that

$$n_p = \begin{cases} 15 & p \in [1, 2, \ldots, 16] \\ 16 & p \in [17, 18, \ldots, 24] \end{cases} \tag{10}$$

and

$$\phi_p = \begin{cases} 2\pi p \frac{1}{16} & p \in [1, 2, \ldots, 16] \\ 2\pi (p - 16) \frac{1}{8} & p \in [17, 18, \ldots, 24]. \end{cases} \tag{11}$$

4.1.1. Rendering

We render the ground truth images using ray tracing with 100 samples per pixel for anti-aliasing, and we use the Lambertian reflectance model to describe the optical properties of our objects of interest.

4.1.2. Noise

As these ground truth images are noise-free, we add noise to make the images more realistic. For this, we model the noise of a pixel with intensity x by a Gaussian distribution with mean x and variance

$$\sigma^2 = \sigma_r^2 + x\sigma_p^2, \tag{12}$$

where the first term σ_r describes the signal-independent sensor read-out noise and the second term $x\sigma_p$ describes the signal-dependent shot noise. We choose σ_r and σ_p by using the noise levels from a baseline camera [23]. Our noise levels are then defined as multiples of this baseline noise level, controlled by k as follows

$$\sigma^2(x, k) = k(4.5 \cdot 10^{-7} + x \cdot 2 \cdot 10^{-5}), \tag{13}$$

such that $k = 1$ gives the noise levels of a baseline camera for $x \in [0, 1]$. After adding noise, we clamp pixel values to the $[0, 1]$ range. Examples of images for different values of k are in Figure 2.

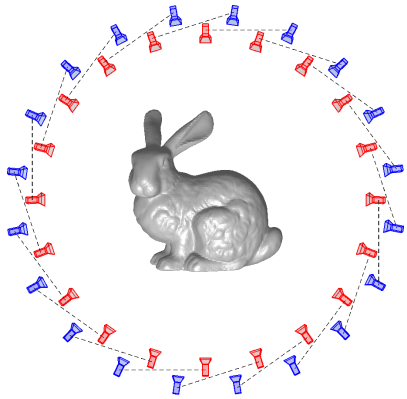


Figure 3. Visualization of one of the circles from our camera-projector setup. In all experiments we use 3 circles with 20 cameras each. Red objects indicate a camera and blue indicates a projector. The dotted lines show which cameras and projectors belong together.

4.2. Quantitative evaluation

We quantitatively evaluate the performance of our method by measuring the symmetric volume difference between the ground truth and a reconstruction as a percentage of the volume of the ground truth. We refer to this as Δ_V and compute it as

$$\Delta_V = \frac{\left|(S \setminus \tilde{S}) \cup (\tilde{S} \setminus S)\right|}{|S|}, \tag{14}$$

where S and $\tilde{S}$ are the ground truth and reconstruction considered as solids, and $|\cdot|$ is the volume of a solid.

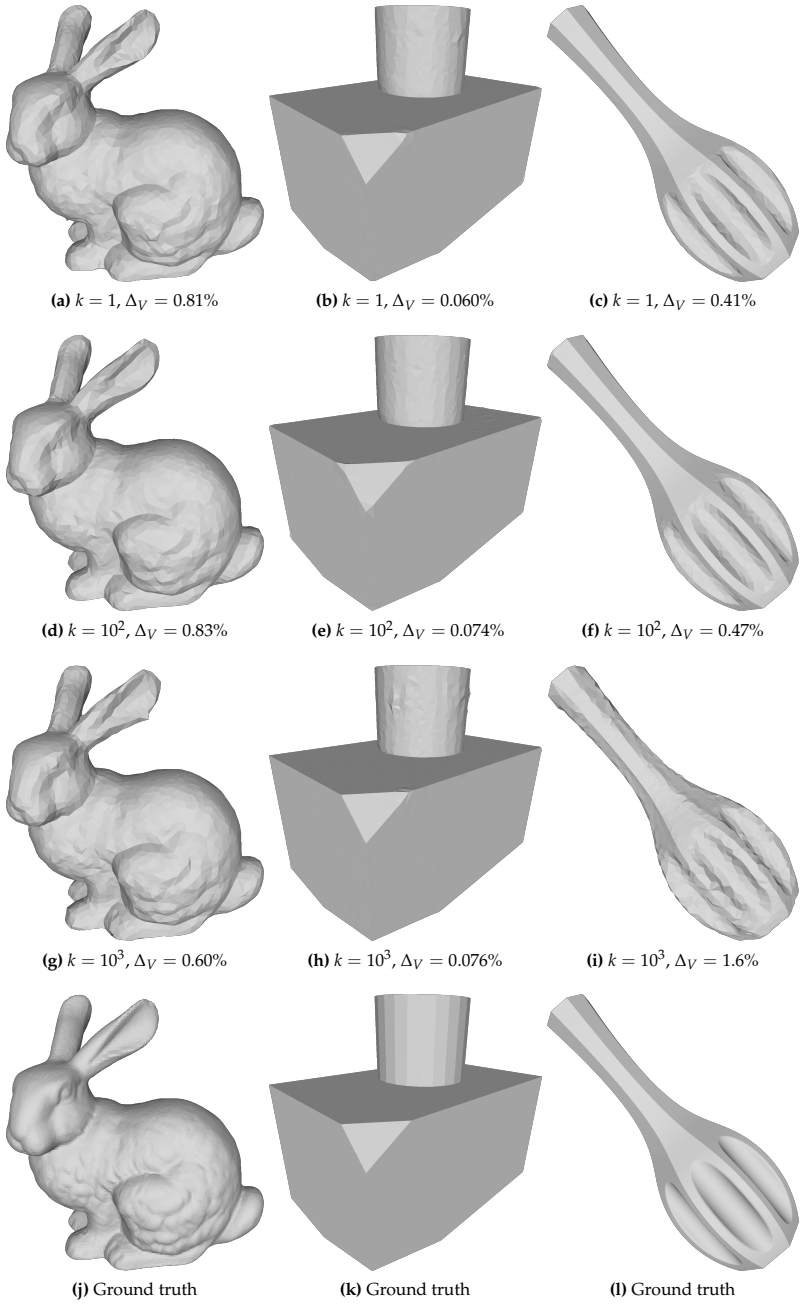
4.3. Experiment details

We evaluate our method on three different shapes. The Stanford Bunny [3], a combination of a cylinder and a box with various truncated corners, and a dandelion vase [24]. These shapes are in the bottom row of Figure 4. When starting from a simple shape as described in Section 3.3.2, we have used the same sphere across all our experiments. When optimizing the simpler problem in Equation 9, we did remeshing for every 25^{th} iteration. At each remeshing step, we decrease the target edge length which yields meshes with finer and finer resolution. For the i^{th} remeshing step we set the target to

$$\ell_i = 0.99^i \cdot 0.025 \cdot d_{BB}, \tag{15}$$

where d_{BB} is the largest diagonal of the bounding box of the current mesh. To be able to solve the problem in Equation 9, we have estimated $\mathbf{X}_c$ using the heterodyne principle [21]. In all our experiments, we have used 60 camera-projector positions organized in 3 circles, with 20 in each circle, to mimic a structured light scanner with the object placed in three different poses on a turntable rotating 18° between each image. One of these circles is visualized in Figure 3. The camera resolution for all renderings is 1920×1080 pixels.

In Figure 1, we show how our method can reconstruct the Stanford Bunny starting from a sphere. The reconstruction was done for $k = 1$. After finishing the optimization based on Equation 9, we remesh the result by halving the target edge length ℓ from the last remeshing step, to get a mesh with even finer resolution. The two in-progress images shown, are during the initial optimization directly on $\mathbf{X}_c$. The final reconstruction contains many of the fine details of the true bunny (Figure 4j).



(a) $k = 1$, $\Delta_V = 0.81\%$ **(b)** $k = 1$, $\Delta_V = 0.060\%$ **(c)** $k = 1$, $\Delta_V = 0.41\%$

(d) $k = 10^2$, $\Delta_V = 0.83\%$ **(e)** $k = 10^2$, $\Delta_V = 0.074\%$ **(f)** $k = 10^2$, $\Delta_V = 0.47\%$

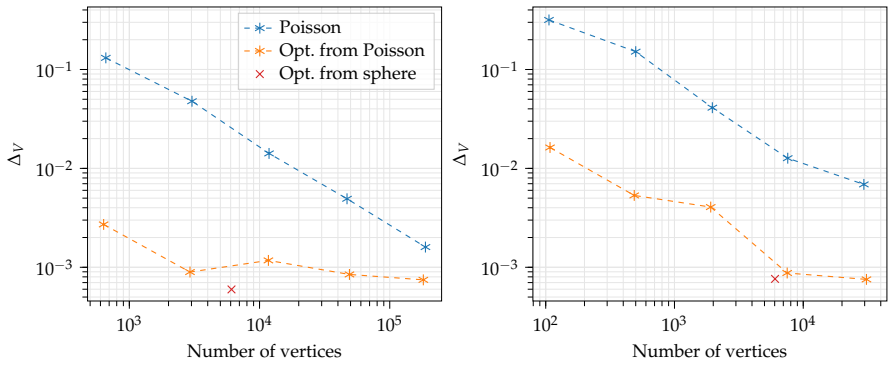
(g) $k = 10^3$, $\Delta_V = 0.60\%$ **(h)** $k = 10^3$, $\Delta_V = 0.076\%$ **(i)** $k = 10^3$, $\Delta_V = 1.6\%$

(j) Ground truth **(k)** Ground truth **(l)** Ground truth

Figure 4. Reconstructions made by our method on three different objects, for increasing levels of noise k.

	Bunny	Box	Vase
Reconstructions	3 350	6 000	2 300
Ground truth	34 817	4 619	44 852

Table 1. Approximate number of vertices of each of the objects in Figure 4.



(a) Low image noise ($k = 1$) **(b)** High image noise ($k = 10^3$)

Figure 5. Comparison showing Δ_V as a function of the number of vertices in the mesh for reconstructions of the box with cylinder. Poisson is the Poisson reconstruction, where only the connected component with the largest volume has been kept, as the Poisson reconstruction sometimes produces multiple connected components for high levels of noise. Opt. from Poisson and sphere is our method using the initial guesses described in Section 3.3. The Poisson reconstruction was done for spatial octree depths of 4 to 8.

To examine how our method is affected by noise, we reconstruct all three objects starting from a sphere with varying levels of image noise ($k \in [1, 10^2, 10^3]$), which we show in Figure 4. We see that our method is largely unaffected by noise and is still able to reconstruct the shape. The details lacking in our reconstructions are mostly due to us not having made the mesh fine enough to represent these details, which can also be seen in Table 1.

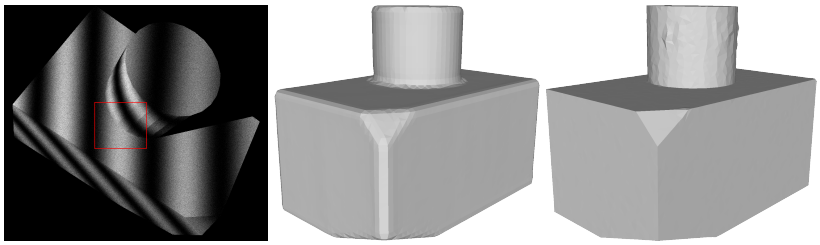
Finally, we compare against Screened Poisson reconstruction in Figure 5, for multiple depths of the reconstruction. The point cloud used for the Poisson reconstruction for $k = 1$ contains 22 million points and the one for $k = 10^3$ contains 17 million points. Our method consistently achieves a lower error than the Poisson reconstruction. The meshes from some of these data points are shown in Figure 6.

5. Discussion

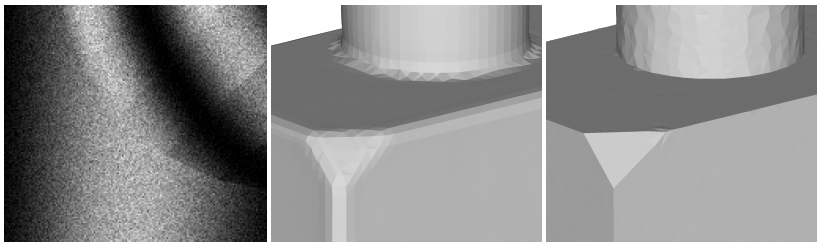
Our method has the same implicit assumption about the appearance of the object that is necessary for structured light. This assumption is that the amount of global light in each pixel, i.e., the light that is not directly reflected, is the same for all patterns. This is approximately true for high-frequency sinusoidal patterns [25].

Although we have used the Lambertian reflectance model to render our synthetic data, we do not expect this to be a limitation of our method. It does not rely on the assumption of Lambertian reflectance due to the use of structured light, and therefore does not rely on estimating a texture map of the object, neither implicitly nor explicitly.

In our experiments, we have had the advantage of knowing the camera and projector positions exactly, which is not the case when working with real data. This can however be remedied relatively easily, by including the positions of these as parameters in the optimization problem. Additionally,



(a) A noisy input image. **(b)** Poisson reconstruction with 7 064 vertices. $\Delta_V = 1.28\%$ **(c)** Our method with 6 038 vertices. $\Delta_V = 0.08\%$



(d) Crop of noisy image. **(e)** Close-up of Poisson reconstruction. **(f)** Close-up of our method.

Figure 6. Our method can reconstruct meshes with sharp edges from very noisy images ($k = 10^3$). Our method has a lower error (Δ_V) than a Screened Poisson reconstruction with a comparable number of vertices (spatial octree depth 7).

we do not need to purely rely on our estimates of $\mathbf{A}_c$, $\mathbf{B}_c$, as these can also be optimized for. We do however expect both $\mathbf{A}_c$, $\mathbf{B}_c$, and the camera-projector positions to be known quite accurately and would suggest letting them be part of the optimization only once the original optimization problem has converged.

In order to solve the optimization problem, we compute the derivative of our loss function, which involves the derivative of our rendering. This is potentially problematic as our rendering is not differentiable at depth discontinuities, i.e., where non-neighbouring faces of the mesh are bordering each other in the image space. This could e.g., be the ear and body of the Stanford Bunny. Our method will in this case experience aliasing error in the renderings and derivatives. However, since the faces are observed from multiple views simultaneously, there is often another view where the same edges are observed with continuous depth. The problem is thus mitigated and since it occurs only for a small percentage of the pixels for each iteration it was not a problem in our optimization experiments.

A disadvantage of our method is that it is not able to handle topology changes. In practice, this means, that the initial mesh must have the same topology as the true object.

The choice of using a mesh as the surface representation to optimize has some advantages. It is very efficient to compute the gradient of our loss function for a mesh, as each pixel only influences a constant number of elements in the gradient, which makes it suitable for parallel implementation on a GPU (Graphics Processing Unit).

6. Conclusion

Our primary contribution is a novel model for computing a surface mesh directly from structured light images without the need for an intermediate point cloud. With this model, we have shown that the direct computation of a triangle mesh gives higher accuracy and is particularly good at reconstructing sharp features such as corners and edges, which are smoothed out using the two-step Screened Poisson reconstruction that we compare with. Further, we have obtained very high robustness to noise by

optimizing the vertex positions directly from the images. Finally, our approach completely avoids partial scans that must subsequently be aligned, because the optimization is done for all images at once. This demonstrates the advantage of directly reconstructing a surface from structured light images using differentiable rendering.

Author Contributions: Conceptualization, J.W., J.F.and A.D.; methodology, J.N., M.H.and J.B.; software, J.N.and M.H.; investigation, J.N.and M.H.; visualization, J.N.; supervision, A.D., J.B.and J.F.; writing–original draft preparation, J.N., M.H., J.F.and A.D.; writing–review and editing, all authors. All authors have read and agreed to the published version of the manuscript.

Funding: The work presented in this paper is part of the research project "FlexDraperProduct: Developing an integrated production-ready solution for automated layup of prepregfiber plies", funded by Innovation Fund Denmark (Grant No. 8057-00011B). The support is gratefully acknowledged.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A. Derivation of Equation 6

In this appendix, we derive the gradients of the elements of $\widetilde{\mathbf{X}}_c(\mathbf{v})$ with respect to the vertices $\mathbf{v}$. Let $\widetilde{X}_c^{i,j}$ be the i,j^{th} element of $\widetilde{\mathbf{X}}_c(\mathbf{v})$. This is found by tracing a ray through the center of the i,j^{th} pixel in the camera until it intersects the surface at a point $\mathbf{r}$, that is

$$\mathbf{r} = \mathbf{o} + t\mathbf{d}, \tag{A1}$$

for some t, where $\mathbf{o}$ is the position of the camera and $\mathbf{d}$ is the direction of the ray. Modelling the projector as a pinhole camera, this point is projected back into the projector by

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix}. \tag{A2}$$

If we define $\mathbf{p}_1 = \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} \end{bmatrix}$ and $\mathbf{p}_3 = \begin{bmatrix} P_{3,1} & P_{3,2} & P_{3,3} \end{bmatrix}$ then

$$\widetilde{X}_c^{i,j} = \frac{q_1}{q_3} = \frac{\mathbf{p}_1 \cdot \mathbf{r} + P_{1,4}}{\mathbf{p}_3 \cdot \mathbf{r} + P_{3,4}}. \tag{A3}$$

The point $\mathbf{r}$ intersect a triangle with vertices $\mathbf{p}_a, \mathbf{p}_b$, and $\mathbf{p}_c$. The normal of this triangle can be computed by

$$\mathbf{n} = (\mathbf{p}_c - \mathbf{p}_b) \times (\mathbf{p}_a - \mathbf{p}_c). \tag{A4}$$

Using the normal we can write t as

$$t = \frac{(\mathbf{p}_b - \mathbf{o}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}. \tag{A5}$$

Using the chain-rule we have that

$$\frac{\partial \widetilde{X}_c^{i,j}}{\partial \mathbf{p}_a} = \frac{\partial \widetilde{X}_c^{i,j}}{\partial t} \frac{\partial t}{\partial \mathbf{p}_a} \tag{A6}$$

$$= \left(\sum_{i=1}^{3} \frac{\partial \widetilde{X}_c^{i,j}}{\partial p_i} \frac{\partial p_i}{\partial t} \right) \frac{\partial t}{\partial \mathbf{p}_a} \tag{A7}$$

$$= \left(\frac{\partial \widetilde{X}_c^{i,j}}{\partial \mathbf{r}} \cdot \mathbf{d} \right) \frac{\partial t}{\partial \mathbf{p}_a}. \tag{A8}$$

Here

$$\frac{\partial \widetilde{X}_c^{i,j}}{\partial \mathbf{r}} = \frac{\frac{\partial q_1}{\partial \mathbf{r}} q_3 - q_1 \frac{\partial q_3}{\partial \mathbf{r}}}{{q_3}^2} \tag{A9}$$

$$= \frac{\frac{\partial q_1}{\partial \mathbf{r}} - \widetilde{X}_c^{i,j} \frac{\partial q_3}{\partial \mathbf{r}}}{q_3} \tag{A10}$$

$$= \frac{\mathbf{p}_1 - \widetilde{X}_c^{i,j} \mathbf{p}_3}{q_3}. \tag{A11}$$

The last missing term in Equation (A8) is

$$\frac{\partial t}{\partial \mathbf{p}_a} = \frac{\partial}{\partial \mathbf{p}_a} \frac{(\mathbf{p}_b - \mathbf{o}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \tag{A12}$$

$$= \frac{\frac{\partial}{\partial \mathbf{p}_a} \left[(\mathbf{p}_b - \mathbf{o}) \cdot \mathbf{n} \right] (\mathbf{d} \cdot \mathbf{n}) - \left[(\mathbf{p}_b - \mathbf{o}) \cdot \mathbf{n} \right] \frac{\partial \mathbf{d} \cdot \mathbf{n}}{\partial \mathbf{p}_a}}{(\mathbf{d} \cdot \mathbf{n})^2} \tag{A13}$$

$$= \frac{\frac{\partial \mathbf{n}}{\partial \mathbf{p}_a}^\top (\mathbf{p}_b - \mathbf{o}) - t \frac{\partial \mathbf{n}}{\partial \mathbf{p}_a}^\top \mathbf{d}}{\mathbf{d} \cdot \mathbf{n}} \tag{A14}$$

$$= \frac{\frac{\partial \mathbf{n}}{\partial \mathbf{p}_a}^\top (\mathbf{p}_b - \mathbf{o} - t\mathbf{d})}{\mathbf{d} \cdot \mathbf{n}} \tag{A15}$$

$$= \frac{\frac{\partial \mathbf{n}}{\partial \mathbf{p}_a}^\top (\mathbf{p}_b - \mathbf{r})}{\mathbf{d} \cdot \mathbf{n}}, \tag{A16}$$

with

$$\frac{\partial \mathbf{n}}{\partial \mathbf{p}_a} = \frac{\partial}{\partial \mathbf{p}_a} (\mathbf{p}_c - \mathbf{p}_b) \times (\mathbf{p}_a - \mathbf{p}_c) \tag{A17}$$

$$= \frac{\partial}{\partial \mathbf{p}_a} (\mathbf{p}_c - \mathbf{p}_b) \times \mathbf{p}_a \tag{A18}$$

$$= \frac{\partial}{\partial \mathbf{p}_a} [\mathbf{p}_c - \mathbf{p}_b]_\times \mathbf{p}_a \tag{A19}$$

$$= [\mathbf{p}_c - \mathbf{p}_b]_\times, \tag{A20}$$

where we utilise that a cross-product $\mathbf{a} \times \mathbf{b}$ can be written as a matrix-vector product $[\mathbf{a}]_\times \mathbf{b}$ using a skew-symmetric matrix. Inserting into Equation (A16) we get

$$\frac{\partial t}{\partial \mathbf{p}_a} = \frac{(\mathbf{p}_b - \mathbf{r}) \times (\mathbf{p}_c - \mathbf{p}_b)}{\mathbf{d} \cdot \mathbf{n}}. \tag{A21}$$

If we write $\mathbf{r}$ using barycentric coordinates, $\mathbf{r} = \lambda_a \mathbf{p}_a + \lambda_b \mathbf{p}_b + \lambda_c \mathbf{p}_c$ with $\lambda_a + \lambda_b + \lambda_c = 1$, then Equation (A21) reduces to

$$\begin{aligned}
\frac{\partial t}{\partial \mathbf{p}_a} &= \frac{(\mathbf{p}_b - (\lambda_a \mathbf{p}_a + \lambda_b \mathbf{p}_b + \lambda_c \mathbf{p}_c)) \times (\mathbf{p}_c - \mathbf{p}_b)}{\mathbf{d} \cdot \mathbf{n}} \\
&= \frac{(\lambda_a (\mathbf{p}_c - \mathbf{p}_a) + (1 - \lambda_b)(\mathbf{p}_b - \mathbf{p}_c)) \times (\mathbf{p}_c - \mathbf{p}_b)}{\mathbf{d} \cdot \mathbf{n}} \\
&= \frac{\lambda_a (\mathbf{p}_c - \mathbf{p}_a) \times (\mathbf{p}_c - \mathbf{p}_b)}{\mathbf{d} \cdot \mathbf{n}} \\
&= \frac{\lambda_a \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}.
\end{aligned} \tag{A22}$$

Putting it all together, we have (Equation 6)

$$\frac{\partial \widetilde{X}_c^{i,j}}{\partial \mathbf{p}_a} = \left(\frac{\mathbf{p}_1 - \widetilde{X}_c^{i,j} \mathbf{p}_3}{q_3} \cdot \mathbf{d} \right) \frac{\lambda_a \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}. \tag{A23}$$

The derivatives with respect to $\mathbf{p}_b$, and $\mathbf{p}_c$ can be found with similar derivations.

References

1. Gomes, L.; Bellon, O.R.P.; Silva, L. 3D reconstruction methods for digital preservation of cultural heritage: A survey. *Pattern Recognition Letters* **2014**, *50*, 3–14.
2. Nocerino, E.; Stathopoulou, E.K.; Rigon, S.; Remondino, F. Surface reconstruction assessment in photogrammetric applications. *Sensors* **2020**, *20*, 5863.
3. Turk, G. The Stanford Bunny. https://www.cc.gatech.edu/~turk/bunny/bunny.html, 2000.
4. Berger, M.; Tagliasacchi, A.; Seversky, L.M.; Alliez, P.; Guennebaud, G.; Levine, J.A.; Sharf, A.; Silva, C.T. A survey of surface reconstruction from point clouds **2017**. *36*, 301–329.
5. Horn, B.K.P. Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object From One View. PhD thesis, Massachusetts Institute of Technology, 1970. MAC TR-79.
6. Horn, B.K.P. Obtaining Shape from Shading Information. In *The Psychology of Computer Vision*; Winston, P.H., Ed.; McGraw-Hill, 1975; chapter 4, pp. 115–155.
7. Rockwood, A.P.; Winget, J. Three-dimensional object reconstruction from two-dimensional images. *Computer-Aided Design* **1997**, *29*, 279–285.
8. Zhang, L.; Seitz, S.M. Image-based multiresolution shape recovery by surface deformation. Videometrics and Optical Methods for 3D Shape Measurement, 2000, Vol. 4309, *Proceedings of SPIE*, pp. 51–61.
9. Inagaki, A.; Tagawa, N.; Minagawa, A.; Moriya, T. Computation of shape and reflectance of 3D object using moire/spl acute/phase and reflection model. International Conference on Image Processing (ICIP 2001). IEEE, 2001, Vol. 2, pp. 177–180.
10. Naganuma, S.; Tagawa, N.; Minagawa, A. Estimation of 3D shape and reflectance using multiple moiré images and shading model. ACM Symposium on Applied Computing (SAC 2003), 2003, pp. 943–950.
11. Naganuma, S.; Tagawa, N.; Minagawa, A.; Moriya, T. Simultaneous determination of object shape and color by Moire analysis using a reflection model. International Conference on Pattern Recognition (ICPR 2004). IEEE, 2004, Vol. 3, pp. 202–205.
12. Isidoro, J.; Sclaroff, S. Stochastic mesh-based multiview reconstruction. First International Symposium on 3D Data Processing Visualization and Transmission. IEEE, 2002, pp. 568–577.
13. Isidoro, J.; Sclaroff, S. Stochastic Refinement of the Visual Hull to Satisfy Photometric and Silhouette Consistency Constraints. International Conference on Computer Vision (ICCV 2003). IEEE, 2003, Vol. 2, pp. 1335–1342.
14. Yu, T.; Xu, N.; Ahuja, N. Shape and view independent reflectance map from multiple views. European Conference on Computer Vision (ECCV 2004). Springer, 2004, pp. 602–615.
15. Yu, T.; Xu, N.; Ahuja, N. Shape and view independent reflectance map from multiple views. *International journal of computer vision* **2007**, *73*, 123–138.
16. Wu, C.; Wilburn, B.; Matsushita, Y.; Theobalt, C. High-quality shape from multi-view stereo and shading under general illumination. Conference on Computer Vision and Pattern Recognition (CVPR 2011). IEEE, 2011, pp. 969–976.
17. Ha, H.; Oh, T.H.; Kweon, I.S. A multi-view structured-light system for highly accurate 3D modeling. International Conference on 3D Vision (3DV 2015). IEEE, 2015, pp. 118–126.
18. Loubet, G.; Holzschuch, N.; Jakob, W. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on Graphics* **2019**, *38*, 228:1–228:14.
19. Liu, S.; Li, T.; Chen, W.; Li, H. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. International Conference on Computer Vision (ICCV 2019), 2019, pp. 7708–7717.
20. Kazhdan, M.; Hoppe, H. Screened poisson surface reconstruction. *ACM Transactions on Graphics* **2013**, *32*, 1–13.
21. Reich, C.; Ritter, R.; Thesing, J. White light heterodyne principle for 3D-measurement. Sensors, Sensor Systems, and Sensor Data Processing; Loffeld, O., Ed. SPIE, 1997, Vol. 3100, pp. 236 – 244. doi:10.1117/12.287750.
22. Zhou, Q. PyMesh—Geometry Processing Library for Python. *Software available for download at https://github.com/PyMesh/PyMesh* **2020**.
23. Adobe Systems Incorporated. *Ditial Negative (DNG) Specification*, 2019.
24. Steve. Dandelion Vase. www.thingiverse.com/thing:157102, 2013.

25. Nayar, S.K.; Krishnan, G.; Grossberg, M.D.; Raskar, R. Fast separation of direct and global components of a scene using high frequency illumination. In *ACM SIGGRAPH 2006 Papers*; 2006; pp. 935–944.