UNIVERSITY OF CALIFORNIA,
IRVINE

Spatially Augmented Reality on Dynamic, Deformable Surfaces and its Applications

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Muhammad Twaha Ibrahim

Dissertation Committee:
Professor Aditi Majumder, Chair
Professor Gopi Meenakshisundaram
Associate Professor Shuang Zhao
Professor Raj M. Vyas

2024

DEDICATION

To my Creator and the best of His Creation

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I would like to thank AFWERX SBIR Subcontract for funding my research.

I would also like to thank the Link Fellowship for providing me funding during the final year of my program.

I would like to thank IEEE, ACM and Elsevier for granting me permission to reuse previously published work in this thesis. Portion of Chapter 2 is a reprint of [39, 41] with permission from ACM and Elsevier. Portions of Chapters 3 and 4 are reprints of [33, 34] with permission from IEEE. Portion of Chapter 5 is a reprint of [35] with permission from IEEE. Portion of Chapter 6 is a reprint of [36, 40] with permission from IEEE and Elsevier.

There are many people I would like to thank.

First of all, my advisor Aditi Majumder, for teaching me many things, including performing rigorous research, presenting my work and many important life lessons. I am also grateful to her for her support, guidance and mentorship throughout these years.

I am also deeply grateful to my co-advisor, Gopi Meenakshisundaram who, like Aditi, has played a very important role in my research, providing guidance and discussion on challenging problems and being very supportive throughout my studies.

I am grateful to Dr. Raj M. Vyas and Dr. Lohrasb R. Sayadi for their guidance in applying my research to solve meaningful and impactful problems.

I am also grateful to Dr. Shuang Zhao for being on my dissertation committee.

I am also very grateful to the team at Summit Technology Laboratory for their assistance, guidance and support, especially Stephen Clark, Bharati Jha, Alexander "Sasha" Sidenko, Meghana Urs, Ranga Sai Shreyas Manchikanti, Abhishek Nalawade and Sarvesh Thakur.

I am also grateful to my friends in the iGravi Lab.

I am also grateful for my friends in and around Irvine and the world.

I am deeply grateful to my family for always being there for me: my brother Awais for always having my back, my brother Aalamgeer for always pushing me to do better, and my sister Noorya for always listening to me. I am also grateful to my siblings-in-law, Ahsan, Kanza and Maheen for always being a source of support; my beloved nieces and nephews who bring so much joy in my life. And of course, I am grateful to my parents: my mother, who among many things, always reminded me that *"This time too shall pass"*; and my late father, who was the main source of inspiration for me to pursue graduate studies.

Finally, more than anyone else, I am grateful to Allah (God) Al-Mighty. It is He who has created me, gave me a loving family, wonderful friends and sincere advisors and mentors. It is He who gave me the ability to read and write, and research the world He created.

If I tried to count all His blessings on me, I would never be able to number them.

To Him we belong, and to Him we shall all return.

VITA

Muhammad Twaha Ibrahim

EDUCATION

Doctor of Philosophy in Computer Science **2024**
University of California Irvine *Irvine, CA, USA*

Master of Engineering in Electrical and Computer Engineering **2014**
Cornell University *Ithaca, NY, USA*

Bachelor of Science in Computational Sciences **2011**
National University of Science and Technology *Rawalpindi, Pakistan*

RESEARCH EXPERIENCE

Computer Vision Researcher **2020–2024**
Summit Technology Laboratory *Irvine, California*

Graduate Research Assistant **2017–2024**
University of California, Irvine *Irvine, California*

Research Intern **2018**
Google LLC. *Venice, California*

Computer Vision Researcher **2014–2015**
Futurewei Technologies Inc. *Bridgewater, New Jersey*

TEACHING EXPERIENCE

Teaching Assistant **2017–2024**
University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

Illuminating Precise Surgical Stencils using Projection-based Augmented Reality — **2024**
Elsevier SmartHealth Journal

Real-time Seamless Multi-Projector Displays on Deformable Surfaces — **2024**
IEEE Transactions on Visualization and Computer Graphics

3D Gamut Morphing for Non-Rectangular Multi-Projector Displays — **2023**
IEEE Transactions on Visualization and Computer Graphics

Dynamic projection mapping on deformable stretchable materials using boundary tracking — **2022**
Elsevier Computers and Graphics

REFEREED CONFERENCE PUBLICATIONS

Self-Calibrating Dynamic Projection Mapping System for Dynamic, Deformable Surfaces with Jitter Correction and Occlusion Handling — **Oct 2023**
IEEE International Symposium on Mixed and Augmented Reality

Automated Landmark Detection for AR Based Craniofacial Surgical Assistance System — **Jul 2023**
International Conference on Artificial Intelligence and Virtual Reality

Projector-Camera Calibration on Dynamic, Deformable Surfaces — **Mar 2023**
IEEE Conference on Virtual Reality and 3D User Interfaces

Projector Illuminated Precise Stencils on Surgical Sites — **Mar 2023**
IEEE Conference on Virtual Reality and 3D User Interfaces

Augmented Reality Patient-Specific Registration for Medical Visualization — **Nov 2022**
ACM Symposium on Virtual Reality Software and Technology

Ambient Light Tolerant Laser-Pen Based Interaction with Curved Multi-Projector Displays — **Jun 2022**
HCI International Conference

Dynamic Projection Mapping of Deformable Stretchable Materials — **Nov 2022**
ACM Symposium on Virtual Reality Software and Technology

ABSTRACT OF THE DISSERTATION

Spatially Augmented Reality on Dynamic, Deformable Surfaces and its Applications

By

Muhammad Twaha Ibrahim

Doctor of Philosophy in Computer Science

University of California, Irvine, 2024

Professor Aditi Majumder, Chair

Spatially Augmented Reality (SAR), also known as projection mapping, uses multiple projectors to illuminate surfaces of arbitrary shape and size and create seamless, large-scale displays. Traditional SAR assumes that the projection surface is static and rigid. This restriction was partially addressed by Dynamic-SAR, where the surface is rigid and of known shape but can be moved around. However, no prior work has addressed SAR using multiple projectors on deformable surfaces, where the shape is unknown and constantly changing. Thus, multi-projector SAR on deformable surfaces introduces several challenges, including projector-camera calibration on a deformable surface, real-time surface shape recovery and real-time multi-projector warping and blending. My thesis is the first attempt to develop a comprehensive framework for achieving seamless multi-projector displays on deformable surfaces. Furthermore, I will also be presenting its applications in the medical domain to enable remote surgical guidance by using SAR to illuminate surgical stencils on a physical surgical site precisely.

Chapter 1

Introduction

Spatially Augmented Reality (SAR), also known as projection mapping, offers a unique way to augment the real-world with virtual data, modifying the appearance of physical objects using digital imagery. Unlike Augmented Reality and Virtual Reality (AR/VR) systems that require users to wear cumbersome headsets to provide an individual experience, SAR eliminates the need for any wearables by transforming the visual appearance of surfaces, providing a shared experience for all viewers. Thus, SAR systems enable novel applications such as surgical guidance, visualization, design and entertainment. Traditionally, SAR has been performed on static and rigid surfaces using one or more projectors and cameras, and is known as static SAR. It requires a one-time calibration step, and is followed by a warp-and-blend correction step that combines all the projectors into one unified and seamless display. However, static SAR systems have a major limitation: if the surface moves, the entire system has to be disrupted for recalibration.

This limitation gave rise to Dynamic SAR (D-SAR) systems, where the surface may be dynamic. The dynamic nature of the surface can mean the following, mentioned in order of increasing complexity: **(i)** the surface may be a rigid body, but can be moved around and

rotated, **(ii)** the surface could be deformable, like a sheet of paper, and **(iii)**, the surface can additionally be stretchable, like an elastic fabric. The goal is to project on such surfaces, whether rigid, deformable and/or stretchable, such that the projection continuously conforms to the changing surface shape, appearing as if it is part of the object. Thus, unlike SAR on static objects, SAR on dynamic and deformable objects presents several challenges, including accurate and high-speed reconstruction of the surface geometry, high-speed parameterization of the display surface and low-latency adaptive projection.

To meet the above challenges in realtime, D-SAR systems typically employ RGB-D cameras to assist with the shape recovery process, which in addition to RGB images, provide realtime depth of a scene. The depth map from the RGB-D camera is used to recover the surface shape, which in turn is used to warp the display content accordingly based on the system calibration parameters. The projector then projects this image on the display surface making it appear conformal to the surface shape.

In order to achieve scale for a more immersive experience, SAR systems use multiple projectors to illuminate large surfaces. However, using multiple projectors on deformable surfaces introduce additional challenges, such as realtime surface reconstruction from multiple cameras and realtime multi-projector blending. Below, we list the main challenges for achieving a realtime, seamless multi-projector display on defomrable surfaces using SAR.

1.1 Main Challenges

1. **Accurate Geometric Calibration**: Geometric calibration in the context of SAR systems entails recovering the parameters of all devices i.e. projectors and cameras, being used. The parameters consist of the intrinsics i.e. focal length, principal point and distortion coefficients, and the extrinsics i.e. the rotation and translation of the device with respect to the world origin. Accurate calibration of devices is crucial for

SAR to achieve a seamless display; errors in calibration manifest as misalignments in the projector overlapping regions, breaking the seamlessness of the display. Current SAR calibration methods assume the surface to be static and rigid and therefore, cannot be used for a deformable surface. Furthermore, these methods may return calibration parameters that do not match the physical arrangement of the devices even if they work for that particular 3D arrangement of devices and surface. However, D-SAR requires the calibration parameters to match the physical arrangement of the devices otherwise misalignments in the projections become clearly visible when the surface moves. Therefore, we require a method to accurately calibrate the multi-projector, multi-camera rig which does not require a rigid and static surface.

2. **Real-time Surface Reconstruction**: Achieving large-scale displays requires the D-SAR system to use multiple cameras, each observing only part of the surface. Therefore, each camera provides the depth of only part of the surface. For a multi-projector D-SAR system, the depth from all the cameras must be combined in real-time to recover the full surface 3D shape accurately. This 3D shape is then used for geometric correction of the display content.

3. **Real-time Content Warp**: The 3D shape recovered and the geometric calibration from the previous steps is then used for warping the display content for every projector. Since the surface shape is constantly changing, the warping also has to change and therefore, must be performed in realtime.

4. **Real-time Projector Blending**: When multiple projectors overlap, the overlapping regions becomes much brighter. Even with geometric correction that aligns the content in the overlapping regions, the viewer will see brightness seams in the overlapping regions. In order to remove these brightness seams for a seamless display, the overlapping projectors are blended by gradually dimming the brightness contribution at every surface point from each projector. Static SAR systems only need to perform this

blending once based on the surface shape. However, D-SAR systems require blending to be performed continuously based on the changing surface shape. This is because the deforming surface shape changes the shape of the overlapping region of the projectors. Additionally, the blending algorithm must be fast and efficient for realtime implementation.

5. **Quick and Automated Setup**: Manually correcting, warping and blending projectors for a seamless multi-projector display is very cumbersome and time consuming and can take several days for perfect alignment of projectors in a static SAR system. Thus, automating the entire SAR pipeline i.e. calibration, warping and blending, has been one of the main focuses of SAR research. For a dynamic SAR system, manual correction, warp and blend is not possible since the surface shape is changing in realtime. We want a system that completely takes away the need for any manual intervention at any step in the D-SAR system and does not require a trained user to set up the display. Ideally, a lay user should be able to casually place the projectors and cameras and with the push of a button, be able to calibrate, warp and blend a multi-projector display on a deformable surface.

In this thesis, we address all the above challenges to achieve a seamless, realtime multi-projector display on deformable surfaces. In Chapter-2, we explain a method for achieving SAR on a deformable surface using a single projector-camera pair. In Chapter-3, we address the problem of geometric calibration for D-SAR by proposing a counterintuitive method that leverages the dynamic nature of the surface to calibrate a single projector-camera pair.

Depth cameras used in D-SAR systems show a lot of noise in the depth map. This manifests as a jittery projection which is unpleasant to view and can leave viewers fatigued. Furthermore, any objects that may occlude the projection surface cause the display to glitch. In Chapter-4, we discuss solutions for both these problems to achieve a practical and deployable D-SAR system.

In Chapter-5, we extend the methods in the previous chapters and propose a scalable method to achieve D-SAR on deformable surfaces using multiple projectors. This includes accurate geometric calibration of a multi-camera, multi-projector system, a realtime surface reconstruction method and a realtime multi-projector blending method.

Finally, in Chapter-6, we present an application of our research in D-SAR for remote surgical guidance using projectors to illuminate surgical stencils on patients.

Chapter 2

Single PDC Deformable-SAR

In this chapter, we propose a general framework for efficiently performing dynamic projection mapping of deformable, stretchable materials that is also backward compatible to non-deformable rigid materials using a single *calibrated PDC*, or **P**rojector-**D**epth **C**amera pair. Specifically, the main contributions in this chapter are as follows:

1. **Surface geometry representation**: We use rational B-spline patches to model the projection surface without explicitly modeling the deformation. This allows for integrating the modeling and updating of surface geometry, as well as warping of the projection image in a computationally efficient representation that lends itself for efficient GPU-based parallelization.

2. **Boundary-based tracking**: Distortions in the middle of deformable materials are less perceptible than at the boundaries [23]. We leverage this to track only the boundary of the material using simple markers (e.g. black dots) or depth-features with no explicit markers leading to projection mapping with or without markers.

3. **Mapping stretchable materials**: Using B-spline patch-based representations enable constrained projection mapping (e.g. length preserving mapping) to allow realistic mapping on stretchable materials.

4. **Using consumer-grade hardware**: Our system uses a consumer-grade time-of-flight (ToF) based depth camera (e.g. Azure Kinect, Pico Flexx) to achieve real-time mapping of the projected image on the deformable material such that it appears to be printed on the object. Further, we show that the computation speed is limited by the capture time, demonstrating that our method will be faster with higher speed cameras and projectors.

2.1 Related Work

Projection mapping deals with altering appearances of 3D objects by projecting light on them using projectors, enabling large-scale displays that can be used to visualize high resolution content [37, 38, 132]. A large body of literature exists on projection mapping on static objects [118, 90, 101, 102, 99, 88, 11, 49, 93, 89, 50]. More recently, dynamic projection mapping (DPM), that allows mapping projected light on moving objects, has received much attention [18, 114, 110, 77, 139, 111, 63, 78, 72, 71, 2, 58, 53, 128, 94, 27, 109, 62, 98, 5, 113, 87]. Prior work on dynamic projection mapping can be categorized based on the following: **(a)** *rigid* vs. *deformable* materials, **(b)** projection on *surfaces* vs. *objects*, **(c)** display on *textured* (e.g. using markers or patterns) vs. *texture-less* materials, and **(d)** using *consumer-grade components* vs. *specialized hardware* like a coaxial projector-camera setup.

2.1.1 Rigid Surfaces

When considering projection mapping on dynamic rigid bodies [114, 110, 139, 63, 71, 2, 53, 128, 94, 27, 109, 62], a large body of literature focuses only on rigid, *planar surfaces*. [71, 53, 128] use a *specialized* high-speed projector and camera in a co-axial manner to map the planar surface.

Texture-less surfaces: While [71] uses a high-speed color camera, [128] uses a high-speed depth camera and adjusts the focus based on the captured depth to keep the image focused on the plane as it moves back and forth.

Textured surfaces: [53] uses specialized digital micromirror device (DMD) hardware to project animated content onto *textured*, planar surfaces. The DMD allows projection of special patterns in an imperceptible way between projection frames to recover the four corners of the planar display and track them.

2.1.2 Rigid Objects

Texture-less objects: Projection mapping on moving, rigid, non-planar objects requires computing the orientation and geometry of the objects, also known as shape and pose recovery, in addition to tracking the moving object. [114] uses an expensive, high-speed *coaxial projector-camera pair* to bypass shape recovery and performs projection mapping without any perceptible latency by segmenting the dynamic object from a retro-reflective background.

[139, 2, 94, 27] use dynamic objects of *known* shapes while proposing different tracking methods using different kinds of *consumer-grade components*. [27] detects edges of *known* shapes and tracks them in real-time using a single IR camera to recover the object pose. [94] uses a sparse ICP registration technique to register 2D feature points detected by a standard RGB camera with a *known* 3D shape to track the object. However, the latency is not low enough to allow fast movement of the 3D object.

[110, 63, 58, 109, 62] perform dynamic projection mapping on rigid 3D objects using multiple projectors. [110, 63] use a tightly calibrated system of 2-3 projectors with a single RGB-D camera to map a dynamic, complex 3D object whose shape has been recovered accurately apriori. Using a light-transport based optimization, key features of the point cloud captured by the RGB-D camera are detected and matched to the high-quality, *known* 3D shape. Though this method handles dynamic objects, it has high latency and therefore, the object can only move slowly. This system is extended in [109] to handle stray illumination due to geometric features and inter-reflections in a content-specific manner. Finally, [58] removes the requirement of tight calibration of the devices apriori by auto-calibrating devices while mapping projected images on the dynamic object.

Textured objects: [2] embeds 3D features as *markers* in the object during 3D printing that provide the features to be tracked in real-time. These embedded markers are made imperceptible during tracking via radiometric compensation.

In summary, all works addressing rigid dynamic objects assume a *known*, accurate 3D shape and cannot be used to project on dynamic, deformable and stretchable materials.

2.1.3 Deformable Surfaces

Texture-less surfaces: [18] embeds retro-reflective markers in a surface that are only visible to an IR camera when an IR light source is shined onto it. They use the marker positions to determine the shape to project onto the surface. Similarly, [84, 85] paint markers with IR ink and embed them in various gel-like substances. These markers are visible to an IR camera under IR lighting and are used to determine the deformation of the gel to perform projection mapping. Unlike [18], [84, 85] are able to handle interactive movement and update the projection in real-time. However, the gel-like substances are small and lie on a flat surface, restricting projection mapping to only those regions.

[118] presents a scalable system that can accommodate any number of projectors and cameras and can handle surfaces with any 3D shape without any prior system calibration. Therefore, the system recalibrates with deformations. However, this system can only achieve fast non-real-time recalibration and therefore cannot handle dynamic objects.

[87, 54] use an image of a deformable fabric captured by an RGB camera and use prior techniques [10, 8] to process the image properties (e.g. optical flow) to create an enhanced image that alter the perceived motion or stiffness of the fabric. However, these methods work with precise known motions of the deformable surfaces and do not attempt to capture either the motion or the actual geometry of the deformable surface.

[113] addresses deformable surfaces using a projector and RGB-D camera pair. They model the surface deformations (rather than shape) and recover the deformation parameters from the captured point cloud using GPU-based optimizations. However, the deformation model used does not allow stretchable surfaces and the optimizations cannot achieve real-time updates to accommodate fast moving materials.

Textured surfaces: [77, 78] achieve projection mapping on dynamic, deformable and stretchable materials using a *specialized*, tightly calibrated, expensive, high-frame rate (∼500fps) projector-IR camera system that tracks dot cluster markers printed in the material with invisible IR ink. This results in conformal projection with imperceptible lag that can handle partial occlusion of the surface as well.

2.1.4 Deformable Objects

Texture-less objects: [72] uses a *specialized*, high-speed projector along with three high speed cameras, all arranged in a co-axial manner using mirrors so that they share the same center of projection. Three NIR light sources of different bands are used to illuminate the object. The three cameras are equipped with color filters to sense each of these three light

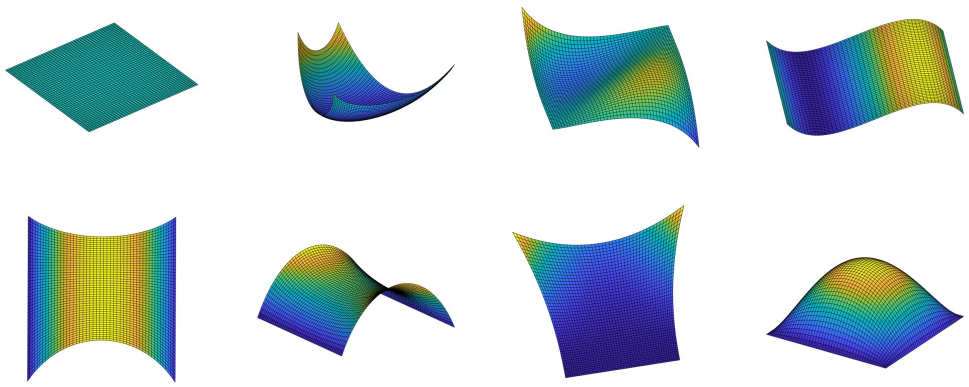


Figure 2.1: Various surfaces that can be represented by a cubic B-Spline patch, including (bottom-row) surfaces stretched down the middle, at the upper two corners and outward from the center.

sources of different bands. The light sources and cameras together use photometric stereo to detect the normals at every pixel location in the camera space. These normals are then used to compute the illumination augmentation required to change the appearance of the object to that of a material with different normals. Due to co-axial arrangements of the projector-camera setup, the correction can be achieved in real time in the camera space irrespective of the dynamic object that is being mapped – either rigid, deformable or fluids – without perceptible lag.

[111, 5] specifically address faces. [5] uses a *specialized* co-axial fast projector-camera system along with LED-based lighting to track facial features in IR. The tracked 2D facial features are used to compute the deformation parameters of the 2D mesh of a face. [111] uses multiple projectors and an RGB-D camera in a tightly calibrated system to accurately estimate the 3D mesh of a face. During projection mapping, they update the deformation parameters of the mesh in real-time to register it with the depth camera output. Use of fast cameras in [5] allows imperceptible lag while [111] achieves almost real-time performance with low latency. [98] uses a Kinect in a specific retail dressing room kind of setting to segment multiple users using depth and projects different T-shirt designs on each person. However, the accuracy

of projection is extremely low and therefore, the projection does not stick to the objects precisely and flickers continuously.

Prior work on tracking deformable objects in real-time using an RGB-D camera [81, 64, 79, 84] cannot be used with dynamic projection mapping since the projection on the surface interferes with the tracking. Therefore, IR camera-based tracking is typically used for dynamic deformable objects. Thus, most works that perform dynamic projection mapping on deformable, texture-less objects either use specialized hardware [72, 5] or are limited to a specific class of deformable objects (e.g. faces) as in [111].

2.1.5 Comparison to Prior Work

The work in this chapter is closest to [113] but instead of modeling deformations, we model the surface geometry using a rational B-spline patch (see Figure 2.1). We use rational B-spline patches because they **(i)** provide a framework to represent all kinds of smooth, deformable surfaces, including stretchable elastics, **(ii)** offer a convenient way to parameterize the surface, **(iii)** lend themselves to accurate computation on GPU for real-time performance, and **(iv)** provide the flexibility to change the surface representation (e.g. linear to quadratic etc.) at run-time without any changes to the system.

Additionally, unlike [114, 77, 72, 71, 53, 5], we don't need *specialized* hardware such as co-axial projector-camera pairs, or high-speed, expensive equipment. Instead, we achieve real-time performance using consumer-grade hardware. While consumer-grade hardware is more accessible and affordable, they provide noisier data at lower spatial resolutions and frame-rates. This puts further demands on the system, which must perform projection mapping faster than the hardware frame-rate to avoid lag, from lower resolution data while being robust to noise.

Despite these limitations of consumer-grade hardware, our system is able to achieve the above objectives with a standard ToF depth camera that is augmented with a registered IR camera calibrated together with a projector. Using an IR-Depth camera enables us to track simple markers (e.g. black dots) leading to a marker-based system, or to track depth features leading to a marker-less system. Like [27], we use boundary-based features (e.g. edges or corners), but to determine the deformable, stretchable surface geometry rather than deformation as in [43, 104]. Therefore, we present the first general framework for projection mapping of deformable materials, including stretchable elastics, using a single projector-camera system in real-time, operating at speeds up to 45fps. Since we depend on per frame surface representation using B-spline patches, we can handle rigid materials as well.

2.2 System Overview

The goal of our projection mapping system is to compute a texture-mapping function $\Omega(q) = p$ that maps a texture coordinate $q \in \mathbb{R}^2$ of a target image I_T to a texture coordinate $p \in \mathbb{R}^2$ of a source image I_S such that I_T conforms to the shape of the projection surface when it gets projected:

$$I_T(q) = I_S(\Omega(q)). \tag{2.1}$$

In this section, we explain how we compute $\Omega(\cdot)$ by tracking the boundaries of a deformable, stretchable display surface with or without markers, in real-time. The complete pipeline of our method is shown in Figure 6.2.

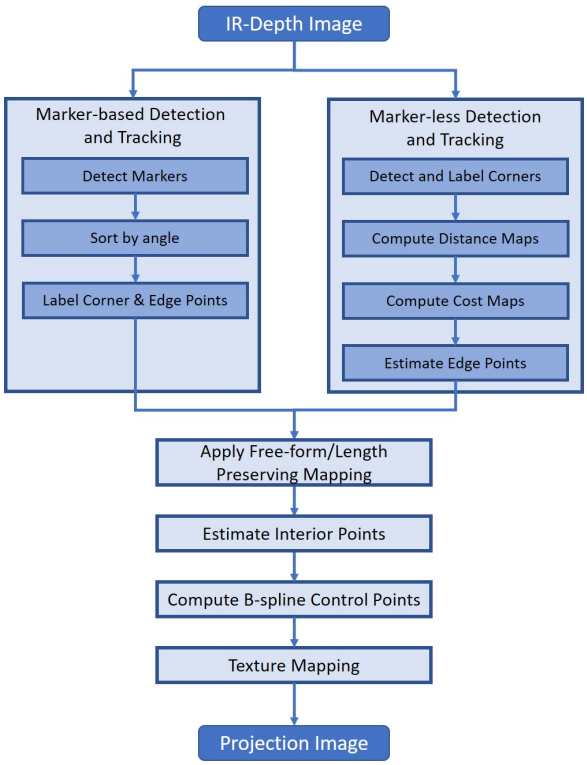


Figure 2.2: Pipeline of our system, showing the marker-based and marker-less algorithms.

2.2.1 System Hardware

Our dynamic projection mapping system consists of a projector, an IR camera and a depth camera (Figure 2.3) that have been geometrically calibrated for their intrinsic and extrinsic parameters using the software implementation of [76, 75]. We assume our projection surface to be a smooth, deformable, elastic rectangle that is visible to the camera without occlusion. Each frame of the IR-depth camera is used to track feature points on the boundary of the deformable projection surface. These features can be explicitly marked using color (e.g. black dots) or by tracking depth features at the boundary of the projection surface. The former results in a system with visible boundary markers (Section 2.2.3), while the latter yields a marker-less system (Section 2.2.5). These features are used to compute the parameters of a rational B-spline patch representing the deformable surface shape. The fitted patch is then used to determine the texture-mapping function $\Omega(\cdot)$. Finally, the projection image is

Table 2.1: Important notation used for B-spline patch in Chapter-2

Variable	Definition
H	Number of B-spline patch control points
T	B-spline knot vector
(M_u, M_v)	Number of markers in the horizontal and vertical directions respectively
(n, m)	Degree of B-spline patch
(u, v)	B-spline patch parameters
$\bar{\boldsymbol{Q}}(u, v)$	3D homogeneous point on the B-spline patch at (u, v)
$\bar{\boldsymbol{q}}(u, v)$	2D homogeneous point corresponding to $\bar{\boldsymbol{Q}}(u, v)$
$Q(u, v)$	3D point on the B-spline patch at (u, v)
$q(u, v)$	2D projector pixel corresponding to $Q(u, v)$
$\boldsymbol{N}(u, v)$	B-spline patch basis function evaluated at (u, v)
$\boldsymbol{C}$	$\mathbb{R}^{4\times H}$ matrix of 3D homogeneous B-spline patch control points
$\bar{\boldsymbol{C}}_p$	$\mathbb{R}^{3\times H}$ matrix of 2D homogeneous B-spline patch control points

warped accordingly and projected onto the display surface in a manner that conforms to the deformable shape.

2.2.2 B-spline Patch Based Model

A rational B-spline patch of degree (n, m) in d dimensions is defined in parameters (u, v) by a set of $H = (r + 1) \times (s + 1)$ control points $C_{0,0}, C_{0,1}, \ldots, C_{(r,s-1)}, C_{(r,s)}$ and weights $W_{0,0}, W_{0,1}, \ldots, W_{(r,s-1)}, W_{(r,s)}$ as:

$$\begin{aligned} Q(u, v) &= \sum_{i=0}^{r} \sum_{j=0}^{s} P_{i,j}(u, v) C_{i,j}, \\ P_{i,j}(u, v) &= \frac{N_{i,n}(u) N_{j,m}(v) W_{i,j}}{\sum_{k=0}^{r} \sum_{l=0}^{s} N_{k,n}(u) N_{l,m}(v) W_{k,l}}, \end{aligned} \tag{2.2}$$

where $W_{i,j} \in \mathbb{R},\ C_{i,j}, Q(u, v) \in \mathbb{R}^d$. $N_{i,n}(u)$ is the B-spline basis function of degree n. It is defined by a non-decreasing knot vector $T = \{t_0, t_1, ..., t_h\}$ as:

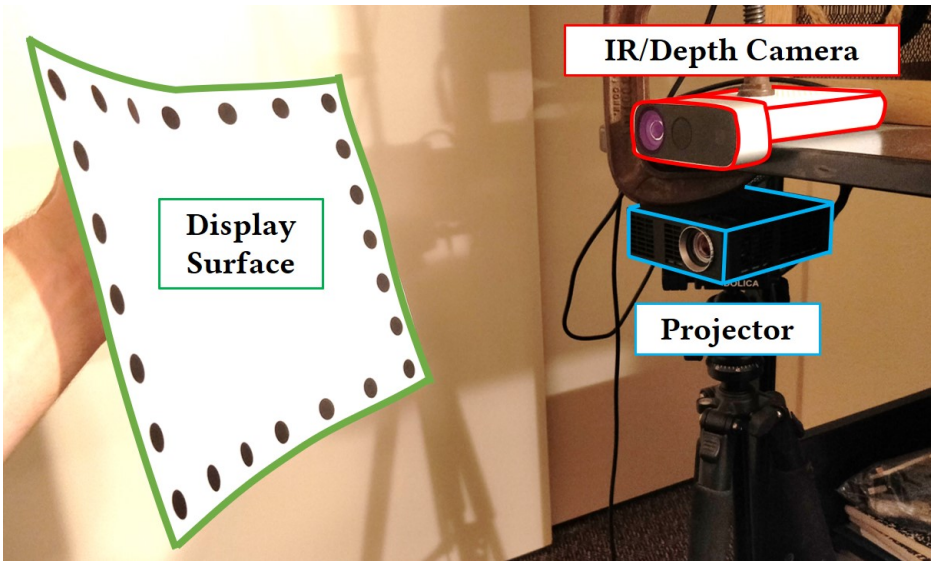


Figure 2.3: Our setup: a geometrically calibrated projector (blue) and IR/Depth camera (red) pair along with the display surface (green).

$$
\begin{aligned}
N_{i,0}(u) &= \begin{cases} 1, & t_i \leq u < t_{i+1} \\ 0, & \text{otherwise} \end{cases}, \\
N_{i,n}(u) &= \frac{u - t_i}{t_{i+n} - t_i} N_{i,n-1}(u) + \ldots \\
&\quad \frac{t_{i+n+1} - u}{t_{i+n+1} - t_{i+1}} N_{i+1,n-1}(u).
\end{aligned}
\tag{2.3}
$$

Note that the number of control points r, the B-spline degree n and the size of the knot vector h must satisfy the identity $h = r + n + 1$ for both directions (u, v). Thus, by changing the B-spline degree and/or the size of the knot vector, one can change the number of control points of the B-spline.

In our work, we model our projection surface as an open, uniform, rational B-spline patch of degree (n, m). In an *open uniform* B-spline of degree n, the first $n + 1$ knots and the last $n + 1$ knots are equal, while the remaining, internal knots are non-decreasing and equally spaced. In a *non-rational* B-spline, the weights of each control point are the same, limiting

their ability to represent more complex shapes such as circles, conics and tori. These shapes can be accurately modeled with a *rational* B-spline, where the weights may not be equal. Furthermore, rational B-splines produce correct results under projective transformation, a property we leverage when computing $\Omega(\cdot)$ to render the image for projection. Thus, a B-spline patch can be used to represent various shapes as shown in Figure-2.1 and is parameterized by (u, v) in the horizontal and vertical directions respectively (Figure-2.4). In matrix form, a rational B-spline patch in 3D can be expressed as:

$$\bar{\boldsymbol{Q}}(u,v) = \begin{bmatrix} X(u,v)\ W(u,v) \\ Y(u,v)\ W(u,v) \\ Z(u,v)\ W(u,v) \\ W(u,v) \end{bmatrix} = \boldsymbol{C}\ \boldsymbol{N(u,v)}$$

$$\bar{\boldsymbol{C}} = \begin{bmatrix} X_{00}W_{00} & X_{01}W_{01} & \dots & X_{rs}W_{rs} \\ Y_{00}W_{00} & Y_{01}W_{01} & \dots & Y_{rs}W_{rs} \\ Z_{00}W_{00} & Z_{01}W_{01} & \dots & Z_{rs}W_{rs} \\ W_{00} & W_{01} & \dots & W_{rs} \end{bmatrix} \in \mathbb{R}^{4\times H}, \tag{2.4}$$

$$\boldsymbol{N}(u,v) = \begin{bmatrix} N_{0,n}(u)N_{0,m}(v) \\ N_{0,n}(u)N_{1,m}(v) \\ \vdots \\ N_{r,n}(u)N_{s,m}(v) \end{bmatrix} \in \mathbb{R}^{H\times 1},$$

where $H = (r+1)\times(s+1)$. Here, $\begin{bmatrix} X_{ij}W_{ij} & Y_{ij}W_{ij} & Z_{ij}W_{ij} & W_{ij} \end{bmatrix}$ is a 3D B-spline control point in homogeneous coordinates, and $Q(u,v) = \begin{bmatrix} X(u,v) & Y(u,v) & Z(u,v) \end{bmatrix}$ is a 3D point on the B-spline patch.

Representing the display surface using a B-spline patch lets us express $\Omega(\cdot)$ as a function of the (u, v) parameters. Using the control points $\bar{\boldsymbol{C}}$, we can compute $Q(u, v)$ for any (u, v) parameters. Since our system is geometrically calibrated, we can project $Q(u, v)$ onto the projector image plane to determine the 2D projector pixel $q(u, v)$. This creates a mapping between the projector pixel $q(u, v)$ and the display surface parameters (u, v):

$$\Omega(q(u, v)) = (u, v), \tag{2.5}$$

i.e. every pixel $q(u, v)$ in $\Omega(\cdot)$ maps to a display surface parameter (u, v). Substituting Equation 2.5 into Equation 2.1, we get:

$$I_T(q(u, v)) = I_S(u, v). \tag{2.6}$$

Thus, our goal is to compute $q(u, v)$ for any (u, v) display parameter. We achieve this by computing the control points of the B-spline patch that represent the display surface using a system of linear equations. The key challenge of our work is: **(i)** tracking the deformable, stretchable surface, **(ii)** determining the (u, v) parameters of 3D points on the surface, **(iii)** setting up a system of linear equations using these 3D points, **(iv)** solving the system to compute the control points, **(v)** using the control points to compute $q(u, v)$, and **(vi)** perform texture-mapping for projection, all in *real-time*.

A method to track stretchable display surfaces using markers and determining the (u, v) parameters for the tracked points is detailed Section 2.2.3, while a method for markerless tracking is elaborated in Section 2.2.5. Setting up the system of linear equations and solving it

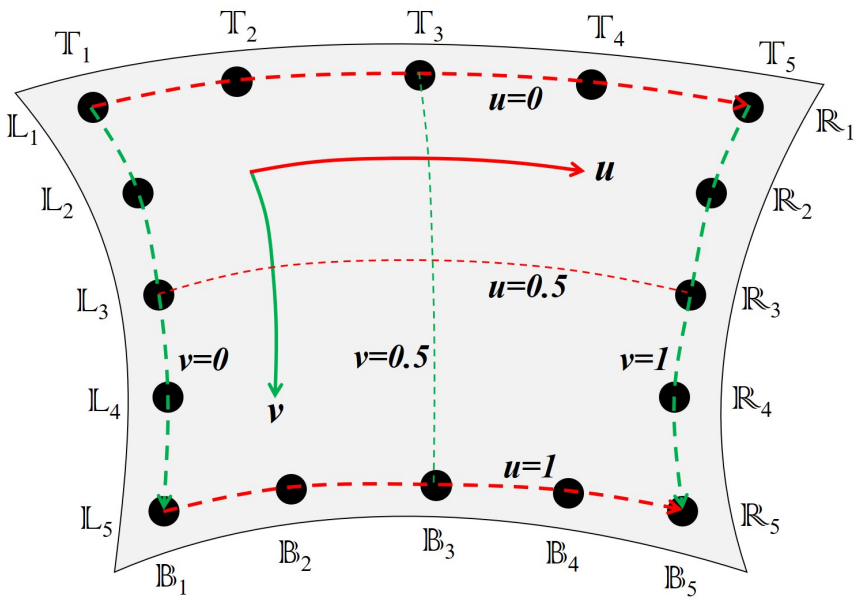


Figure 2.4: Illustration of a deformable surface with dots as markers. The iso-parametric boundary curves at $u = 0$ (top), $u = 1$ (bottom), $v = 0$ (left) and $v = 1$ (right) are shown along with the boundary labels for each marker $(\mathbb{T}_i, \mathbb{B}_i, \mathbb{L}_j, \mathbb{R}_j)$. The (u, v) parameterization of the top-left corner is $(0, 0)$ and bottom-right corner is $(1, 1)$.

to compute the control points of the B-spline patch is explained in Section 2.2.4. Computing $q(u, v)$ and performing texture-mapping is explained in Section 2.2.6.

2.2.3 Marker-based Tracking of Stretchable Surface

Marker Placement

Our projection surface is a smooth rectangle made of a diffuse, deformable and elastic material. We define the boundaries of our surface with markers such that: (i) four markers specify the four corners of this rectangle, (ii) a fixed number of markers, known apriori, are placed equally spaced along each edge, (iii) including the corners, the top and bottom edges have equal number of markers M_u, and the left and right edges have equal number of markers M_v. We denote markers on the top and bottom edges with $\mathbb{T}_i$ and $\mathbb{B}_i$, $1 \leq i \leq M_u$ and markers

on the left and right edges with $\mathbb{L}_j$ and $\mathbb{R}_j$, $1 \leq j \leq M_v$ respectively. Figure-2.4 shows an example of our surface, where three markers are placed along each edge in addition to the corner markers. Note that the shape recovery does not depend on the surface being white and will therefore work for textured surfaces as well. However, this work does not focus on photometric corrections for textured surfaces. Therefore, any flat colored surface will yield acceptable projections.

Since the markers $\mathbb{T}_i, \mathbb{B}_i, \mathbb{L}_j$ and $\mathbb{R}_j$ are equally spaced on each edge, we can pre-compute their (u, v) parameters as $(u_i, 0), (u_i, 1), (0, v_j)$ and $(1, v_j)$ respectively, where $u_i = \frac{i-1}{M_u - 1}, v_j = \frac{j-1}{M_v - 1}$. This arrangement gives us clear matches between the boundary markers and their (u, v) parameters on the B-spline patch.

Marker Detection and Tracking

The markers are tracked every frame and used to compute the 3-D control points of the B-spline patch. We use IR imagery in order to avoid interference caused by the visible projection. The markers, being black, are usually well-visible in the IR camera against the surface. In the first frame, we use a simple blob detector on the IR image to detect the markers. In subsequent frames, each detected marker is tracked using KLT feature tracking [67, 120, 107]. This results in fast and robust tracking of the markers. If tracking fails, we retract back to marker detection and repeat the procedure.

Assigning (u, v) Parameters to Markers

The detection and tracking step provides marker locations in the camera image space. However, this is not sufficient to establish the (u, v) parameters for each marker. This objective is achieved in the marker (u, v) assignment step, detailed as follows.

Sort by Angle: Let I_{IR} denote the IR-camera image and I_{PC} denote the point cloud image from the depth camera. Let $\mathbb{Q}$ denote the set of boundary markers and $\mathbb{Q}_i = \{q_i, Q_i\}$ denote i-th marker, where $q_i \in \mathbb{R}^2$ is the 2D location of the marker in I_{IR} and $Q_i = I_{PC}(q_i) \in \mathbb{R}^3$ is the corresponding 3D point in the point cloud image. We compute the centroid of all the detected markers, $\mathbb{Q}_{avg}$, as:

$$\mathbb{Q}_{avg} = \{q_{avg}, Q_{avg}\} = \frac{1}{|\mathbb{Q}|} \sum_{i=1}^{|\mathbb{Q}|} \{q_i, Q_i\} \tag{2.7}$$

The first step is to sort the detected markers by the angle along the (u, v) surface, denoted by θ, from a reference vector connecting $\mathbb{Q}_{avg}$ to $\mathbb{Q}_1$. Note that the closest approximation of the (u, v) space is available in the 2D IR camera space. Despite perspective projection, the angle of the (u, v) coordinate of a marker from the reference vector will increase monotonically in the IR camera space as we move clockwise from one marker to the next. This invariance can be violated when using 3D coordinates (illustrated in Figure 2.5). Therefore, we perform this sorting by angle in the 2-D IR camera space.

Labeling Corner and Edge Points: Sorting by θ provides the ordering of the markers around the surface, with which we can determine the adjacent markers for each marker. Therefore, if we can assign the (u, v) parameters of any one marker correctly, we can determine the parameters for all the remaining ones. So first, we identify the marker that corresponds to the top-left corner and assign $(u, v) = (0, 0)$ to it. We compute, for every marker $\mathbb{Q}_i$, the angle ϕ_i in 3D between the edges connecting it to its two adjacent neighbors $\mathbb{Q}_{i-1}$ and $\mathbb{Q}_{i+1}$ (Figure 2.5):

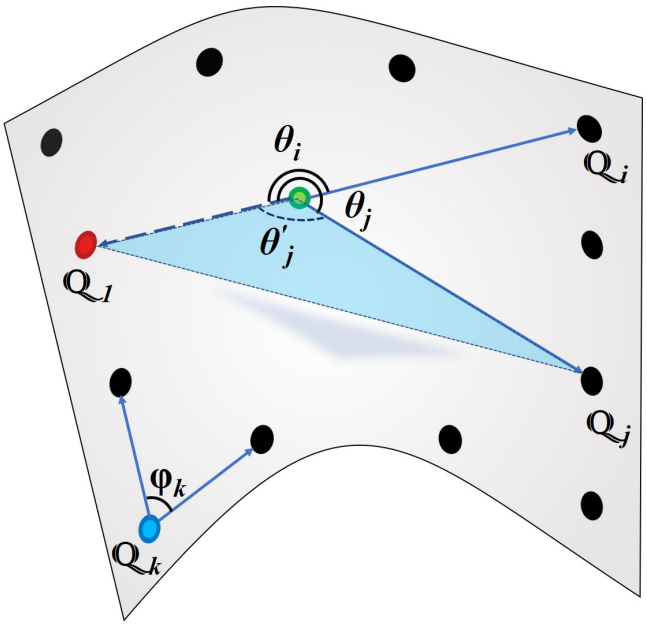


Figure 2.5: Determining the top-left corner for a surface that is folded inward. $\mathbb{Q}_{avg}$ is marked in green. Initially, the detections are not ordered. Note that $\theta_i < \theta_j$ when using 2D image coordinates. However, using 3D coordinates, the angle $\theta_j^{'}$ between the two vectors spans the plane connecting them rather than along the (u, v) surface. This causes $\theta_j^{'} < \theta_i$ which is incorrect. While 2D image coordinates work better than 3D coordinates when relative ordering of angles is required, 3D coordinates are used to label a corner such as $\mathbb{Q}_k$ (blue marker, bottom-left) for which $\phi_k \approx 90°$, which may not be recognizable in 2D camera image space due to perspective distortion.

$$\begin{aligned} \phi_i &= \Gamma(\vec{\alpha_i}, \vec{\beta_i}), \\ \vec{\alpha_i} &= Q_{i-1} - Q_i, \\ \vec{\beta_i} &= Q_{i+1} - Q_i, \end{aligned} \tag{2.8}$$

where $\Gamma(\vec{\alpha}, \vec{\beta})$ computes the angle between vectors $\vec{\alpha}, \vec{\beta}$.

By construction, corner points make an angle close to 90° with their neighbors and edge points make an angle close to 180°, resulting in two clusters of points where the number of points in the cluster $\phi_i \approx 90°$ would be four, corresponding to the four corners of the display. However, due to surface deformations, we may not get exactly four corners. Theoretically, if

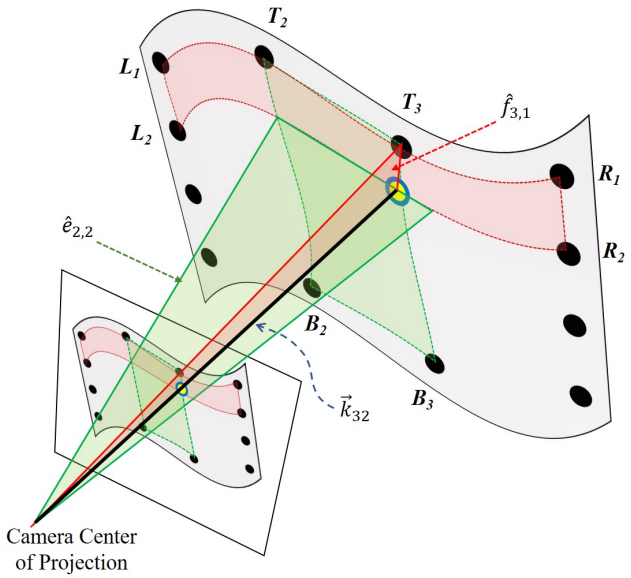


Figure 2.6: Estimating interior points of the surface by normal interpolation. Markers used to compute the plane normals are labeled. The green region on the surface is the area scanned by the plane as it is interpolated between $\hat{t}_2^3$ and $\hat{b}_2^3$, while $\hat{e}_{2,2}$ is shown in solid green. The red region is the area scanned by the plane as it is interpolated between $\hat{l}_1^2$ and $\hat{r}_1^2$, while $\hat{f}_{3,1}$ is shown in solid red. The ray $\vec{k}_{32}$ generated by intersecting $\hat{e}_{2,2}$ and $\hat{f}_{3,1}$ is used to estimate the interior point Q_{32} (blue-yellow point).

the stretching distortion is extreme at a corner, the corner marker can have an angle close to 180° between its horizontal and vertical edge neighbors leading to false negatives, and if the folding distortion is extreme, then an edge marker can have close to 90° with its neighbors along the same edge leading to false positives. We note that the stretching distortion is unrealistic and hence we can ass ume that there are no false negatives. With respect to the folding distortion and hence false positives, we can eliminate false positives and find the correct corners using the invariance that there are alternately $(M_u - 2)$ and $(M_v - 2)$ markers between the corner markers.

Note that any corner can be considered the top-left corner (with $(u, v) = (0, 0)$) as long as it is tracked in subsequent frames and other corners are labeled correctly with respect to the top-left corner. Let the four corners be $\mathbb{C}_k, 1 \leq k \leq 4$. To determine the top-left corner, we approximate the distance along the surface between $\mathbb{C}_k$ and its adjacent clockwise

corner $\mathbb{C}_{k+1}$ by summing the Euclidean distances between neighboring markers from $\mathbb{C}_k$ to $\mathbb{C}_{k+1}$. This results in two corners with a larger distance (the longer edges of the display) compared to the other two (the shorter edges of the display). From the corners with the two largest distances, the corner closest to the camera origin is labeled as the top-left corner and $(u, v) = (0, 0)$ is assigned to that marker. Then, the (u, v) parameters for all markers are assigned using the cyclic order of markers computed earlier.

All markers are labeled once during the initialization phase. In subsequent frames, each marker and its label is tracked, allowing our projection display to maintain correct orientation even when the surface is rotated upside down.

Estimating Interior Points

Boundary markers and their (u, v) assignments are not enough to compute the B-spline patch. We need additional 3D points and their (u, v) parameters in the interior of the projection surface where no markers exist. Therefore, we estimate an additional $(M_u - 2) \times (M_v - 2)$ interior 3D points and assign their (u, v) parameters at $(\frac{i-1}{M_u-1}, \frac{j-1}{M_v-1})$, where $2 \leq i \leq (M_u - 1), 2 \leq j \leq (M_v - 1)$.

In order to estimate these 3D points, we assume a piece-wise linear representation of the boundary curves, which are also iso-parametric (see Figure-2.4). Let $t_i, b_i, l_j, r_j \in \mathbb{R}^2$ denote the 2D coordinates of these markers in IR-camera space, and $T_i, B_i, L_j, R_j \in \mathbb{R}^3$ denote their 3D coordinates. Let the markers along each edge be denoted by $\mathbb{T}_i = \{t_i, T_i\}, \mathbb{B}_i = \{b_i, B_i\}, \mathbb{L}_j = \{l_j, L_j\}$ and $\mathbb{R}_j = \{r_j, R_j\}$. We compute the normals of the planes passing through every adjacent pair of these markers and the camera center of projection:

$$\begin{aligned} \hat{t}_i^{i+1} &= \zeta(T_i, T_{i+1}), \quad & \hat{l}_j^{j+1} &= \zeta(L_j, L_{j+1}), \\ \hat{b}_i^{i+1} &= \zeta(B_i, B_{i+1}), \quad & \hat{r}_j^{j+1} &= \zeta(R_j, R_{j+1}), \\ 1 &\leq i \leq (M_u - 1), \quad & 1 &\leq j \leq (M_v - 1), \end{aligned} \tag{2.9}$$

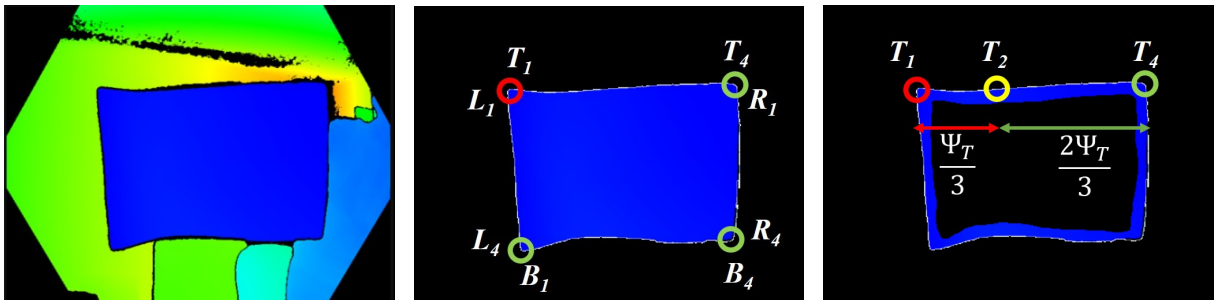
where $\zeta(K_1, K_2)$ computes the normal of the plane passing through 3D points K_1, K_2 and the camera center of projection. This leads to $(M_u - 1)$ normals for each of the top and bottom curves, and $(M_v - 1)$ normals for the left and right curves. Note that while the boundary markers are *tracked* every frame, the interior points and their (u, v) parameters are *estimated* for each frame.

We interpolate between each $\hat{t}_i^{i+1}$ and $\hat{b}_i^{i+1}$ to estimate the normals $\hat{e}_{i,j}$ for iso-parametric curves $v_j = \frac{j-1}{M_v - 1}$:

$$\hat{e}_{i,j} = \hat{t}_i^{i+1}(1 - v_j) + \hat{b}_i^{i+1} v_j. \tag{2.10}$$

In Figure-2.6, the green region shows the area scanned by the plane as its normal is interpolated between $\hat{t}_2^3$ and $\hat{b}_2^3$, while the interpolated plane $\hat{e}_{2,2}$ at $v_2 = \frac{1}{3}$ is shown in solid green. Similarly, we interpolate between $\hat{l}_j^{j+1}$ and $\hat{r}_j^{j+1}$ to estimate the normals $\hat{f}_{i,j}$ that pass through $u_i = \frac{i-1}{M_u - 1}$:

$$\hat{f}_{i,j} = \hat{l}_j^{j+1}(1 - u_i) + \hat{r}_j^{j+1} u_i. \tag{2.11}$$



(a) (b) (c)

Figure 2.7: Distance-based marker-less tracking. **(a)** The depth map. **(b)** The segmented point cloud along with the results of corner detection (the top-left corner is highlighted in red). **(c)** The boundary point cloud I_B. T_2 is the point on the top edge that is at a distance $\frac{1}{3}\Psi_T$ from T_1 (red circle) and $\frac{2}{3}\Psi_T$ from T_4 (green circle).

The red region in Figure-2.6 shows the area scanned by the plane as its normal is interpolated between $\hat{l}_1^2$ and $\hat{r}_1^2$, while the interpolated plane $\hat{f}_{3,1}$ at $u_3 = \frac{2}{3}$ is shown in solid red.

These interpolated normals are used to estimate the interior 3D points at (u_i, v_j) for $2 \leq i \leq (M_u - 1), 2 \leq j \leq (M_v - 1)$. We compute the ray $\vec{k}_{ij}$ by intersecting the interpolated horizontal and vertical planes $\hat{e}_{(i-1,j)}$ and $\hat{f}_{(i,j-1)}$. Figure-2.6 shows $\vec{k}_{32}$, the intersection of $\hat{e}_{2,2}$ and $\hat{f}_{3,1}$ as a black line. Since $\vec{k}_{ij}$ passes through the camera center of projection (COP), we find the 2D coordinate q_{ij} by projecting it onto the IR camera image plane. The 3D point at (u_i, v_j) can be determined from the depth camera as $Q_{ij} = I_{PC}(q_{ij})$. This step is performed in every iteration.

Note that whenever a stretch is applied to the display surface, *the distances between markers change in 3D but the distances in* (u, v)*-space do not.* This allows a B-spline patch to model stretches to the surface.

2.2.4 Computing Control Points

Once we have determined 3D points on the surface and their (u, v) assignments, we can compute the control points by solving the following system of linear equations for the control points $\bar{\boldsymbol{C}}$:

$$
\begin{aligned}
\bar{\boldsymbol{Q}} &= \bar{\boldsymbol{C}}\boldsymbol{A}, \\
\bar{\boldsymbol{Q}} &= \begin{bmatrix} X(u_1,v_1) & X(u_2,v_2) & \dots & X(u_F,v_F) \\ Y(u_1,v_1) & Y(u_2,v_2) & \dots & Y(u_F,v_F) \\ Z(u_1,v_1) & Z(u_2,v_2) & \dots & Z(u_F,v_F) \\ 1 & 1 & \dots & 1 \end{bmatrix}, \\
\bar{\boldsymbol{C}} &= \begin{bmatrix} X_{00} & X_{01} & \dots & X_{rs} \\ Y_{00} & Y_{01} & \dots & Y_{rs} \\ Z_{00} & Z_{01} & \dots & Z_{rs} \\ 1 & 1 & \dots & 1 \end{bmatrix}, \\
\boldsymbol{A} &= \begin{bmatrix} \boldsymbol{N}(u_1,v_1) & \boldsymbol{N}(u_2,v_2) & \dots & \boldsymbol{N}(u_F,v_F) \end{bmatrix},
\end{aligned}
\tag{2.12}
$$

where F is the number of (u, v) assignments, $\bar{\boldsymbol{Q}} \in \mathbb{R}^{4\times F}$, $\bar{\boldsymbol{C}} \in \mathbb{R}^{4\times H}$, $\boldsymbol{N}(u, v) \in \mathbb{R}^{H\times 1}$ and $\boldsymbol{A} \in \mathbb{R}^{H\times F}$. Note that all the weights W_{ij} are set to 1, making this a non-rational B-spline.

In order to solve Eq-2.12, the condition $F \geq H$ must be met. While $F = (M_u \times M_v)$, depends on the number of markers, the number of control points, H, depends on the B-spline degree and the size of the knot vector (see Sec-2.2.2). By changing these two parameters, we can change the number of control points and use different B-spline models even if the number of markers remains the same (see Sec-2.2.6 and Figure-2.15).

2.2.5 Marker-less Tracking of Stretchable Surface

In our display, the distance between adjacent markers is the same when no stretch is applied. However, in the absence of explicit markers, the *geodesic* distance between 3D points can be used to assign the (u, v) parameters. A *geodesic* is a curve representing the shortest path between two points on a surface and the *geodesic* distance is the length of that curve. In this section, we outline how we use this concept to create projection displays for marker-less, deformable and stretchable surfaces.

In each frame, we segment out the display surface from the point cloud image assuming there is no occlusion of the surface. This provides a mask that corresponds to the segmented surface (Figure 2.7b). We extract the boundary mask by taking the difference of the surface segmentation mask with its morphological erosion and compute the boundary point cloud I_B (highlighted in blue in Figure 2.7c).

While the markers provided explicit brightness features in our marked display, for a marker-less display, the only implicit features are the four corners of the display. We find them using the Harris corner detector [26] on the segmented mask and track them using the KLT feature tracker. Then, we assign the (u, v) parameters to these corners as outlined in Section 2.2.3.

Distance-based Edge Marker Estimation

Recall that the parameters for a marker on the top edge, denoted by $\mathbb{T}_i$, is $(u_i, 0)$, where $u_i = \frac{i-1}{M_u-1}$. For the marker-less display, the goal is to find all $\mathbb{T}_i$s, even though they are not explicitly marked on the surface. From the corner detection step, we already have $\mathbb{T}_1$ and $\mathbb{T}_{M_u}$. Let Ψ_T denote the length of the top edge. Other $\mathbb{T}_i$s are found such that **(i)** they are along the top edge, **(ii)** their *geodesic* distance from $\mathbb{T}_1$ is $u_i\Psi_T$, and **(iii)** their *geodesic* distance from $\mathbb{T}_{M_u}$ is $(1 - u_i)\Psi_T$.

Let $\boldsymbol{G}$ denote the set of 3D points (from the point cloud image) on the geodesic between two points G_1 and G_{M_G}, where M_G is the number of points in $\boldsymbol{G}$. Let G_i denote the i-th 3D point on the geodesic. The geodesic distance, $\sigma(\boldsymbol{G})$ of $\boldsymbol{G}$ is:

$$\sigma(\boldsymbol{G}) = \sum_{i=2}^{M_G} ||G_i - G_{i-1}|| \tag{2.13}$$

However, computing $\sigma(\boldsymbol{G})$ is a time consuming step that can not exploit GPU-based parallelism. This increases the end-to-end latency of our system and causes a visible lag during projection mapping. To overcome this, we approximate the geodesic distance by computing the *Euclidean* distance between points on the surface instead, which can be computed rapidly on the GPU. The Euclidean distance $\psi(\boldsymbol{G})$ is:

$$\psi(\boldsymbol{G}) = ||G_{M_G} - G_1|| \tag{2.14}$$

Figure 2.7c shows the point $\mathbb{T}_2$ on the top edge at a Euclidean distance $\frac{1}{3}\Psi_T$ from $\mathbb{T}_1$ (the top-left corner) and $\frac{2}{3}\Psi_T$ from $\mathbb{T}_4$ (top-right corner).

Searching for 3D points in the boundary point cloud image I_B that are at specific Euclidean distances from the edge corners to assign their (u, v) parameters is slow and cannot exploit GPU-based parallelism. Instead, we compute distance maps that can be efficiently computed on the GPU. Let $S_{TL}, S_{TR}, S_{BL}, S_{BR}$ denote these distance maps containing the Euclidean distance of each 3D point in I_B from the corners top-left, top-right, bottom-left and bottom-right respectively:

$$
\begin{aligned}
S_{TL} &= \psi(I_B - T_1), \quad S_{TR} = \psi(I_B - T_{M_u}), \\
S_{BL} &= \psi(I_B - B_1), \quad S_{BR} = \psi(I_B - B_{M_u}).
\end{aligned}
\tag{2.15}
$$

Let $\Psi_T, \Psi_L, \Psi_B, \Psi_R$ denote the lengths of the top, left, bottom and right edges of the display surface. Finally, let $\omega_T(u_i), \omega_B(u_i)$ denote the cost maps of assigning parameter u_i to each 3D point along the top and bottom edges respectively and $\omega_L(v_j), \omega_R(v_j)$ denote the cost maps of assigning parameter v_j to each 3D point along the left and right edges respectively, where $v_j = \frac{j-1}{M_v - 1}$. We compute these cost maps as:

$$
\begin{aligned}
\omega_T(u_i) &= |S_{TL} - u_i \Psi_T| + |S_{TR} - (1 - u_i)\Psi_T|, \\
\omega_B(u_i) &= |S_{BL} - u_i \Psi_B| + |S_{BR} - (1 - u_i)\Psi_B|, \\
\omega_L(v_j) &= |S_{TL} - v_j \Psi_L| + |S_{BL} - (1 - v_j)\Psi_L|, \\
\omega_R(v_j) &= |S_{TR} - v_j \Psi_R| + |S_{BR} - (1 - v_j)\Psi_R|.
\end{aligned}
\tag{2.16}
$$

The markers $\mathbb{T}_i, \mathbb{B}_i, \mathbb{L}_j$ and $\mathbb{R}_j$ are the points with the minimum cost and are computed as:

$$
\begin{aligned}
t_i &= \arg\min(\omega_T(u_i)), \quad T_i = I_{PC}(t_i), \\
b_i &= \arg\min(\omega_B(u_i)), \quad B_i = I_{PC}(b_i), \\
l_j &= \arg\min(\omega_L(v_j)), \quad L_j = I_{PC}(l_j), \\
r_j &= \arg\min(\omega_R(v_j)), \quad R_j = I_{PC}(r_j).
\end{aligned}
\tag{2.17}
$$

In Figure 2.7c, T_2 is the 3D point where $\omega_T(u_2)$ is the least. Note that the cost maps, and the resulting edge markers computed using Euclidean distance will not be as accurate than when using geodesic distance. However, this difference is not visually perceptible in the final projected image. Instead, the lag caused due to computing the geodesic distance is significantly more perceptible. Hence, we trade off accuracy of the edge markers for faster computation time.

Thus, the 3D points of markers along all four edges are computed and the appropriate (u, v) parameters for each of these markers is assigned. The remaining 3D points i.e. those interior to the surface, are determined by the same method mentioned in Section 2.2.3. Control points are computed as described in Section 2.2.4.

2.2.6 Projection Mapping

To achieve projection mapping, we need to compute the projector pixel $q(u, v)$ and warp the projection image according to Equation 2.6. We sample the display surface densely by evaluating the B-spline patch at different (u, v) coordinates using the control points $\bar{\boldsymbol{C}}$ to get $\bar{\boldsymbol{Q}}(u, v)$, a 3D homogeneous point on the B-spline patch:

$$\bar{\boldsymbol{Q}}(u, v) = \boldsymbol{C}\ \boldsymbol{N}(u, v) \tag{2.18}$$

Then, this 3D homogeneous point is projected onto the 2D projector image plane using the projection matrix $\boldsymbol{M_p} \in \mathbb{R}^{3\times4}$ to get $\bar{\boldsymbol{q}}(u, v)$, the 2D homogeneous projector pixel coordinates:

$$\bar{\boldsymbol{q}}(u, v) = \begin{bmatrix} x(u, v)\ w(u, v) \\ y(u, v)\ w(u, v) \\ w(u, v) \end{bmatrix} = \boldsymbol{M_p}\ \bar{\boldsymbol{Q}}(u, v) \tag{2.19}$$

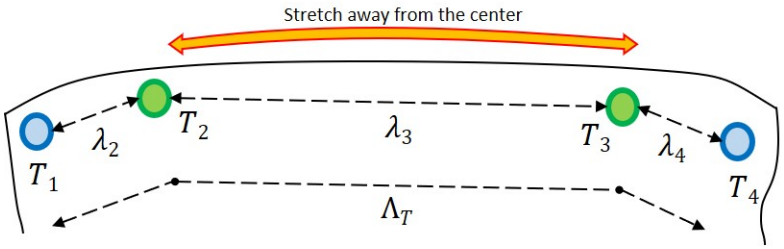
However, applying a projective transformation to a large number of 3D points as in Equation-2.19 is a time-consuming operation, increasing the end-to-end latency of our system. This is where we exploit the properties of *rational* B-splines to improve the efficiency of our system. Instead of applying a projective transformation to the 3D homogeneous points $\bar{\boldsymbol{Q}}(u,v)$, we apply the same transformation to the control points $\bar{\boldsymbol{C}}$ to get $\bar{\boldsymbol{C}}_{\boldsymbol{p}}$, the 2D homogeneous control points. Substituting $\bar{\boldsymbol{Q}}(u,v)$ from Equation-2.18 into Equation-2.19:

$$\begin{aligned} \bar{\boldsymbol{q}}(u,v) &= \boldsymbol{M}_{\boldsymbol{p}}\ \bar{\boldsymbol{C}}\ \boldsymbol{N}(u,v) \\ \bar{\boldsymbol{q}}(u,v) &= \bar{\boldsymbol{C}}_{\boldsymbol{p}}\ \boldsymbol{N}(u,v) \\ \bar{\boldsymbol{C}}_{\boldsymbol{p}} &= \begin{bmatrix} x_{00}w_{00} & x_{01}w_{01} & \dots & x_{rs}w_{rs} \\ y_{00}w_{00} & y_{01}w_{01} & \dots & y_{rs}w_{rs} \\ w_{00} & w_{01} & \dots & w_{rs} \end{bmatrix} \in \mathbb{R}^{3\times H} \end{aligned} \tag{2.20}$$

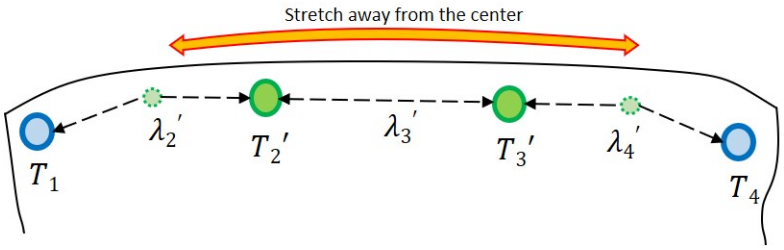
The 2D projector pixel is $q(u,v) = [x(u,v)\ y(u,v)]$. Then, we warp the projection image I_S according to Equation-2.6 to get I_T, which gets projected.

Improving B-spline Approximation

Since the display surface is deformable, it is possible that a B-spline patch with fixed parameters i.e. the degree and number of control points, may not be able to represent it accurately. Thus, we designed our system to automatically change the B-spline parameters if the current model does not accurately register the display surface geometry. We do this by measuring the registration error between the 3D display surface as measured by the depth camera and its B-spline representation using the control points computed in Section 2.2.4 at different (u,v) values. If the error is beyond a certain threshold, we increase the degree and/or number of control points. It is important to note that the maximum degree and number of control



(a) Before interpolation



(b) After interpolation

Figure 2.8: Performing length-preserving mapping. **(a)** A stretch away from the center causes $\lambda_2 \neq \lambda_3 \neq \lambda_4$. **(b)** By interpolating T_2, T_3 along the line $\{T_2, T_3\}$ to T_2', T_3' respectively, $\lambda_2' \approx \lambda_3' \approx \lambda_4' \approx \frac{\Lambda_T}{3}$, resulting in length-preserving mapping.

points of the B-spline patch is limited by the number of markers on each edge. Figure-2.15 shows projection mapping onto a deformable surface with different B-spline models: planar, quadratic, piece-wise planar and cubic.

Free-form Mapping on Stretchable Surfaces

Typically, the stretch applied to elastic surfaces is isometric i.e. the stretch is applied evenly. This increases the distances between adjacent markers along an edge evenly as well. However, when a non-isometric stretch is applied to the surface, stretching the surface unevenly, the distances between adjacent markers may not be the same. Since we do not constrain the B-spline patch to preserve distances, the projection expands to accommodate the part of the surface that gets stretched. This is known as *free-form* mapping. Figure 2.11 (top row) demonstrates the noticeable distortion of the projection when a non-isometric stretch is applied to the middle two boundary points while the corners remain fixed.

While the marker-based implementation lends itself naturally to free-form mapping, to achieve the same for the marker-less case, the edge lengths $\Psi_T, \Psi_B, \Psi_L, \Psi_R$ in Equation 2.16 are used to denote the surface edge lengths *at rest*. As a result, when a non-isometric stretch is applied, the distances between edge and corner points remain the same, but the distances between adjacent edge points do not.

Length Preserving Mapping on Stretchable Surfaces

However, we can constrain the B-spline patch to preserve distances, enabling a *length-preserving* mapping, even when non-isometric stretches are applied to the surface by moving the 3D point of each marker along each edge (except for the corners) such that the distances between all adjacent 3D points along that edge become equal. Let E_i denote the 3D point of the i-th marker on an edge $\mathbb{E}$ with number of markers M_E. Let λ_i denote the distance between adjacent markers $\{E_{i-1}, E_i\}$ and Λ_E the total estimated length of the edge $\mathbb{E}$:

$$\begin{aligned} \lambda_i &= \|E_i - E_{i-1}\|, \quad 2 \leq i \leq M_E \\ \Lambda_E &= \sum_{j=2}^{M_E} \lambda_j \\ \lambda'_E &= \frac{\Lambda_E}{M_E - 1} \end{aligned} \tag{2.21}$$

Here, λ'_E denotes the distance between two adjacent markers if all E_i's were equally spaced along $\mathbb{E}$. So, if $\lambda_i < \lambda'_E$, E_i is closer to E_{i-1} than if they were equally spaced along $\mathbb{E}$.

To perform length-preserving mapping, the goal is to find, for each non-corner point E_i, a point E'_i along the edge $\mathbb{E}$ such that $\lambda'_i \approx \lambda'_E$, where $\lambda'_i = \|E'_i - E_{i-1}\|$. For e.g. if $\lambda_i < \lambda'_E$, the point E_i is closer to E_{i-1} and should be moved away from E_{i-1} toward E_{i+1}.

Depending on λ_i, each non-corner E_i is either linearly interpolated toward E_{i-1} or E_{i+1}. If $\lambda_i < \lambda'_E$, E'_i is determined by linear interpolation along the line $\{E_i, E_{i+1}\}$, moving it away from E_{i-1} and increasing λ'_i. Similarly, if $\lambda_i > \lambda'_E$, then E'_i is determined by linear interpolation between $\{E_{i-1}, E_i\}$, reducing its length. This algorithm, that processes each segment between the markers *in sequence*, is summarized in Algorithm-1.

Algorithm 1 Length Preserving Mapping

procedure MAKELENGTHPRESERVING(EdgePoints E)

Compute edge lengths

$\Lambda = 0, M = \text{len}(E)$

for $i = 2$ *to* M **do**

  $\lambda_i = \|E_i - E_{i-1}\|$

  $\Lambda = \Lambda + \lambda_i$

end

$\lambda' = \frac{\Lambda}{M-1}$

Update edge points

for $i = 2$ *to* $(M-1)$ **do**

  if $\lambda_i < \lambda'$ **then**

    $t = \frac{\lambda' - \lambda_i}{\lambda_{(i+1)}}$

    $E_i = (1-t)E_i + tE_{i+1}$

  end

  else if $\lambda_i > \lambda'$ **then**

    $t = \frac{\lambda'}{\lambda_i}$

    $E_i = tE_{i-1} + (1-t)E_i$

  end

  $\lambda_i = \|E_i - E_{i-1}\|$

  $\lambda_{i+1} = \|E_{i+1} - E_i\|$

end

return E

end procedure

Linearly interpolating E_i's between its two neighbors in this way ensures that the resulting E_i''s are approximately equally spaced along $\mathbb{E}$. Figure 2.8 shows an example of T_2 and T_3

being moved to new positions $T_2^{'}$ and $T_3^{'}$ using the aforementioned technique. Figure 2.11 (left-column) compares the grid squares with and without length-preserving mapping.

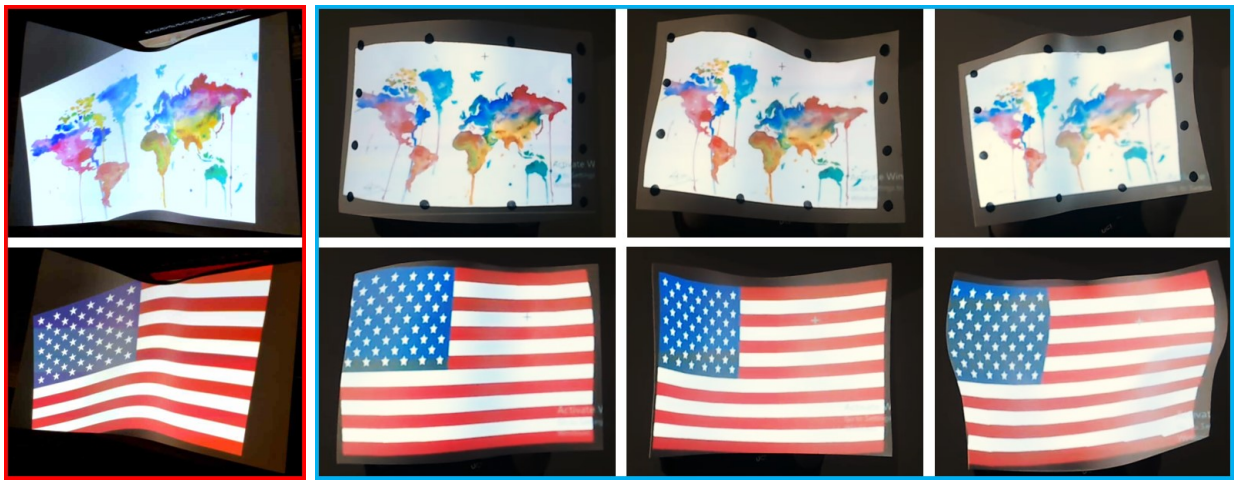
Note that the marker-less implementation lends itself naturally to length-preserving mapping when the edge lengths $\Psi_T, \Psi_B, \Psi_L, \Psi_R$ in Equation 2.16 are used to denote the *current* surface edge lengths instead of edge lengths *at rest*.

2.3 Implementation Details

Our hardware setup consists of a standard desktop workstation with an Intel Core i7-9700K CPU @ 3.6GHz with 64GB of RAM and an NVIDIA GeForce RTX 2080 SUPER GPU. We used OpenCV's CUDA implementation to track the markers, compute the distance maps and cost maps. OpenGL was used for texture-mapping and rendering the final display. Estimating (u, v) assignments for 3D display surface points and computing the B-spline control points is performed on the CPU. Both the marker-based and marker-less systems can run at close to camera frame rates.

In our implementation, we used Optoma DLP Projector. We tested our system using two different cameras: the Microsoft Azure Kinect and the CamBoard Pico-Flexx. Both cameras provide registered IR and point cloud images, while the Azure Kinect additionally provides a registered RGB image as well. However, we only use the IR and point cloud images. The Azure Kinect depth camera resolution is 320x288 @ 30fps. The Pico-Flexx provides depth images at 224x171 resolution when run at 45fps. Notice the low spatial resolutions of both cameras.

To reduce the run-time of our system, we pre-compute the coefficient matrix $\boldsymbol{A}$ (Section 2.2.4) once for different combinations of B-spline parameters i.e. degree, knot vector and number of control points. This also allows us to dynamically modify the number of markers



(a) (b)

Figure 2.9: **(a)** Projection onto a non-planar surface without any mapping applied. **(b)** Projection mapping onto marker-based (top row) and marker-less (bottom row) surface.

as well. Depending on the B-spline model parameters and the number of markers, the corresponding matrix $\boldsymbol{A}$ is used to compute the control points every iteration using linear least-squares. This means that the system must assign the 3D points to the correct (u, v) parameters used to compute $\boldsymbol{A}$. These assumptions allow fast computation of the control points.

From our experiments, for changing the B-spline parameters, the registration error threshold is set to 10%.

Computing distance maps and cost maps (see Section 2.2.5) for the marker-less display is computationally intensive. Thus, we made some important design decisions to speed up the computation without sacrificing accuracy of conforming to the deformable surface.

1. **Down-sizing distance and cost maps:** We compute the distance maps and cost maps using point cloud images with half resolution in both horizontal and vertical directions for the high-resolution Azure Kinect camera. We do not perform this step for the Pico-Flexx camera since it already has low resolution.

2. **16-bit signed integer representation:** We perform calculations on 16-bit signed integer arrays instead of floating-point arrays. If the depth camera provides point cloud data in meters, we convert it to millimeters and then store it as 16-bit signed integer. This gives a big performance boost ($\sim$100ms/frame) without compromising the shape conformity of the final display.

3. **Temporal coherency for localized calculations:** We exploit temporal coherency between successive point cloud images to further reduce the time to compute edge markers. Instead of computing the distance maps and cost maps along all edges, we compute them in local windows (31×31) centered on the marker locations in the previous depth frame. Initially, the window centers are linearly interpolated between adjacent corners. When a new depth frame is available, the distance maps and costs maps are computed for 3D points on the edges inside these windows only. These cost map "windows" are used to determine the new edge markers. The window centers are also updated to the location of the new markers in the depth camera.

4. **Noise and jitter removal:** Searching for the 3D point with the minimum cost ω (see Equation 2.17) is a time-consuming step even for small windows. Further, noise in the point cloud image introduces errors in the estimation of the edge markers, resulting in a jittery projection. Although applying a smoothing filter to the point cloud image can reduce noise, it adds extra operations and increases the latency. To avoid the additional time taken by pixel-level search and noise smoothing without adding more operations, we average 3D points inside the windows, weighted by the inverse of their costs ω i.e. a 3D point with a lower cost has a higher weight. This results in faster computation of the edge markers while smoothing out errors in the point cloud image and removing jitter.

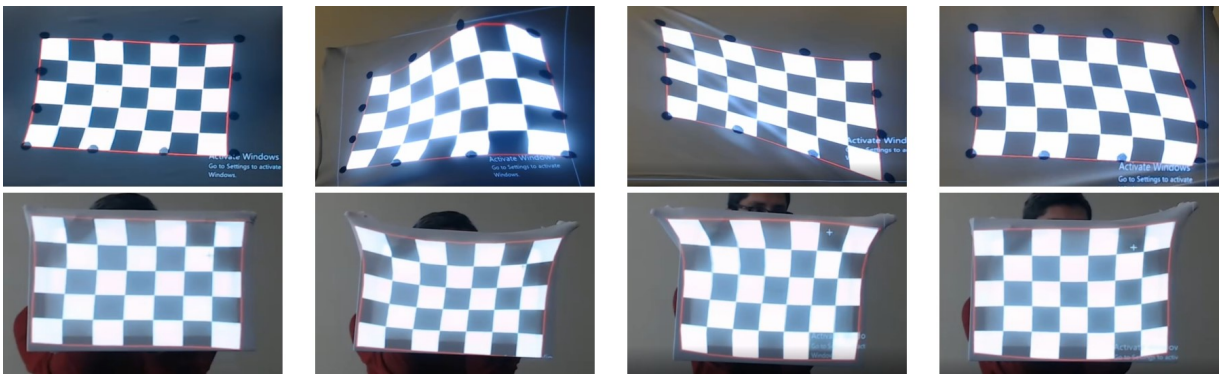


Figure 2.10: A checkerboard pattern projected onto (top-row) marker-based and (bottom-row) marker-less display surface. Note that when the surface is stretched, the tiles in the stretched region also expand.

2.4 Results

Figure-2.9b shows the results of our projection mapping system on a marked and marker-less surface undergoing different non-elastic deformations. Note how the projection conforms to the shape of the surface for all different deformations, especially when compared to Figure-2.9a, which shows the same images without any mapping applied. *Please see the accompanying video for demonstrations on dynamic movement of the surface.*

Figure 2.10 demonstrates that our method can adapt to stretchable materials for marker-based and marker-less surfaces. Similar effects are also demonstrated for stretches from both directions when projecting a checkerboard pattern (see video).

Figure 2.11 shows the difference between free-form mapping and length-preserving mapping. When the two interior points of a boundary edge are stretched away from each other while keeping the corners fixed, in the former the stretch is visually perceptible from the movement of the content. But, with length-preserving correction, the stretch becomes imperceptible giving an impression that the material is not elastic. Note that the length preserving mapping provides perceptually pleasing results for common contents like faces, scenery and buildings (Figure 2.11).

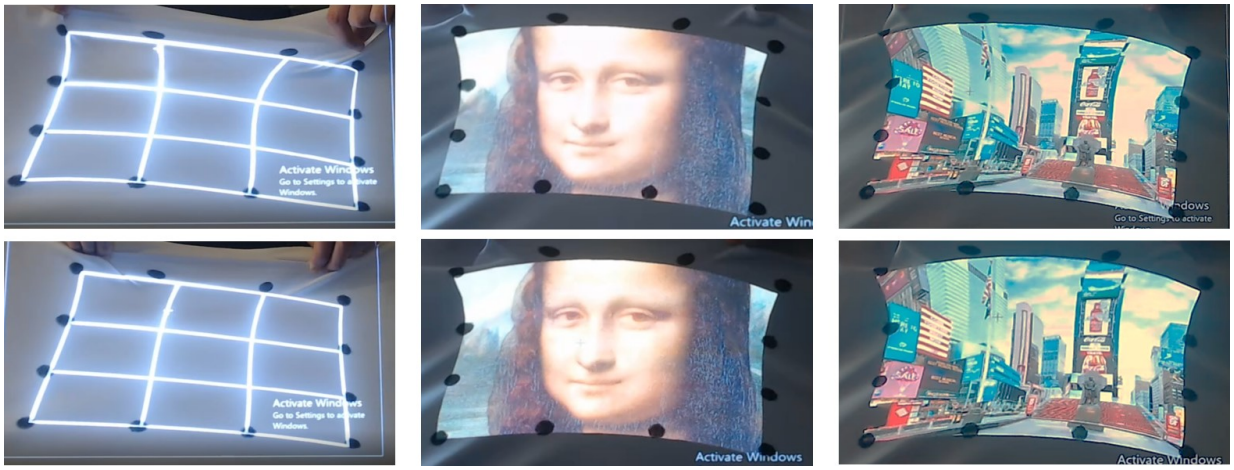


Figure 2.11: Free-form mapping (top-row) vs. length-preserving mapping (bottom-row). Note how the vertical grid lines follow the dots in free-form mapping but evenly space out with length preserving mapping. The same stretch is applied to the middle and right columns. Note the distortion caused by the stretch makes the images with free-form mapping look unnatural (e.g. Mona Lisa's forehead). Using length-preserving mapping, the images look less distorted while still conforming to the surface shape.

Figure 2.12 shows some applications of our system. The top row shows our system being used for T-shirt design. A user can project designs of their choice onto the T-shirt and interact with the fabric to see how it will look. The bottom row shows how our system can be used to view volumetric data (e.g. MRI/CT scans of the human body). By moving the display towards and away from the camera, the user can view slices through the volume. Additionally, non-planar, curved cross-sections of the surface can be used to compare different slices at the same time. *The video shows dynamic results.* Figure-2.13 shows our system being used for scientific visualization on a markerless, rigid, dynamic surface.

2.4.1 Performance and Evaluation

We evaluated our system by varying different parameters, such as the number of markers along the edges and the number of control points, and measuring **(i)** the surface registration between the B-spline patch and the actual surface geometry, and **(ii)** the end-to-end latency.

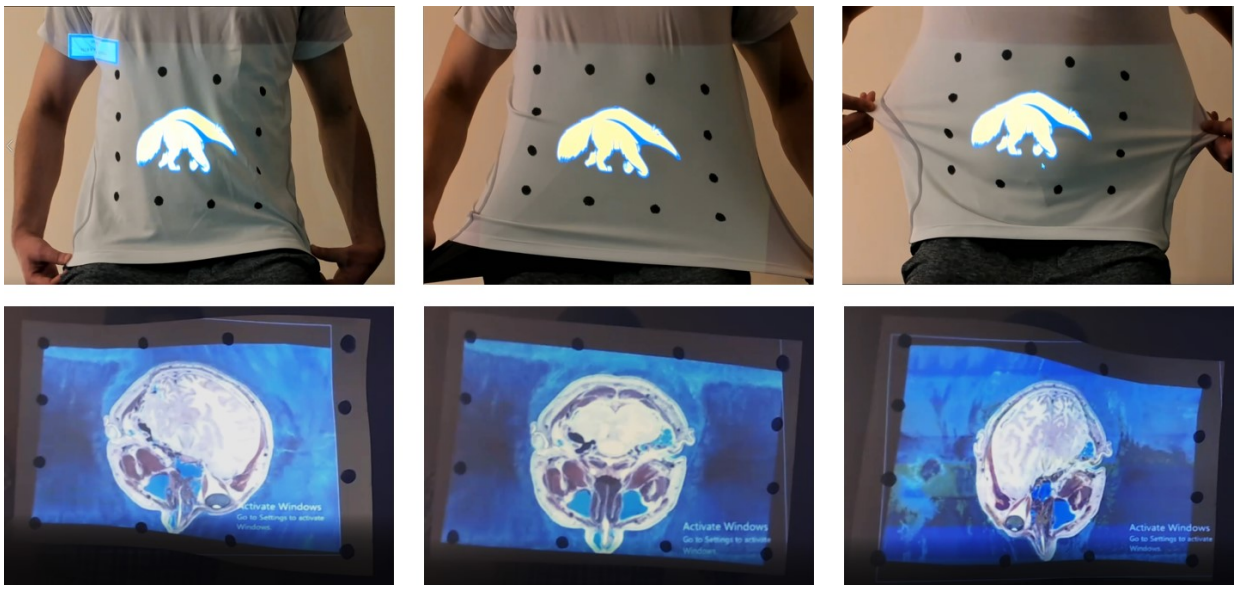


Figure 2.12: Two applications of our system. The top row shows our system being used for designing a t-shirt while the user interacts with the fabric. The bottom row shows the user deforming the surface to view volumetric data at curved cross-sectional slices and comparing them.

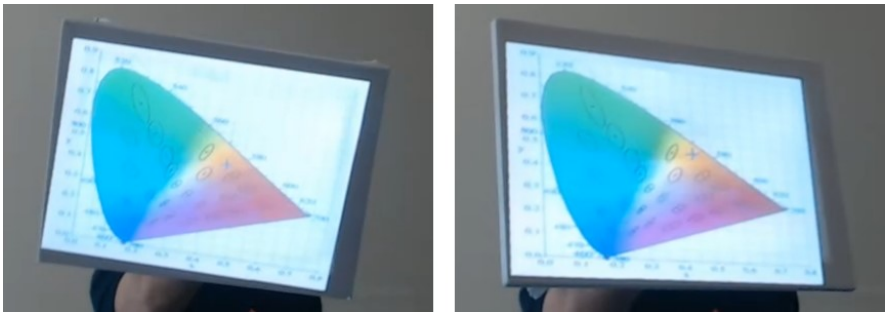
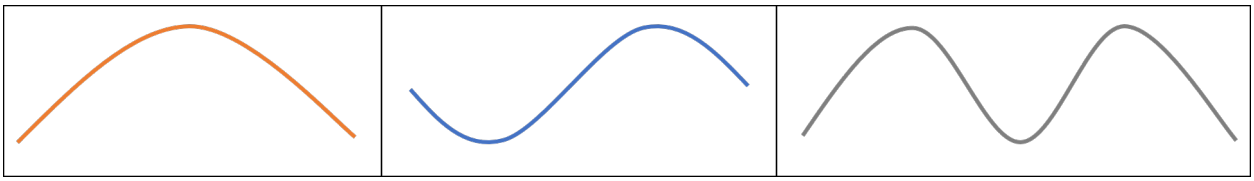


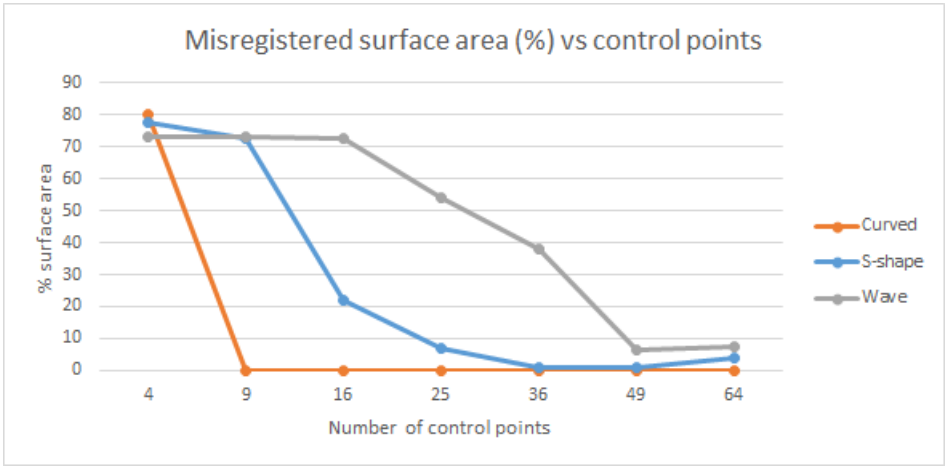
Figure 2.13: Marker-less display on a rigid, dynamic surface.

Furthermore, we also tested the accuracy of the internal 3D points for various shapes since that directly affects the B-spline patch that gets computed.

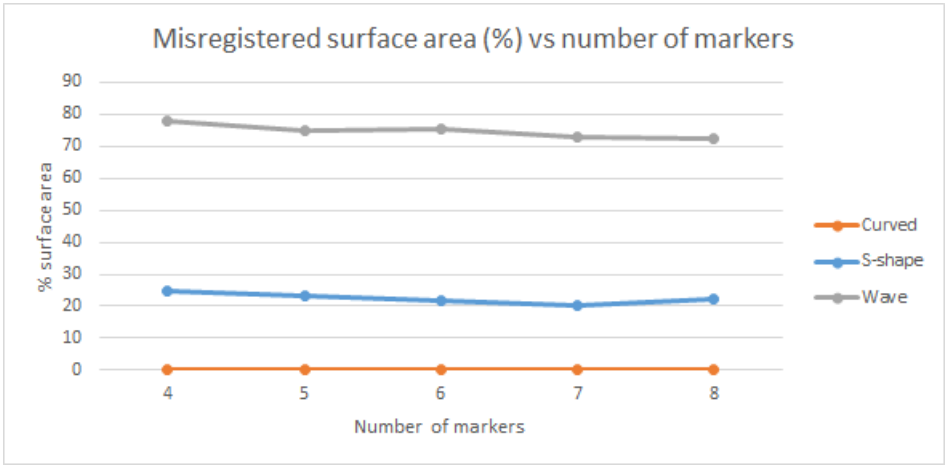
To evaluate surface registration, we compute the distances between the B-spline 3D points and the point cloud image. The misregistered surface area is the ratio of the number of points whose distance is greater than 10mm to the total number of points. We measured the surface misregistration for three different shapes: a curve, an S-shape and a wave as shown in Figure-2.14. Figure-2.14d shows the effect of increasing the control points on the surface misregistration for all three shapes, while keeping the number of markers constant at



(a) Curve (b) S-shape (c) Wave



(d)



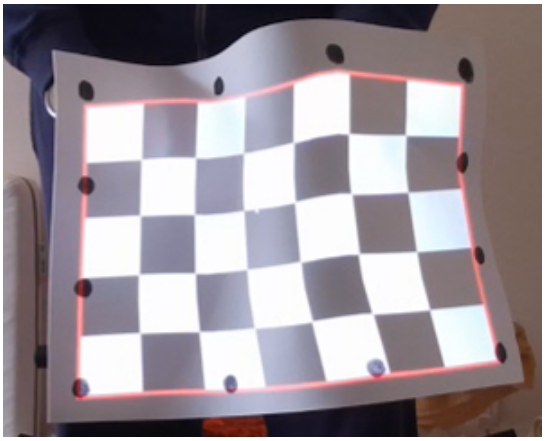
(e)

Figure 2.14: Effect of various parameters on the surface registration. **(a)-(c)**: the surface profiles used to study the effect on the registration error by **(d)** the number of control points, and **(e)** the number of markers.

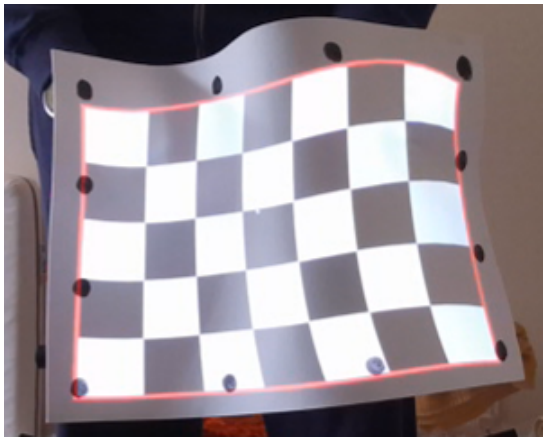
$(M_u, M_v) = (8, 8)$. For the curve shape, four control points are not sufficient to represent it, therefore we see a high ($\sim$80%) misregistration. However, increasing the control points to 9 or more results in excellent registration since the B-spline patch can represent the shape accurately. We see a similar significant decrease in misregistration for the S-shape: from

(a) Planar ($H = 4$)

(b) Quadratic B-spline ($H = 9$)



(c) Piece-wise planar ($H = 16$)



(d) Cubic B-spline ($H = 16$)

Figure 2.15: Projection mapping on a surface deformed as an S-shape while changing the B-spline model. Notice that although **(c)** and **(d)** conform to the surface well and have the same number of control points ($H = 16$), the projection mapping is not identical. This is visible at the top edge, where **(c)** has visible discontinuities compared to **(d)**.

80% with 9 control points to 20% with 16 control points. As for the wave, since it is more complex, it requires an even larger number of control points (49) to represent the shape accurately.

Figure-2.14e shows the effect of increasing the number of markers on the surface misregistration for the three shapes. In this experiment, we modeled the surface using a degree-3 B-spline patch with 16 control points while changing the number of markers. Since 16 control points are sufficient to represent the curved shape, its surface misregistration is close to zero.

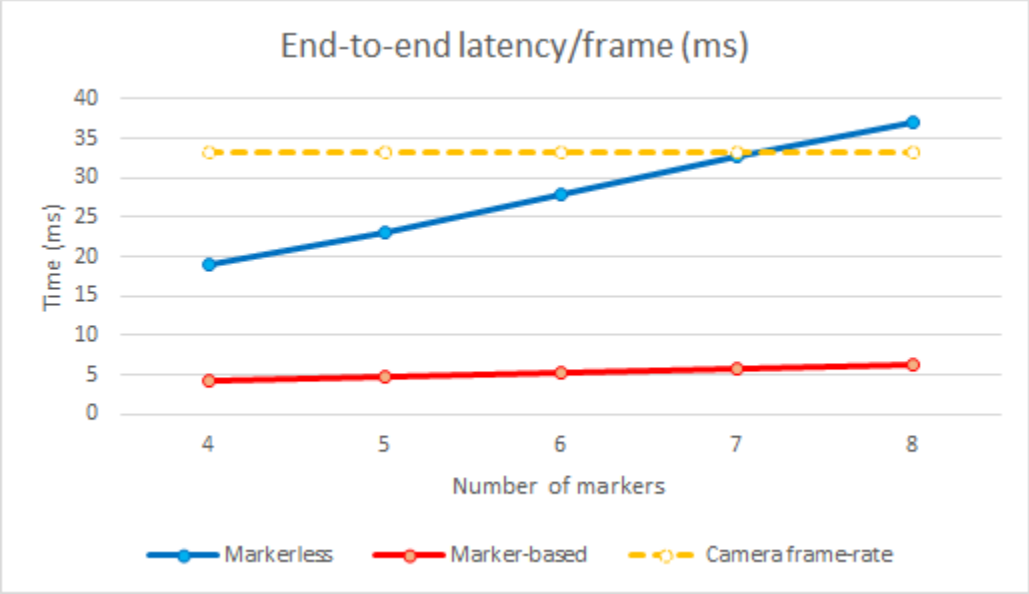


Figure 2.16: Effect of the number of markers on the end-to-end latency for marker-based and marker-less systems.

However, for the S-shape and the wave, we only see a slight decrease in the misregistration as we increase the number of markers. This is because it is the number of control points, rather than the number of samples, that limits the ability of the B-spline patch to represent those shapes. Therefore, to improve surface registration, the number of control points must be increased, an observation that is confirmed by Figure-2.14d.

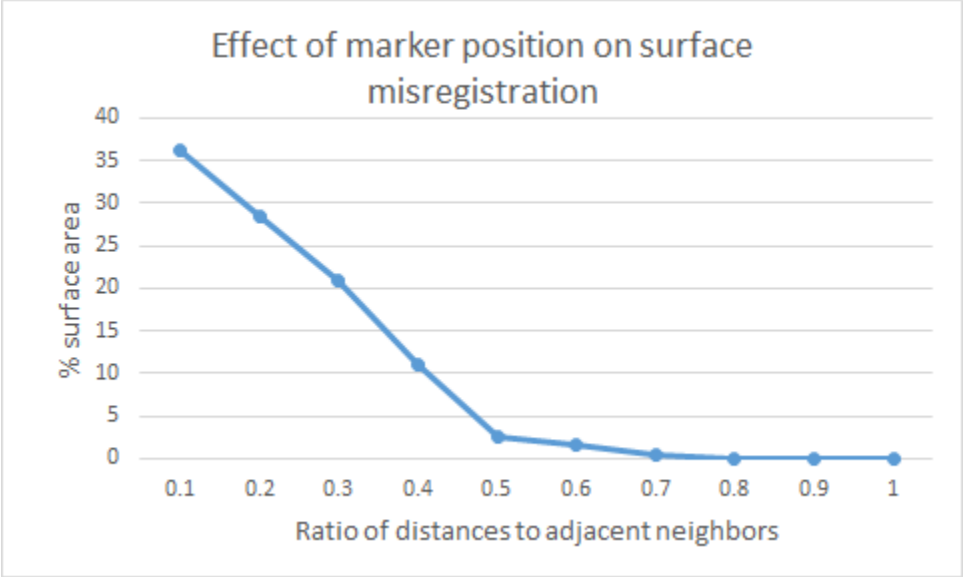
The surface misregistration can be visually observed as well. Figure-2.15 shows projection mapping on an S-shape surface while changing the number of control points. Notice that both planar and quadratic models (Figure-2.15 (a)-(b)) are not rich enough to adequately represent the surface shape and therefore, do not conform to the boundaries well. This is also confirmed by Figure-2.14d, where they have a high misregistration for the S-shape. However, the piece-wise planar and cubic models (Figure-2.15 (c)-(d)) are able to represent the shape well and hence, projection mapping conforms to the boundaries. Also notice that while both the piece-wise planar and cubic models have the same number of control points ($H = 16$), they represent two different shapes since they are of two different degrees ($(n, m) = (1, 1)$ and $(3, 3)$ respectively). This is why the projections do not appear identical.

The difference is especially visible at the top edge, where the piece-wise planar model has visible discontinuities.

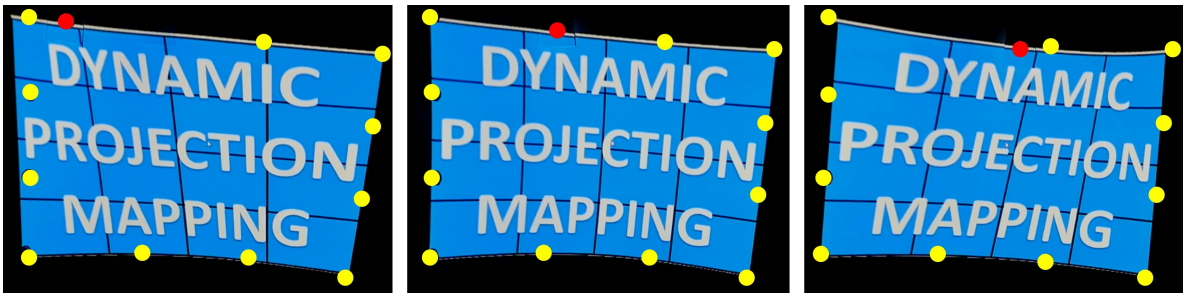
Figure-2.16 shows the end-to-end latency, i.e. the time from when a new camera frame is available till the final rendering and display, for both the marker-based and marker-less systems, for different number of edge markers. Note that increasing the number of markers results in an insignificant increase in the latency for the marker-based system, which consistently runs close to 5 ms/frame. On the other hand, the number of markers has a significant effect on the end-to-end latency for the markerless system. This is because increasing the number of markers increases the number of cost maps that need to be computed, a computationally intensive step. Despite that, the markerless system performs faster than the camera frame-rate (33.33 ms) for up to seven markers on each edge. As such, the end-to-end latency becomes limited by the camera frame rate.

Figures-2.14 and 2.16 also reveal an interesting trade-off of our system. Figure-2.14 suggests that one should increase the number of control points to achieve better surface registration, but that can only be done by increasing the number of markers. However, the number of markers has a direct impact on the end-to-end latency, as shown in Figure-2.16, especially for the marker-less system, which increases at a rate of 4.5ms/marker. Therefore, the number of markers required depends on the demands of the application in terms of speed and accuracy of registration. In general, $5 \leq (M_u, M_v) \leq 7$ and $25 \leq H \leq 49$ leads to excellent performance (in both speed and accuracy) for the marker-based and marker-less systems for a large variety of applications.

Our marker-based system requires the user to place markers on the surface. Therefore, we investigated how accurate the user must be when placing these markers. For this experiment, the surface had four markers on each edge, was shaped like a curve (Figure-2.14a) and modeled by a quadratic B-spline (with 9 control points). We moved the marker T_2 towards T_1 and measured the distances between T_2 and its two adjacent neighbors. For each such



(a) Surface misregistration for various marker positions

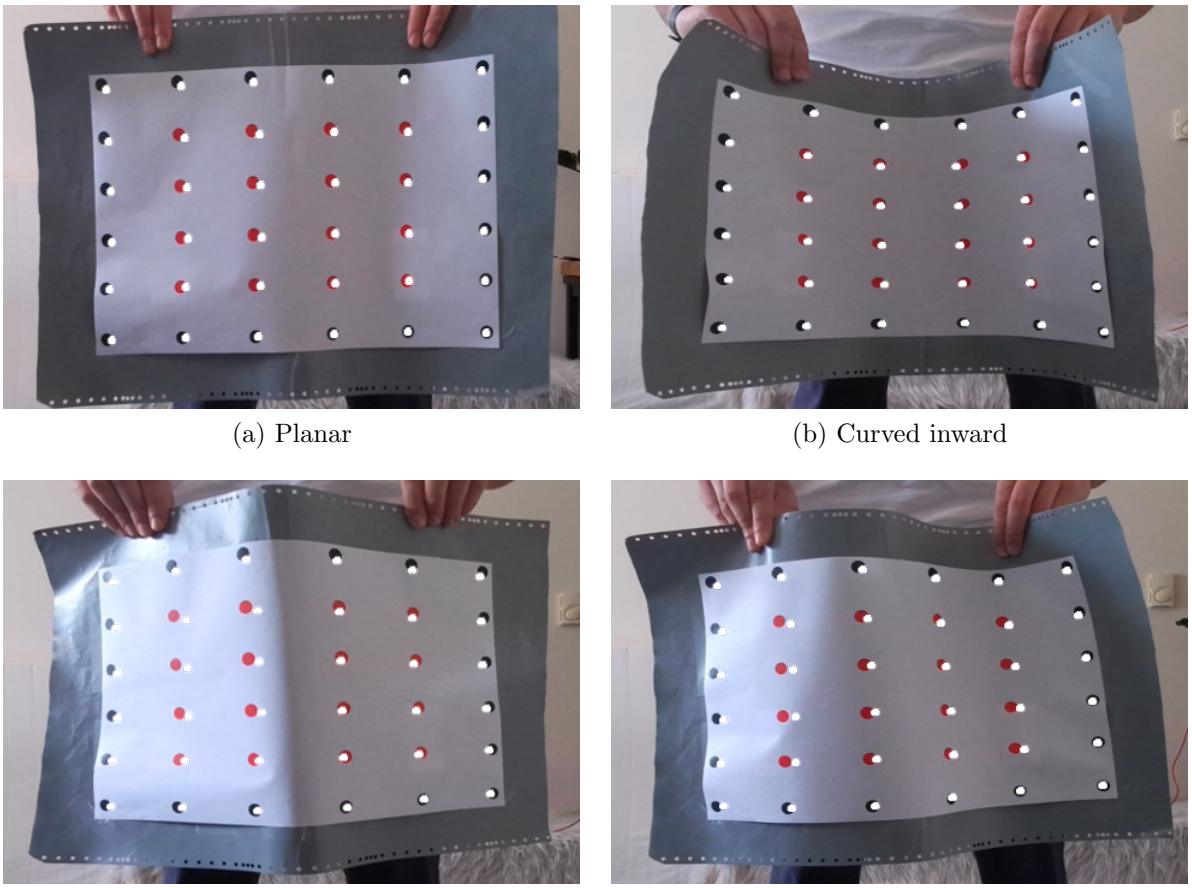


(b) Effect on projection mapping on marker placement

Figure 2.17: **(a)** Effect of marker placement on the surface area misregistration. When the distance ratio (*x-axis*) is 1, the marker is at its ideal position while smaller values mean increasing deviation from the ideal. **(b)** shows the projection when moving the marker T_2 (in red) from its ideal position *(middle)* towards T_1 *(left)* or towards T_3 *(right)*.

position, we calculated the surface misregistration as a function of the ratio of the larger distance to the smaller distance. A distance ratio of 1 means T_2 is placed directly between its two neighbors (its ideal position), whereas a value of 0.5 means T_2 is twice as far away from one neighbor compared to the other.

The effect of marker placement on the surface misregistration is shown in Figure-2.17a. Notice that as the distance ratio decreases (i.e. larger deviation from ideal position), the surface misregistration increases. This can be visibly confirmed in Figure-2.17b, which shows the projection becoming quite distorted when T_2 is moved too close to either of its neighbors



(a) Planar (b) Curved inward

(c) Curved outward (d) Sinusoidal wave

Figure 2.18: Error when computing internal 3D points. The red and black markers (ground truth points) are compared with the white overlayed markers (points estimated by our system). The error for each shape is **(a)** 2.59mm. **(b)** 2.27mm, **(c)** 4.99mm, **(d)** 6.59mm.

(left and right) compared to when it is at its ideal position (middle). Figure-2.17a also shows that even if a marker is placed twice as close to one of its neighbors compared to the other (a ratio of 0.5), the surface misregistration still remains relatively low ($< 5\%$). This means that, for a standard A4-size paper (21.$\times$29.7 cm), a marker needs to be within $\sim$3cm of its ideal position to yield good registration. This gives the user significant flexibility when placing markers, even if they are not precisely equidistant from each other.

To measure the accuracy of the internal 3D points, we compute the average error between the points estimated by our system and their ground truth points. The ground truth internal points are obtained by a display surface with the full marker grid printed. We measured this

error for different shapes of the surface. As shown in Figure-2.18, the largest error is around 6.59mm, which is not perceivable by viewers.

Chapter 3

Auto-calibration of Single PDC

In the previous chapter, we explained a D-sAR system that does not use any custom solutions. A consumer projector is paired with a off-the-shelf RGB-D camera (e.g. Kinect) to create a PDC unit which is used to achieve D-SAR that adapts the projection on the dynamic, deformable surface. The RGB-D camera captures the depth of the moving deformable surface which is leveraged by the projector to appropriately distort the display content to continuously conform to the dynamically changing surface shape. However, such a D-SAR system needs to be fully calibrated (i.e. the intrinsic and extrinsic parameters of the devices need to be estimated accurately) before it is deployed. This calibration is usually done using a static rigid object. Therefore, anytime the projector or camera is moved with respect to the other device, this static rigid object has to be brought back for recalibration.

In this chapter, we explain a method that achieves an automated projector-camera pair calibration using the dynamic, deformable surface itself. We project a set of ArUco markers on the moving surface which is captured by the RGB-D camera. We use an optimization technique that starts by computing a coarse estimate of the projector instrinsics and extrinsics, and refines them over a number of frames to get accurate intrinsic and extrinsic for both the

projector and camera. Since this removes the use of any static props for calibration, anytime the projector-camera pair is moved it can be recalibrated quickly using the same dynamic and deformable surface that is being used for projection. Furthermore, our method does not require the camera and projector to be synchronized either.

3.1 Related Work

Projector-camera calibration: There has been enormous work done in the domain of projector-camera calibration. Not all of them are used for DPM, but are germane to our work nevertheless. Most prior works use an RGB camera. Some, like ours, use an RGB-D camera (e.g. Kinect) and is considered as a single unit since the RGB and the Depth cameras are calibrated by the manufacturers to provide registered depth and color images. Further, there are different types of calibration methods. In some methods [3, 134], projector calibration is dependent on the results of camera calibration and therefore error in camera calibration is propagated to the projector parameters. However, other methods [76, 135] perform camera and projector calibration independently.

Single Projector-RGB camera System: Most single projector-camera systems use structured light scanning of a known calibration object to establish pixel correspondences between the projector and camera followed by calibration [76, 3, 29, 30, 135, 106, 52, 17, 95, 134]. The structured light scanning might be a single-pattern scanning or multi-pattern scanning. In [17, 29, 30, 52, 76], the calibration object is a checkerboard pattern printed on a planar surface. The checkerboard is used to calibrate the camera using Zhang's method [137]. The projector is treated like an inverse camera and is calibrated like a camera but using the projector-camera pixel correspondences generated from the structured light scan.

Methods that use only planar projection surface [3, 106, 135] need not rely on structured light scanning to generate pixel correspondences. Instead, they estimate a homography between the camera and projector using a rigid planar calibration board by pre-warping the projected image to align with a printed pattern on the calibration board. For example, Audet et.al [3] capture images of a checkerboard grid of specific markers called BCH markers [122] printed on a planar board. By detecting the corners of the markers, they are able to compute the homography between the camera and the calibration board. They also compute another homography to pre-warp an image of another BCH marker checkerboard grid such that the projected markers align with the empty spaces in the checkerboard marker grid on the board. Using these homographies, they perform the projector-camera calibration. Yang et.al [135] compute a pre-warp using a random dot marker grid, while Shahpaski et.al [106] perform both geometric and radiometric calibration by computing a pre-warp for a planar checkerboard pattern.

Other methods [131, 65, 134] lift the restriction of a planar calibration object, instead relying on objects of arbitrary geometry to perform calibration. This also allows these methods to project on non-planar surfaces.

Single Projector, RGB-D camera System: Resch et.al [95] use structured light scanning and a precise 3D mesh of an arbitrary object (that can be obtained by a laser scan) to iteratively refine the calibration parameters of the projector-camera system.

Multi-Projector, Single RGB camera System: Using measured priors (e.g. aspect ratio, size of radius) for specific non-planar surfaces (e.g. vertically extruded surfaces, swept surfaces or spherical surfaces) [99, 101, 103, 100] use a single uncalibrated camera to find the device parameters in a multi-projector system. The knowledge of the particular type of the shape and its measured priors help in constraining the optimization thereby removing the use of any special calibration pattern.

Multi-Projector, Single RGB-D camera System: Kurth et.al [59] use a depth camera to calibrate their multi-projector rig used for dynamic projection mapping. They use a precise 3D mesh of an arbitrary complex object (e.g. a human bust) recovered via a laser scan to constrain the optimization process to achieve the parameters for the devices.

Multi-Projector, Multi RGB-Camera Systems: When using multi-projector systems on planar screens, explicit calibration of the devices are avoided by using homography based methods [14, 7, 6]. When handling non-planar surfaces, the most common way to calibrate multi-projector systems has been to use one of the single projector-camera calibration methods to calibrate different device pairs separately and then connect them in a tree like fashion to calibrate all the devices [92, 91]. However, errors creep in due to each device being calibrated separately and getting sub-pixel accuracy in registration becomes impossible. Further, anytime the devices are moved, a new calibration has to be performed bringing in the rigid 2D/3D calibration pattern.

Comparison with prior work

For creating a comprehensible display on a moving deformable surface, a D-SAR system requires automatic and accurate calibration of the PDC unit. [131, 118] are the only works that achieve an automated calibration (without the use of specific 2D or 3D calibration objects) of multiple projectors and RGB cameras in a system that projects on a complex rigid 3D shape. In this chapter, we explain a method that achieves the same for moving display surfaces.

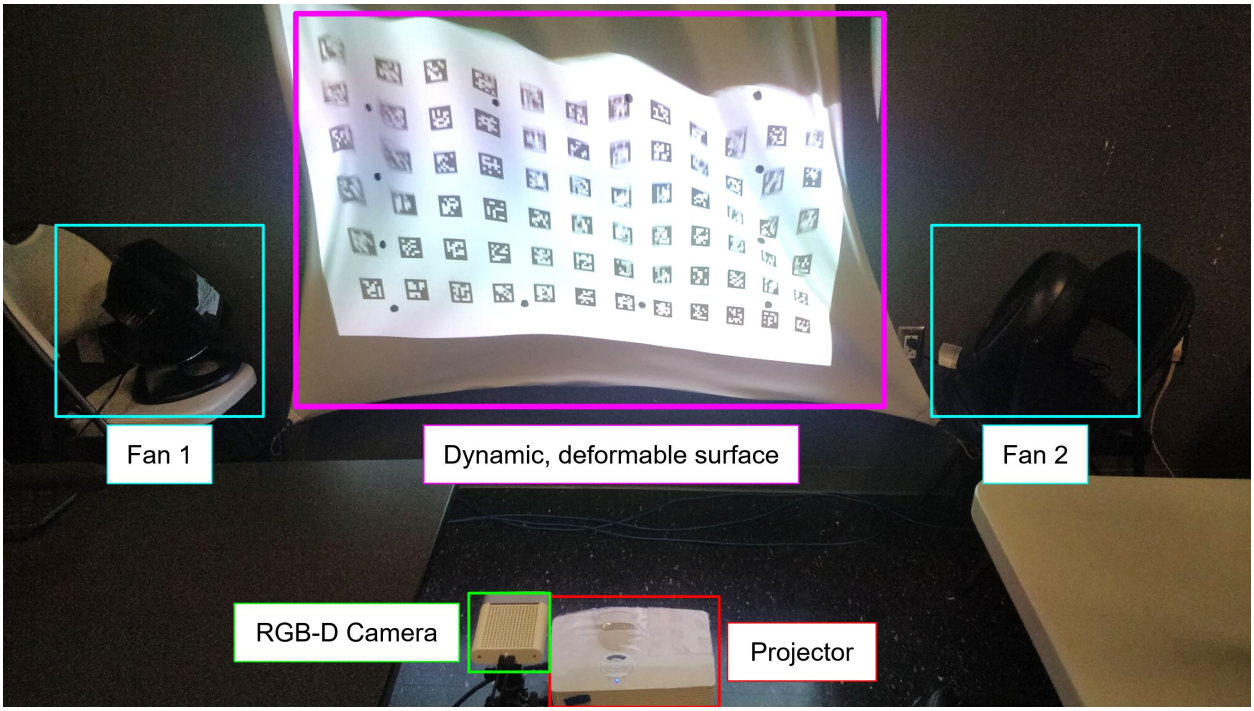


Figure 3.1: Our setup, comprising a RGB-D camera and a projector positioned towards a dynamic, deformable surface.

3.2 Method

Our setup consists of a projector and an RGB-D camera positioned to project on and capture a surface respectively that may be dynamic and deformable (Figure 5.1). For screen we use an elastic tendo screen and for movement we use two fans from two sides that can be set to three different speeds from slow to fast. We assume that the RGB-D camera provides us with registered RGB image and depth mesh, in keeping with the norm for consumer RGB-D cameras. Further, we assume a rough estimate of the projector focal length, which can be estimated using methods such as [9, 28], or by using the calibration toolbox provided by Moreno et al. [76]. While [76] requires a planar calibration pattern to estimate the focal length, [28] works for non-planar 2D-3D correspondences. Focal length can be estimated just once for a projector of the same brand and make, before adding the projector in the DPM setup and need not be re-estimated after that. For the camera, usually metadata or

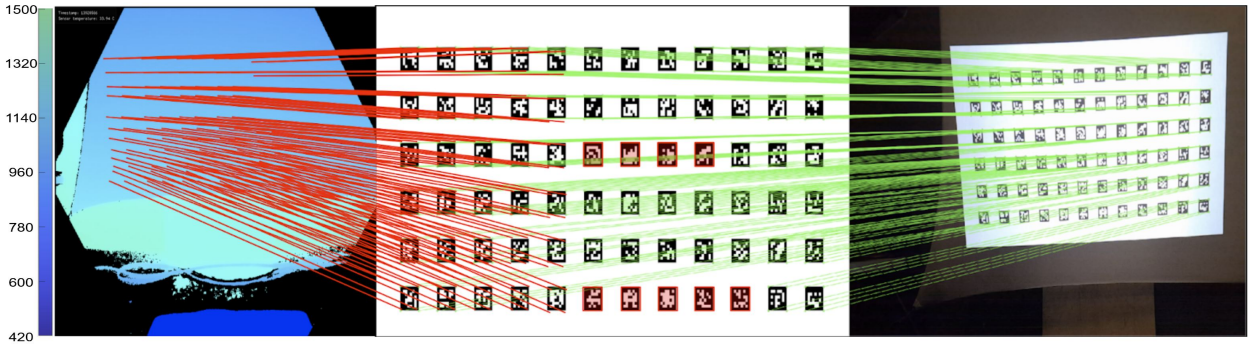


Figure 3.2: (*Left*) The depth map (in mm), (*Middle*) the projected image and (*Right*) the camera image. The corresponding pixels at the corners of the ArUCo markers between the projector, depth map and camera image are shown by the red and green lines respectively. Markers that were not detected in the camera image are highlighted in red.

API provides the intrinsics, which we can used as an initial estimate for quick convergence of optimizations, although we do not require it.

3.2.1 Background

We model both the projector and the camera with a pinhole camera model extended with radial and tangential distortion. We assume that the RGB-D camera's center of projection (COP) is at the origin, its pose is aligned with the XYZ axis with the principal axis being along the positive Z axis. Let $c \in \mathbf{R}^2$ represent the image of the 3D point $d = [X\ Y\ Z]^T \in \mathbf{R}^3$ in the camera image plane. Then d and c are related by $c = K_c \cdot \Lambda(\tilde{c})$ where $\tilde{c} = [\tilde{c_x}\ \tilde{c_y}\ 1]^T = [\frac{X}{Z}\ \frac{Y}{Z}\ 1]^T$ and $K_c \in \mathbf{R}^{3\times3}$ is the camera intrinsic matrix and $\Lambda(\cdot) \in \mathbf{R}^{3\times1}$ is the lens distortion function. The (3×4) extrinsic parameter matrix is identity. Mathematically, K_c and $\Lambda(\tilde{c})$

are given by:

$$K_c = \begin{bmatrix} f_x & 0 & n_x \\ 0 & f_y & n_y \\ 0 & 0 & 1 \end{bmatrix}; \tag{3.1}$$

$$\Lambda(\tilde{c}; D_c) = \begin{bmatrix} \tilde{c} \cdot \rho(\tilde{c}) + \tau(\tilde{c}) \\ 1 \end{bmatrix}; \tag{3.2}$$

$$\rho(\tilde{c}) = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.3}$$

$$\tau(\tilde{c}) = \begin{bmatrix} 2k_4 \tilde{c_x} \tilde{c_y} + k_5(r^2 + 2\tilde{c_x}^2) \\ k_4(r^2 + 2\tilde{c_y}^2) + 2k_5 \tilde{c_x} \tilde{c_y} \end{bmatrix}; \tag{3.4}$$

$$r^2 = \tilde{c_x}^2 + \tilde{c_y}^2; \tag{3.5}$$

where (f_x, f_y) is the focal length, (n_x, n_y) is the camera principle point, $D_c = \{k_1, k_2, k_3, k_4, k_5\}$ are the distortion coefficients where (k_1, k_2, k_3) are radial distortion coefficients and (k_4, k_5) are the tangential distortion coefficients.

We assume the same pinhole camera model for the projector. However, we assume its rotation and translation with respect to the camera to be $R_p \in \mathbf{R}^{3\times3}$ and $T_p \in \mathbf{R}^{3\times1}$ respectively. Let p denote the projector pixel that illuminates the 3D point d. Then:

$$d' = \begin{bmatrix} X' & Y' & Z' \end{bmatrix}^T = R_p \cdot d + T_p \tag{3.6}$$

$$\tilde{p} = \begin{bmatrix} \tilde{p_x} & \tilde{p_y} & 1 \end{bmatrix}^T = \begin{bmatrix} \frac{X'}{Z'} & \frac{Y'}{Z'} & 1 \end{bmatrix}^T \tag{3.7}$$

$$p = K_p \cdot \Lambda(\tilde{p}; D_p) \tag{3.8}$$

where K_p, D_p are the projector intrinsic matrix and distortion coefficients respectively.

3.2.2 Image Acquisition

Calibration of the project-camera unit requires finding correspondences between 3D coordinate d_i, the 2D camera pixel c_i that images it, and the corresponding 2D projector pixel p_i that illuminates it. Traditional systems usually use multi-frame structured light scan (SLS) methods to find the correspondences between c_i and p_i from which d_i is deciphered. However, in the proposed method, RGB-D camera directly provides us the correspondence between d_i and c_i. And, we use a pattern akin to single-shot SLS patterns [29, 30] to link these correspondences to p_i.

Our pattern is based on ArUCo markers [20]. Each ArUCo marker comprises of a black and white square with a unique pattern and ID. When captured by a camera, an image containing the ArUCo markers can be decoded to find the IDs and the four corners of the markers. We project a *sequence of grids of ArUCo markers* with known IDs and corners onto a dynamic, deformable surface. The RGB-D camera captures the projected sequence and the ArUCo marker detection algorithm detects the locations and IDs of the markers in that image and establish the correspondence between the detected c_i, its corresponding d_i and the known location of the detected points in the projector p_i. (Figure 3.2).

Each image in the sequence is projected for several frames. Since dynamic projection surface can move during these frames, even when using a single grid of ArUCo markers, we can obtain a large number of $c_i \rightarrow d_i$ correspondences. Across different grids, we design the location of the corners and the IDs of the ArUCo markers to be different to increase the number of correspondences even further. Unique ID across grids also prevents incorrect correspondences due to asynchronous projection and capture of the patterns if the camera and projector are not synchronized. We perform marker corner detection at sub-pixel accuracy in all the images captured by the RGB camera and use the depth from the depth camera at the registered location to establish the correspondence between c_i, d_i and p_i.

3.2.3 Camera Calibration

As mentioned before, our camera extrinsic matrix is identity and we need to only determine the camera intrinsic matrix and distortion coefficients using the correspondences. Let $\{c_i^t, d_i^t\}$ denote the i-th correspondence at time t. Let K_c^t and D_c^t denote the intrinsic matrix and distortion coefficients at time t. We optimize for K_c, D_c by minimizing the reprojection error E_c^t given by:

$$\arg\min_{K_c^t, D_c^t} E_c^t = \frac{1}{2}\sum_i (|c_i^t - K_c^t \cdot \Lambda(\tilde{c_i^t}; D_c^t)|)^2 \tag{3.9}$$

$$\tilde{c_i^t} = \frac{d_i^t}{Z_i^t} = \begin{bmatrix} X_i^t/Z_i^t & Y_i^t/Z_i^t & 1 \end{bmatrix}^T \tag{3.10}$$

The camera parameters are optimized every frame using per frame correspondences, initializing the optimization with the parameters from the previous frame. For the first frame $t = 1$, we assume the distortion coefficients to be zero, and compute the intrinsic matrix K_c^1 directly from the 2D-3D correspondences. If the camera API provides an initial estimate of the intrinsics, we only use the intrinsic matrix and ignore the distortion coefficients. We assume the optimization for a frame t has converged when any one of the two conditions are met: (1) the number of iterations is greater than 1000, or (2) the absolute difference between the reprojection errors in two successive iterations is less than a threshold, i.e. $|E^t - E^{t-1}| \leq \Delta_c$.

3.2.4 Projector Calibration

For the projector, we need to estimate both the projector intrisics and the extrinsics (i.e. rotation and translation with respect to the camera). Let $\{p_i^t, d_i^t\}$ denote the i-th correspondence between a 2D projector coordinate p_i^t and 3D point d_i^t at time t. Let K_p^t, D_p^t denote the intrinsic matrix and distortion coefficients, and R_p^t, T_p^t denote the extrinsic rotation and

translation at time t. We optimize for K_p, D_p, R_p, T_p by minimizing the reprojection error E_p^t as

$$\arg \min_{K_p^t, D_p^t, R_p^t, T_p^t} E_p^t = \frac{1}{2} \sum_i (|p_i^t - K_p^t \cdot \Lambda(\tilde{p_i^t}; D_p^t)|)^2 \tag{3.11}$$

$$\tilde{p_i^t} = \begin{bmatrix} \tilde{X_i^t}/\tilde{Z_i^t} & \tilde{Y_i^t}/\tilde{Z_i^t} & 1 \end{bmatrix}^T \tag{3.12}$$

$$\begin{bmatrix} \tilde{X_i^t} \\ \tilde{Y_i^t} \\ \tilde{Z_i^t} \end{bmatrix} = (R_p^t \begin{bmatrix} X_i^t \\ Y_i^t \\ Z_i^t \end{bmatrix} + T_p^t)] \tag{3.13}$$

We optimize the projector parameters using Equation-3.11 considering correspondences from each frame separately, moving on to the next frame when convergence for the current frame is reached. Optimizing likewise across multiple frames averages out random noise providing an accurate calibration despite noise in the depth camera.

However, Equation-3.11 is sensitive to initialization of the projector parameters. Poor initialization can result in the optimization getting stuck in local minima, resulting in a projection that will align with the surface only within a narrow depth range. For DPM systems where the surface may be moving in a large range, this can be problematic. Therefore, we devise a new method to obtain an initial estimate of the projector parameters to initialize the optimization in Equation-3.11 assuring robust convergence in optimization.

Coarse Projector Parameter Estimation: Pixels on any straight line in the projector coordinates correspond to a plane in 3D (Figure 3.3). Leveraging this property, we fit a plane through the 3D points corresponding to 2D pixel locations of ArUCo marker corners along

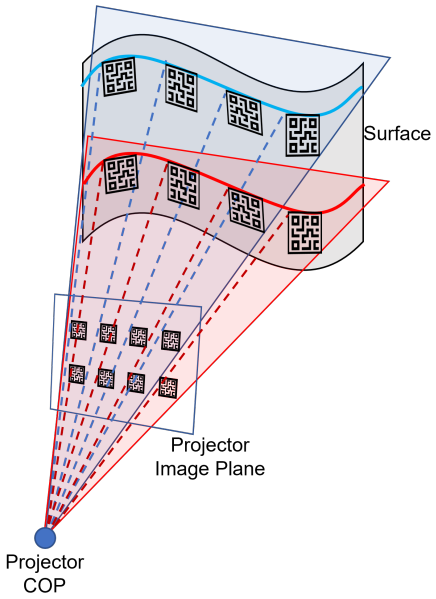


Figure 3.3: Illustration of plane fitting through 3D points of ArUCo marker corners on the surface that correspond to the same projector pixel rows. Intersecting all row planes gives the X-axis of the projector. Intersecting all column planes gives the Y-axis of the projector. The Z-axis is the cross product of the two. The projector COP is obtained by intersecting all row, column and diagonal planes.

every row, column and diagonal across all grids of markers. Intersecting all the planes along rows results in a line whose direction corresponds to the projector's X-axis. Intersection of all planes along columns results in a line along the projector's Y-axis. Taking the cross product of these two vectors gives the Z-axis. Thus, we determine the projector orientation R_p^0 with respect to the camera. The projector center-of-projection (COP) T_p^0 is estimated by finding the intersection of all the planes along the row, column and diagonals. Using this initial estimate of the projector orientation and COP, we compute K_p^0, assuming zero distortion.

Projector Parameter Refinement: We begin the optimization in Equation-3.11 with this more reasonable initialization. Note that if we assume that the 3D location of the projected points corresponding to markers along any row, column or diagonal are collinear, it will present a degenerate case to the plane fitting method. Therefore, this method will fail for static planar projection surface. In our setup (Figure 5.1) using a hanging fabric screen

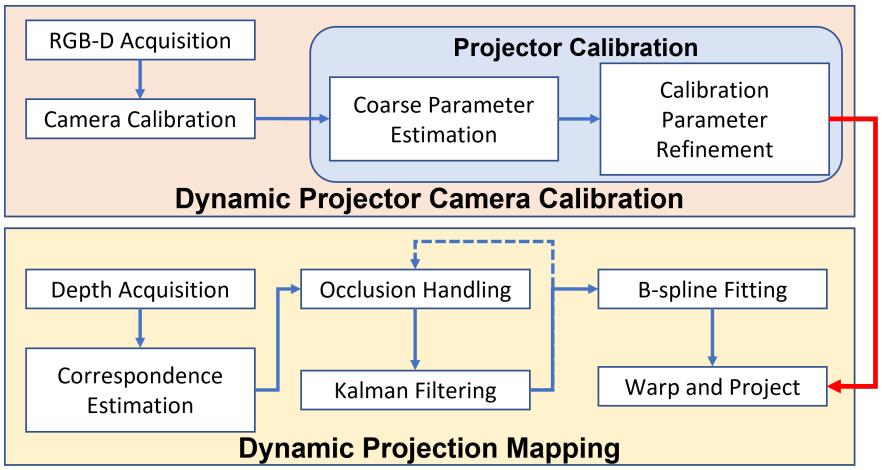
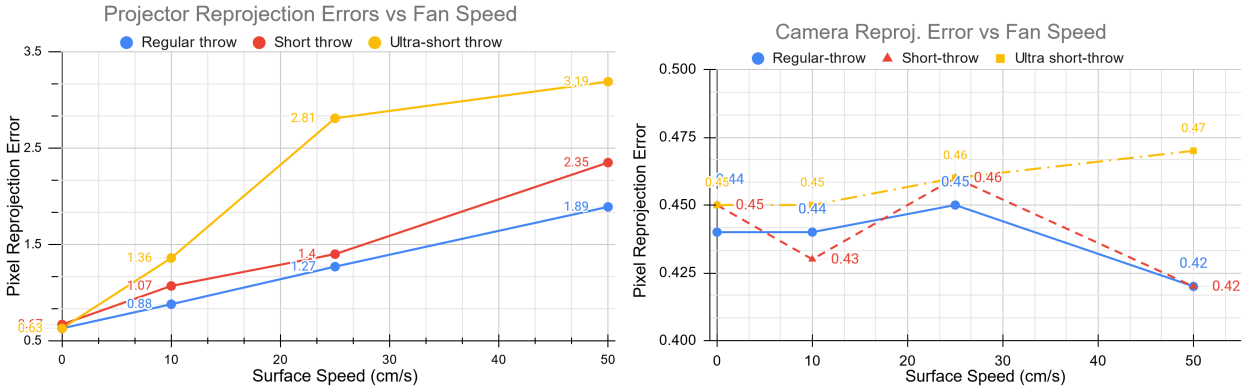


Figure 3.4: The flowchart of the proposed system.



(a) Projector reprojection error

(b) Camera reprojection error

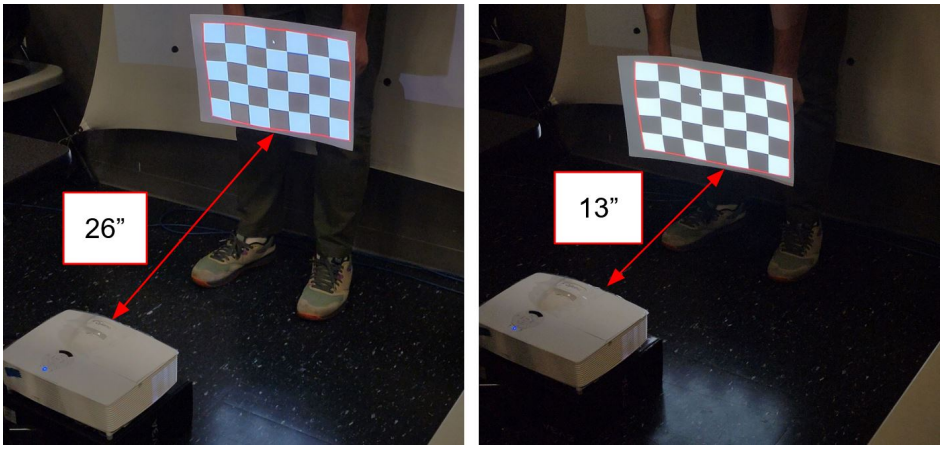
Figure 3.5: Effect of the surface speed on the projector and camera calibration accuracy for different types of projectors.

that becomes planar when static, we have to run this initialization only when the screen is moving.

3.3 Implementation and Results

Figure-5.2 shows the flowchart of our method. First, we calibrate the projector-camera pair. We project and detect ArUCo markers in the RGB-D capture to determine the 2D-3D correspondences for camera calibration. Next, we fit planes through the 3D points that correspond

(a) Marker-based surface



(b) DPM outside calibration volume

Figure 3.6: DPM on **(a)** a marker-based and **(b)** a markerless deformable surface using our calibration. Note how the projection remains aligned with the markerless surface in **(b)** despite moving out of the calibration volume.

to the same projector row, column and diagonal to get a coarse estimate of the projector parameters. Finally, we refine the projector parameters to get an accurate calibration. Using the calibration parameters, we perform D-SAR using the methods proposed in the previous chapters.

We implemented the proposed system in C++. We used OpenCV's APIs for detecting ArUCo markers with sub-pixel accuracy and Levenberg-Marquadt optimization to implement our calibration routines. For the camera, we used an Azure Kinect. We tested our method on three different kinds of projectors - regular throw (Optoma ML750), short throw (Optoma EH200ST) and ultra-short throw (Optoma GT5600). The Azure Kinect itself comprises of two cameras: a time-of-flight IR/depth camera and an RGB camera. Although the depth camera resolution is coarser than the RGB camera i.e. (640×576) vs. (3840×2160), its

API provides an interpolated point cloud rendered from the RGB camera view, similar to other consumer-grade depth cameras like the Intel RealSense Depth Cameras [42]. In our calibration routine, we used 3D points from this interpolated point cloud. The projector resolution was (1920×1080). We projected a grid of ArUCo markers for 20 frames before switching to another grid layout.

3.3.1 Projector-Camera Calibration Accuracy

We tested our proposed calibration method on a dynamic, deformable surface. The projector-camera system was placed around 864mm away from the screen. The width and height of the projection was 1270mm and 762mm respectively. This gives us a working volume of around 1270W x 760H x 865D cubic mm. We placed two fans on either side of the surface which would generate random waves and ripples across the surface and tested our calibration for various speeds of the fans (see Figure-5.1). This helped us to study the impact of the movement of the surface on the accuracy of our calibration technique and the quality of our display. In order to estimate the surface speed, we took the difference between two successive depth maps to measure the velocity and took the average value. We quantified calibration accuracy using reprojection error onto camera and projector images using the calibration parameters (Figure-3.5). Note that the camera reprojection error is less than 0.5 pixel and does not get impacted by the movement of the surface (see Figure-3.5b). A error of less then 0.5 pixel is considered good calibration as per best practices [76]. Even at the highest speed, our projector reprojection error is less than 4 pixels – less than 0.4% when considering the size of the projection image (see Figure-3.5a). The maximum 3D reconstruction error is 3.7mm which is less than 0.4% considering the size of the volume we are operating in. Considering the variation of the different parameters estimated by the calibration process (including projector and camera focal length, translation and rotation), we had an average standard deviation of less than 5%.

Finally, we used the calibration parameters generated by our method to perform DPM on a dynamic, deformable surface. Figure-5.3a-5.3b show the DPM display using calibration parameters generated by our method. Our calibration method is robust and able to generate calibration parameters that are valid even outside the calibration volume. Figure-5.3b shows a user moving a markerless display surface very close to the projector, well out of the calibration volume, which was approximately 34 inches away from the projector. Notice how the projection still remains well aligned within the surface using our calibration parameters.

As for the number of frames used for the calibration, we did not see any significant improvement to the calibration accuracy for more than 100 frames. In general, we capture around 15-25 frames for a successful calibration irrespective of the surface speed, which takes approximately 30 seconds to complete.

Chapter 4

Jitter Removal and Occlusion Handling

In the previous chapter, we presented the first work on accurate calibration of a projector-RGB-D camera pair using a dynamic and deformable surface.

However, even after proper calibration, D-SAR for dynamic, deformable surfaces must deal with other challenges, such as noisy depth cameras and surface occlusion. In this chapter, we discuss these challenges and explain methods to solve them.

Unlike rigid objects where the shape is known apriori, D-SAR on deformable surfaces rely on the depth camera to reconstruct the changing shape of the surface every frame. However, the depth data is usually noisy, and introduces tiny perturbations in the reconstructed shape, even if it is completely static. Since the projection adapts itself to this continuously changing noisy shape, it creates a jitter that can impede comprehension of projected content in addition to being simply annoying. Additionally, occlusion of the surface, for example, by a person walking in front of the projected display, substantially changes the reconstructed shape of the surface as the occluding object gets captured instead of the dynamic projec-

tion surface. This results in a significant distortion of the projected content even in the unoccluded regions, completely disrupting the viewer experience. Our DPM system is motivated by creation of visualizations on the inside walls of tents in remote austere locations during emergency management or military operations. In any such real-world deployment of DPM systems, the issues of jitter and occlusion should be addressed adequately to create comprehensible displays.

In this chapter, we explain methods to address jitter and occlusion when the D-SAR system is in action. For jitter, we employ Kalman filtering on the depth samples of the surface and smooth them over time to determine the surface shape. The Kalman filter is an efficient, recursive filter that uses a series of measurements over time to produce estimates and predictions of unknown variables (in our case, the true surface depth). This results in a comprehensible and perceptibly pleasing display, which we verify through two user studies. Leveraging the depth prediction by the Kalman filter, we devise an algorithm to handle occlusion of the surface by obstructing objects. Therefore, the methods in this chapter provide solutions to important practical challenges in deploying D-SAR systems in the real-world: correcting for jitter resulting from noisy depth cameras and handling surface occlusion.

4.1 Related Work

Jitter Correction: Jitter noise has always been a problem in video captures. Tiny, high frequency motion caused by unstable camera support equipment or the user's hand manifests as jitter which severely degradres the quality of videos. Therefore, most video stabilization techniques serve as a post-processing step to remove the jitter while preserving and smoothing the intentional motion of the camera. They do so by estimating the motion vectors in an image sequence, which can be achieved by pixel-based, block-matching or feature-based algorithms. The interested reader is referred to [24] for more information.

Jang et al. [44, 45] use the Kalman filter to remove jitter noise from a sequence of images captured at various focuses for shape recovery, or shape-from-focus. Kwon et al. [61] compute motion vectors in an image sequence using phase correlation and smooth them over time using the Kalman filter. Yaman et al. [133] use the constant velocity Kalman filter for smoothing motion vector trajectories for image stabilization. More recently, some works [138, 15] have used deep learning to stabilize videos.

For DPM, the jitter reduction algorithm must be real-time, and not rely on the projected content to compute motion vectors. Furthermore, it must be online i.e. smooth frames as they come in and not rely on future frames. Some methods are efficient enough to perform video stabilization in real-time [48, 51]. Jia et al. [48] perform online video stabilization by implementing a constrained constant velocity Kalman filter. We take a similar approach in our work.

Occlusion Handling in Projection Displays: Methods for occlusion handling for projection mapping displays differ based on whether the display surface is static-rigid, dynamic-rigid or dynamic-deformable. For static-rigid and dynamic-rigid surfaces, the shape of the surface is known apriori and remains unaffected by occlusion. Therefore, prior works on multi-projector displays typically focus on identifying occluders for shadow removal by increasing the contribution of one or more projectors to compensate for an occluded projector in an overlapping region. However, for dynamic-deformable surfaces, the system must extract the surface shape every frame which can get affected by an occluder.

In [47, 46], Jayens et al. render an expected projection display image for each camera in a calibrated, multi-projector display. They compare this expected image with the actual camera capture to identify shadows and remove them. Sukthankar et al. [115] also use a similar approach by comparing a reference image to the current camera capture for dynamic shadow compensation. However, these systems are not suitable for complex, deformable surfaces. Lange et al. [63] perform DPM using multiple projectors on a rigid object whose

3D shape is known apriori. In order to remove shadows on the object caused by occlusions, they track occluding objects in the projection volume using a depth camera and adjust the blend weights of other projectors to compensate for the blocked projector. The Pmomo system [139] detects occluding objects in a single projector DPM system for rigid objects to cull projection on the occluder for a more realistic experience.

Flexpad [113] performs DPM on a deformable, non-stretchable surface that a user can interact with. They use optical surface material analysis on the raw IR image from Kinect to identify skin pixels (e.g. the user's hand) that may be occluding the surface. Narita et al. [78] perform DPM on a non-rigid object by printing a precise deformable dot cluster marker (DDCM) pattern on a surface using IR ink. This pattern consists of clusters of dots, numbering from 1-4, arranged in a grid across the entire surface. At each frame, the dot clusters are extracted and identified. In a separate thread, they track each dot cluster across frames. Since they use a high-speed camera ($\sim$1000fps), the displacement of a dot cluster between frames is small. Therefore, they detect the dot cluster in a window around its previously detected location. If a dot cluster is not detected in that region, they use linear interpolation and extrapolation of tracked neighboring clusters to estimate its position.

4.1.1 Comparison to Prior Work

Prior D-SAR systems do not focus on delivering a high quality display where the projection "sticks" to the moving surface making detailed high resolution content (e.g text) to be readable and comprehensible. The goal is to create the perception of the content to be printed on to the moving and deformable projection surface. The proposed jitter and occlusion handling methods take us quite close to this difficult goal resulting in highly comprehensible displays even in the presence of fast and large movements and deformations.

Driven by the application of creating high quality visualizations on inside walls of tents in the presence of environmentals (e.g. wind), we have provided several quantitative evaluations as well as initial user studies to show that the display created by the proposed DPM system indeed is a practical viable solution.

4.2 Method

4.2.1 Jitter Reduction

Following calibration, we use the recovered system parameters to adapt the projection to the dynamic, deformable surface using the method in Chapter-2. At each frame, we find the 3D location of the equi-distant black markers marking the desired rectangular area of projection and fit a B-spline to the detected 3D region. Using the recovered projector parameters, we then warp the projected image by projecting the B-spline surface into projector image plane. However, noise in the depth camera results in small perturbations in the fitted B-spline surface and the warped projection has jitters i.e. a small, high frequency motion. This jitter is perceptually annoying and, more importantly, can render the display unreadable or incomprehensible. Therefore, we employ Kalman filtering on the 3D B-spline surface to account for the depth camera noise and reduce jitter.

The Kalman filter uses a series of measurements observed over time and produces estimates and predictions of the system state by taking into account the uncertainties of the system and the measurements. This results in a more accurate estimate of the system state than by considering a single measurement only. It consists of a predict-correct loop. Once initialized, it predicts the system state, denoted by $x \in \mathbf{R}^{N_x \times 1}$ and uncertainties in the prediction at the next timestep. When a measurement, denoted by $z \in \mathbf{R}^{N_z \times 1}$ is received, it updates (or corrects) the prediction of the current state to produce a new estimate. Mathematically, the

predict step can be written as

$$\hat{x}_{t+1,t} = \mathbf{F}\hat{x}_{t,t} \tag{4.1}$$

$$\mathbf{P}_{t+1,t} = \mathbf{F}\ \mathbf{P}_{t,t}\mathbf{F}^T + \mathbf{Q} \tag{4.2}$$

where $\hat{x}_{t+1,t}$ is the predicted state vector at time $t+1$, $\hat{x}_{t,t}$ is the estimated state at time t. $\mathbf{F} \in \mathbf{R}^{N_x \times N_x}$ is the state transition matrix that extrapolates the current state to the next timestep. $\mathbf{P} \in \mathbf{R}^{N_x \times N_x}$ is a covariance matrix and represents the estimate uncertainty. A higher estimate of uncertainty will cause the Kalman filter to favor the measurement more than the current state in the estimatiion. $\mathbf{Q} \in \mathbf{R}^{N_x \times N_x}$ is the process noise uncertainty.

In the update step, the Kalman filter computes the Kalman gain which weights the prediction and the current measurement to compute an estimate for the current system state. Mathematically,

$$\mathbf{K}_t = \mathbf{P}_{t,t-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{t,t-1}\mathbf{H}^T + \mathbf{R}_t)^{-1} \tag{4.3}$$

$$x_{t,t} = x_{t,t-1} + \mathbf{K}_t(z_t - \mathbf{H}x_{t,t-1}) \tag{4.4}$$

$$\mathbf{P}_{t,t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_{t,t-1}(\mathbf{I} - \mathbf{K}_t\mathbf{H})^{-1} + \mathbf{K}_t\mathbf{R}_t\mathbf{K}_t^T \tag{4.5}$$

where $\mathbf{K} \in \mathbf{R}^{\mathbf{N_x} \times \mathbf{N_x}}$ is the Kalman gain, $\mathbf{H} \in \mathbf{R}^{N_z \times N_x}$ is the observation matrix that maps the state vector to measurement vector space and $\mathbf{R} \in \mathbf{R}^{N_z \times N_x}$ is the measurement uncertainty while $\mathbf{I}$ is the identity matrix. The subscript t denotes the timestep.

In our case, the measurements z are the 3D points on the surface from the depth camera that is used to compute the B-spline surface. Our goal is to compute a more accurate estimate of these 3D points. Therefore, our system state vector x consist of all the 3D points used to compute the B-spline surface. We have implemented three different Kalman filter models based on the assumptions made by the prediction step: *the constant velocity,*

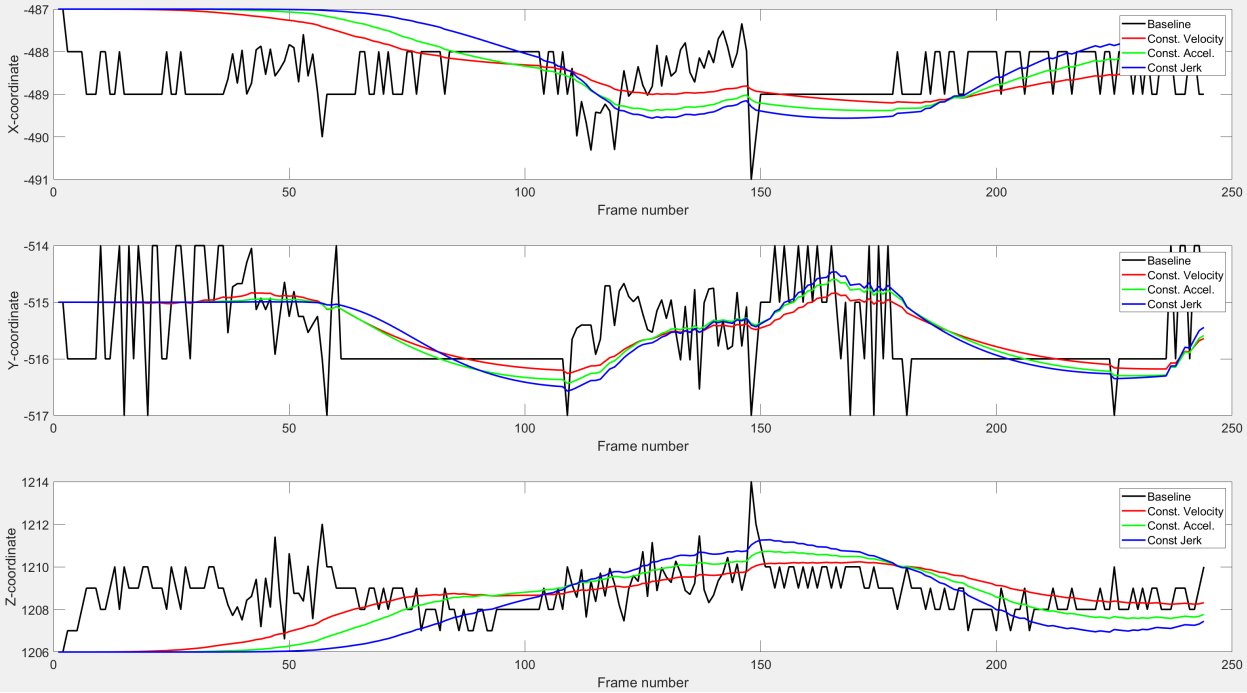
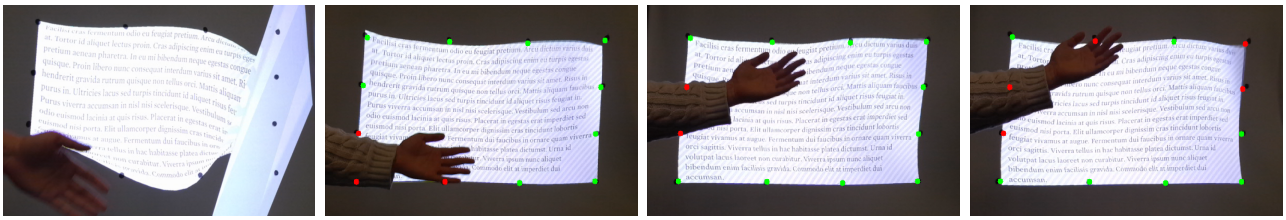


Figure 4.1: Kalman filter smoothing for a stationary 3D point over 250 frames. Note the amount of noise in the data from the depth caemra (black). While all three models smooth it out, the constant velocity filtering (red) varies the least compared to the other two.

constant acceleration and *constant jerk*. Depending on the model being used, the system state also comprises the velocity, acceleration or jerk of each 3D point. The *constant velocity* model assumes that the predicted location only relies on the velocity of the 3D point that does not change between successive frames. The *constant acceleration model* accounts for acceleration and velocity to predict the 3D location assuming that the predicted acceleration remains constant. The *constant jerk* model adds the jerk to the prediction, assuming that the predicted jerk remains constant. These three models therefore have different system states and state transition matrices. The constant velocity model has only the location and velocity of the 3D point in the state vector, while the constant acceleration and constant jerk models additionally include acceleration and jerk respectively. Irrespective of the model used, we only use the estimated location from the state vector at each timestep. The state transition matrix implements the equations of motion. At each frame, we acquire measurements of the surface from the depth camera. These measurements are fed into the Kalman filter which



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

Figure 4.2: Occlusion of a moving surface by a hand moving across it. **(a)** shows the distorted projection without any occlusion handling. **(b)-(d)** Green points are detected points while red points are estimated points due to occlusion. Notice that sometimes, even unoccluded points are not detected by the camera (points on the right edge in **(d)**) but our algorithm is able to determine their position accurately.

produces a more accurate estimate of each sampled 3D point on the surface which are then used to compute the B-spline and warp the content for projection.

4.2.2 Occlusion Handling

We leverage the Kalman filter prediction of the 3D points on the B-spline surface during jitter reduction to handle surface occlusion. We first use the method in Chapter-2 to detect the 3D location of markers at the boundary of the projection region and all the 3D points within the desired boundary and compute the B-spline surface. In occlusion handling step we identify the region of the surface that is occluded by detecting the occluded points from the 3D points used for the B-spline surface. These occluded points are then replaced by the Kalman filter prediction. Assuming that an occluding object will always have depth that is less than that of the projection surface, we compute the minimum depth that was sampled in B-spline region in the previous frame. In the current frame, we mask out any regions whose depth is less than the minimum depth (from the previous frame) by a certain threshold.

Detecting Occluded Markers: In order to determine which sampled points are occluded, we first start with the surface boundary. Our goal is to match each boundary point in the current frame with the detected boundary points in the previous frame. We do this by

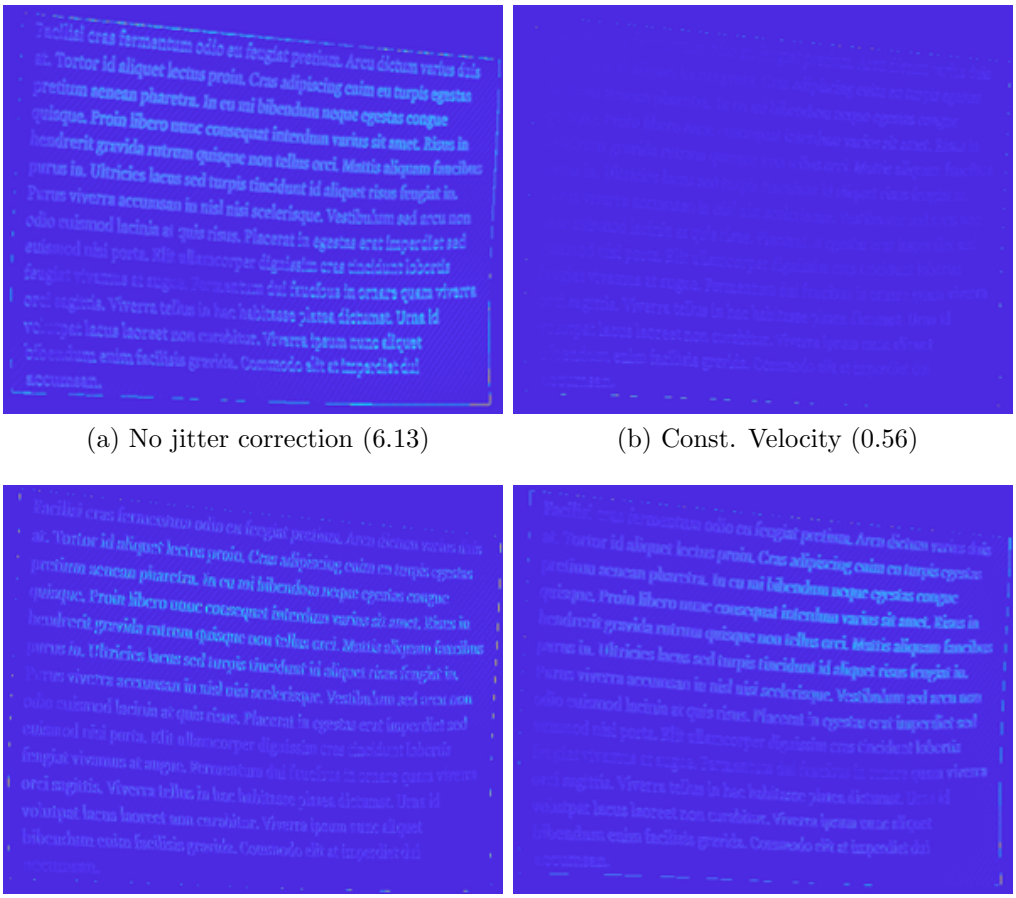
matching each boundary point in the previous frame to the closest 3D point in the current frame. If a boundary marker in previous frame is not matched, it is labeled as occluded and replaced by its Kalman filter prediction from the previous frame.

Accounting for global movement: Detecting occluding markers alone cannot take into account changes to the global position of the entire surface in the current frame. For example, the surface may move forward in unison. This global movement may not be correctly predicted by the Kalman filter and can cause the display to become distorted. Therefore, we additionally compute a rigid transform that transforms 3D points in the previous frame to the points in the current frame in a least-squares sense. For this, we use all the points – both unoccluded and also the occluded ones that are predicted by the Kalman filter. We then apply this rigid transform to points in the previous frame corresponding to occluded points in the current frame to get an estimate of the occluded 3D points in the current frame. Though not very accurate, computing a rigid transform is fast and efficient, does not result in extrapolation artifacts that nonlinear transforms may, and does not compromise the real-time performance of the display.

4.3 Results

Jitter Removal

The results with jitter correction are visibly less jittery compared to the display generated using the raw depth measurements. Figure-4.1 shows how the three filter types smooth out a jittery trajectory of a stationary 3D point. Figure-4.3 shows the average absolute difference maps over successive frames for each model when projecting static content on a stationary surface. Additionally, we conducted a user study to examine the impact of the



(a) No jitter correction (6.13) (b) Const. Velocity (0.56)

(c) Const. Acceleration (2.77) (d) Const. Jerk (3.83)

Figure 4.3: Maps of average absolute difference across a number of successive frames when projecting a static text image on a stationary projection surface. The average absolute differences are **(a)** 6.13, **(b)** 0.56, **(c)** 2.77 and **(d)** 3.83. Note how the difference is very low for constant velocity model shown in **(b)**.

three jitter reduction models on the projection display quality and compared them to the display rendered without jitter reduction. The results are discussed in Section-4.3.1.

Figure-4.2 shows the occlusion of a rapidly moving surface by a hand. Notice how the projection gets severely distorted without any occlusion handling. However, our occlusion handling algorithm estimates the locations of the occluded points reasonably well to keep the projection coherent. Also notice that some points may not get detected by the camera even though they are unoccluded (see Figure-4.2d). These points are treated as occluded points and our algorithm estimates their positions.

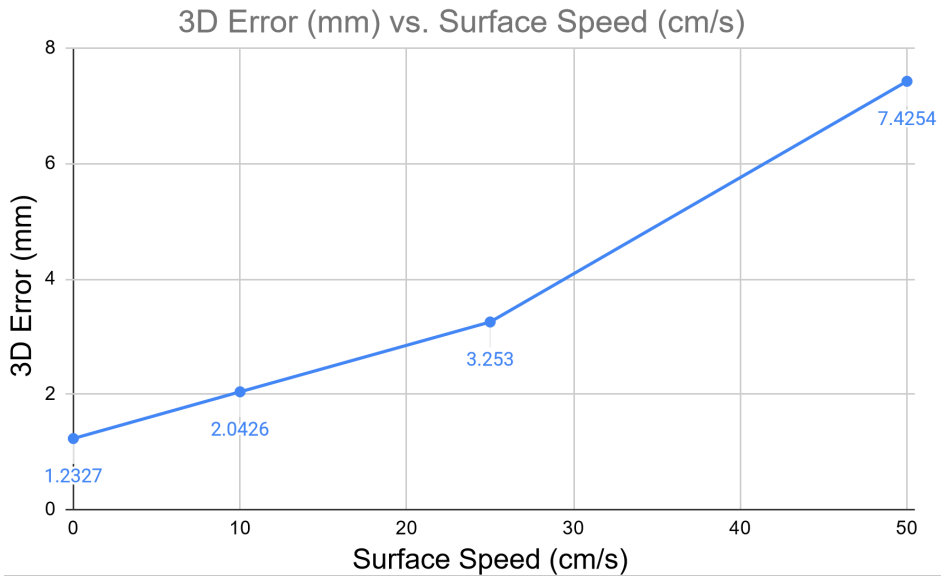


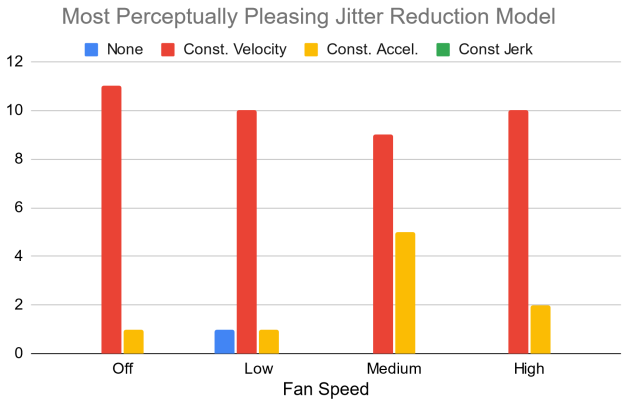
Figure 4.4: Average error between 3D points predicted using our occlusion handling method with their ground truth locations at various surface speeds.

Occlusion Handling

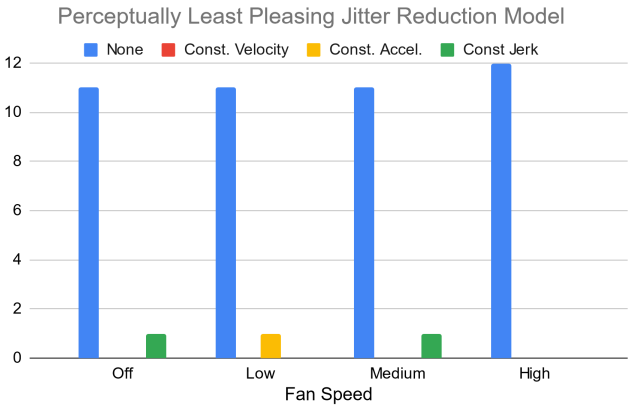
To evaluate the accuracy of our occlusion handling method, we programmatically occluded some surface markers that were detected by our algorithm, estimated their locations using our occlusion handling method and then computed the distance from their true locations. We computed the average distance over multiple frames for several points. Figure-4.4 shows the error between the true locations and the estimated locations using our method for different surface speeds. Note that the error for a static surface is 1.2mm, which is not perceptible to viewers. With higher surface speeds, the error increases, reaching a maximum of 7.5mm when the surface movement resembles a strong gust of wind. Even at such high speeds, the mean error is less than a centimeter and viewers do not perceive it in the final display.

4.3.1 User Study

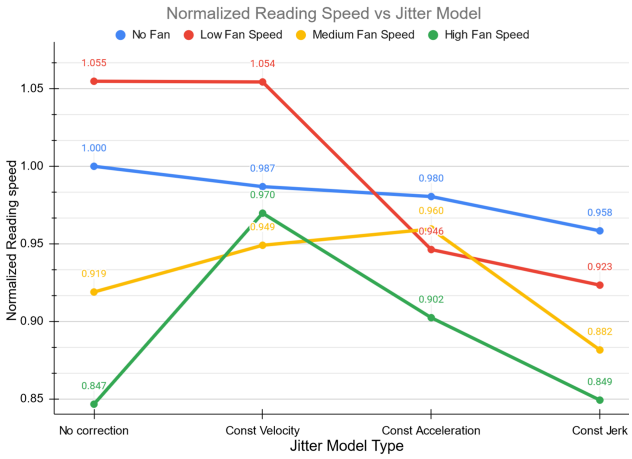
In order to study the effect of the various jitter reduction models, we conducted a qualitative and quantitative user study. Both user studies asked users to evaluate the display for 16



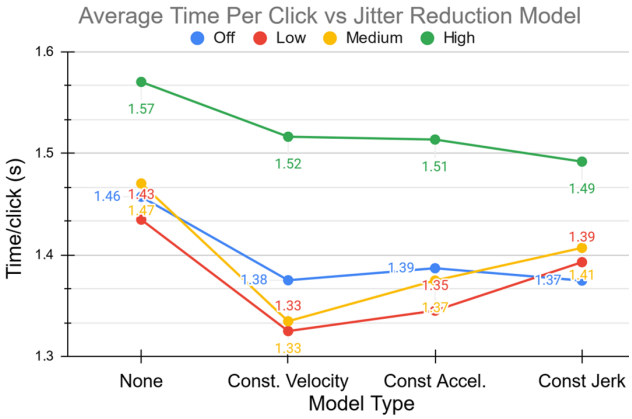
(a) Votes for perceptually most pleasing display



(b) Votes for perceptually least pleasing display



(c) Effect of fan speed and jitter on reading speed



(d) Effect of fan speed and jitter on clicking time

Figure 4.5: Results of our user study. **(a)-(b)** show the votes for the jitter model with the perceptually most pleasing least pleasing display respectively. **(c)-(d)** show the effect of the surface speed and jitter correction model on the reading speed and average clicking time.

cases in total i.e. four fan speeds (off, low, medium, and high) for four jitter models (none, constant velocity, constant acceleration and constant jerk). For the qualitative study, there was one image for each fan speed, while for the quatitative study, there was one image for each case i.e. 16. All images used were different. Viewers were seated approximately 7 feet away, looking directly at a $3.5' \times 2'$ projection area. In total, there were 12 participants (7 males, 5 females), ranging from 20-55 years old.

For the qualitatitve study, users were shown dynamic projections of static text images projected on the surface and were asked to rate the perceptually best and worst display based

on the jitter. They could dynamically switch between the projection display rendered using the four jitter correction models by using four buttons on a keyboard. They were not told which model they are evaluating to remove bias – the models were named Model A, B, C and D. Before voting for each case, the keys were randomized so that users could not be aware which key corresponds to which model type used for rendering. They were instructed to switch between the four models and rate the perceptually best and the perceptually worst display. The experiments were conducted for four different fan speeds.

For the quantitative study, we measured users' reading speed and their tracking speed when using the display at different fan speeds with various jitter models. For measuring the reading speeds, users were timed while reading a short piece of text, approximately 35-40 words long, that was projected on the dynamic display. They were timed while reading different texts, of similar difficulty level, for each of the 16 cases. This removed bias that can be created by repeated reading of the same text. We computed the reading speeds for each user for each case. In order to correct for variations in reading speeds across users, we normalized the reading speeds for each user by their baseline reading speed i.e. when the display was static and without any jitter correction. Then, the normalized reading speeds were averaged across users.

To measure their tracking speeds, users were shown a (2×3) grid of small black squares on a white image at the start of the experiment. Upon starting, one of the 6 black squares would turn red and users were instructed to use the mouse pointer to click it. Every time the user clicked a red square correctly, that square would turn black and another square would randomly turn red. At the same time, the mouse position was reset to the center of the display to ensure it travels the same distance in each experiment. This would continue until all 6 squares had been clicked. We recorded the mean time each user took to click a red square for all 16 cases.

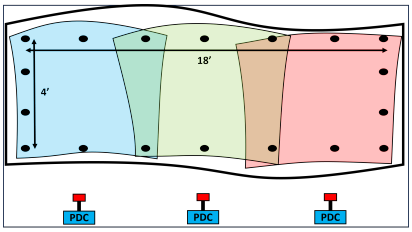
Figure-4.5 shows the results of our user study. Users overwhelmingly found the constant velocity model to be the perceptually most pleasing across all speeds (Figure-4.5a), while no jitter correction was considered the worst (Figure-4.5b). However, we noticed that at higher fan speeds, users were having a harder time deciding between their preference for constant velocity and constant acceleration models, sometimes ranking them equally. This is because the motion of the rapid surface started masking the jitter more. Figure-4.5c shows the result of the quantitative study comparing normalized reading speeds. We noticed that jitter correction did not make a significant impact on the reading speed at lower fan speeds. This is because the text was still relatively localized. However, at higher fan speeds, having some form of jitter correction significantly improved reading speeds. Figure-4.5d shows the effect of the fan speed and jitter model on the clicking time. Users consistently performed better at tracking for the constant velocity model across all fan speeds compared to without jitter correction. Overall, users indicated that any form of jitter correction made reading and tracking easier, irrespective of fan speeds. Combining the results of the user study, we recommend the constant velocity jitter correction model.

Chapter 5

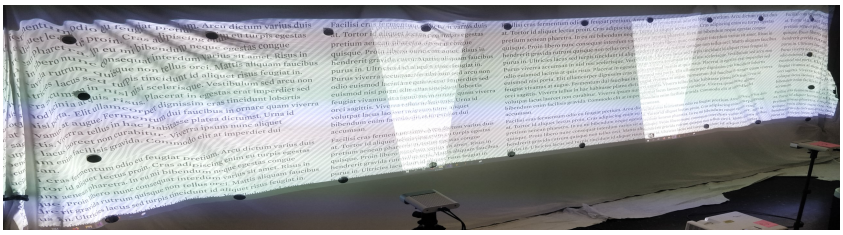
Multi-PDC Deformable-SAR

The previous chapters of this thesis focus only on D-SAR using a single PDC unit. In this chapter, we explain in detail the first end-to-end solution for achieving a real-time, seamless display on deformable surfaces using multiple unsynchronized projectors without requiring any prior knowledge of the surface or device parameters. The system first accurately calibrates multiple RGB-D cameras and projectors using the deformable display surface itself, and then using those calibrated devices, tracks the continuous changes in the surface shape. Based on the deformation and projector calibration, the system warps and blends the image content in real-time to create a seamless display on a surface that continuously changes shape. Using multiple projectors and RGB-D cameras, we provide the much desired aspect of scale to the displays on deformable surfaces.

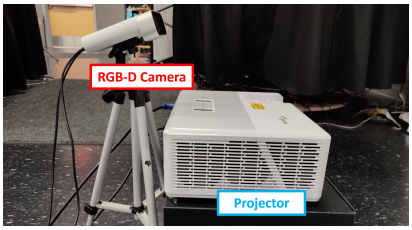
Most prior dynamic multi-projector systems assume rigid objects and depend critically on the constancy of surface normals and non-existence of local shape deformations. These assumptions break in deformable surfaces making prior techniques inapplicable. Point-based correspondences become inadequate for calibration, exacerbated with no synchronization between the projectors.A few works address non-rigid objects with several restrictions like tar-



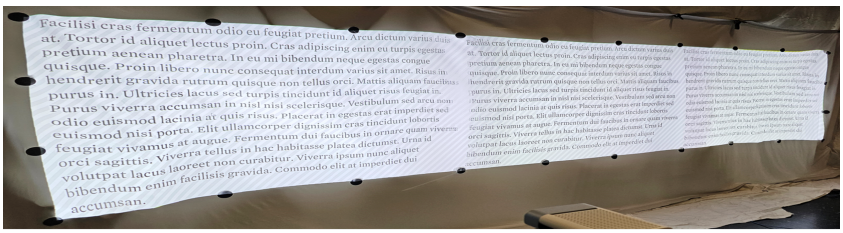
(a) Display Setup



(b) Uncalibrated display



(c) PDC unit



(d) Calibrated display

Figure 5.1: **(a)** Our setup, comprising 3 PDCs with overlapping projection covering a 18' × 4' (5.5m×1.2m) deformable surface. **(b)** The uncalibrated display without any warp and blend. **(c)** A single PDC unit. **(d)** The final seamless display.

geting semi-deformable surfaces (e.g. human face), or using single coaxial (optically aligned) projector-camera pairs, or temporally synchronized cameras.

In this chapter, we show how we break loose from such restrictions and handle multiple projector systems for dynamic deformable fabric-like objects using temporally unsynchronized devices. We devise novel methods using ray and plane-based constraints imposed by the pinhole camera model to address these issues and design new blending methods dependent on 3D distances suitable for deformable surfaces. Finally, unlike all prior work with rigid dynamic surfaces that use a single RGB-D camera, we devise a method that involve all RGB-D cameras for tracking since the surface is not seen completely by a single camera. These methods enable a seamless display at scale in the presence of continuous movements and deformations.

5.1 Background

Multi-projector displays are an easy way to create seamless displays at scale without undertaking massive engineering feats of tiling LCD/LED panels together. Though multi-projector displays on rigid objects (static or dynamic) have been explored extensively, expeditionary or mobile systems often demand such displays on surfaces that continuously move and deform. Examples include emergency management or military tents in austere locations where large data demand large visualizations for command and control, decision making, mission rehearsal and even training during long idle periods. The easiest way to setup such displays is to use projectors on the tent walls. However, such fabric based surfaces suffer continuous movement and deformations from environmentals like wind and vibrations. Similar situation is faced in commercial domain when mobile and inflatable displays need to be setup on tradeshow floors in the presence of ventilation systems blowing air or vibrations nearby from other demonstrations. Mobile educational systems (e.g. mobile planetariums) that travel to serve geographically under-served population can also benefit from displays on moving deforming surfaces.

Except for [111, 1], all *multi-projector* systems explored so far focus on rigid objects, either static [118] or dynamic [110, 80]. Any rigid object offers constant surface normals and no local deformation of the surface. Assuming prior knowledge of the 3D surface along with these invariants, [109] performed precise device calibration on a static rigid object for a dynamic projection mapping setup. Note that the precision required is rather exacting – if the physical set up and its estimation do not match, misregistrations result. To achieve the precision in the presence of movement, the rigid object is usually tracked by a single RGB-D camera. Note again, high accuracy tracking is essential to create a seamless display in which the projected image "sticks" to the rigid object as it moves. Finally, the blending between two overlapping projectors on the rigid object is achieved in real-time leveraging the fact that surface normals do not change with movement. [111] uses multiple projectors to project on

a human face by fitting a parametric face model to the depth camera output. The changing surface normals of the tracked face are used to select the projector rays with the highest illumination quality only. This effectively narrows the blend width in the overlapping region to reduce the visibility of misregistration artifacts.

Ahmed et al. [1] is the only multi-projector system that handles deformable surfaces like fabrics. They use two projectors and two temporally synchronized IR cameras, each pro-cam pair being individually coaxial to achieve a super-imposed display. They calibrate each coaxial pair separately, establishing pixel-to-pixel correspondences between the camera and its corresponding coaxial projector. During projection, each IR camera separately tracks fiducials on an IR grid painted on the deformable surface and using calibration parameters of its corresponding projector, warps the display content for projection, resulting in a superimposed display that does not require realtime blending.

However, when the projection surface is deformable (like a moving fabric), the devices are not temporally synchronized or optically aligned, all the previously mentioned conditions that are leveraged to deliver a seamless display break. The surface normals change continuously and local deformations of the surface are significant. This renders most of the techniques involved (e.g. calibration, blending) inapplicable. In order to scale, single RGB-D camera based tracking is inadequate since the entire deformable surface is not seen by a single RGB-D camera.

5.1.1 Main Contributions

In this chapter, we present novel calibration, tracking and blending methods that enable seamless multi-projector displays at scale on deformable dynamic surfaces. Following are the main contributions in this chapter.

1. *Calibration:* In the absence of point based correspondences in a moving deformable surface, we use ray or plane based constraints imposed by the pinhole camera model on the moving surface to achieve highly precise device calibration.

2. *Multi-camera Surface Tracking:* Our method takes partial information from each RGB-D camera to generate highly accurate tracking of every location of the deformable projection surface.

3. *Multi-projector Blending:* In the absence of constant surface normals, we present a new distance based blending method that can work with deformable dynamic surface.

4. *Complete System Pipeline:* Using the above information we warp and blend the content from multiple projectors that adapts in real time to the changing deformations and movements in the projection surface creating a stable seamless display. To the best of our knowledge, this is the first end-to-end pipeline that allows multiple, temporally unsynchronized projectors and cameras to adapt to the changes in a moving deformable projection surface to create a registered, stable and seamless display.

5.2 Related Work

Although there has been a significant amount of work in single projector systems, the primary focus of this section is on multi-projector systems that are most relevant to us. We consider the prior work in three categories here, *Rigid Static Surfaces*, *Rigid Dynamic Surfaces* and *Deformable (Dynamic) Surfaces*.

5.2.1 Rigid Static Surfaces

When multi-projector displays are used on planar surfaces, the device calibration is avoided by using homography based methods for registration [14, 7, 6, 96]. [99, 101, 103, 100] shows that registration can be achieved without precise device calibration if a prior on the category of shape (e.g. vertically extruded, swept or spherical surfaces) and measurements (e.g. aspect ratio of a cylindrical surface, radius of a spherical surface) of the display surface is known.

When an arbitrary geometry is used, some works use projectors and cameras separately. [92, 91, 90, 123] use a single camera to calibrate different devices separately and combine the results in a tree-like fashion. However, errors across device pairs make it nearly impossible to achieve sub-pixel accuracy in registration. Others calibrate a pro-cam unit (i.e. a unit made of a projector and a RGB camera) together and use these pro-cam units as building blocks for creating a display that has the same number of projectors and cameras. Single pro-cam systems typically employ structured light scanning of a known calibration object to establish pixel correspondences between the projector and camera followed by calibration [76, 3, 29, 30, 135, 106, 52, 17, 95, 134]. The calibration object can be a checkerboard pattern [17, 29, 30, 52, 76] or fiducials (e.g. QR codes) printed on a planar board [3, 135, 106], or even objects of arbitrary geometry to perform calibration, the latter of which allows projection mapping on non-planar surfaces. Such methods capture the static calibration object in several different poses, model the projector like an inverse camera [105] and use the camera calibration method by Zhang [137] to achieve full calibration. [131, 118] are the only works that achieve automated calibration (without the use of specific 2D or 3D props) of multiple projectors and RGB cameras projecting on a complex rigid 3D shape.

In terms of blending, systems with rigid static surface need to compute a blend mask only once. As noted by [108], the simplest approach to intensity blending in multi-projector displays is to illuminate each surface point from only one projector. However, this approach

can make misregistrations due to calibration errors as well as color and intensity differences between projectors clearly visible. [90, 93, 136] implement blending across multiple projectors by feathering the blend mask in the overlapping regions. However, the blend weights are computed in 2D projector-image space and does not consider the 3D geometry of the surface. Therefore, these methods are restricted to surfaces with simpler geometry (e.g. planes). There exist more advanced methods that approach blending as a part of the overall color non-uniformity of the multi-projector display. They compute blend maps that make color variations imperceptible while maximizing display quality, for planar [68] or arbitrary surface geometry [119].

5.2.2 Rigid Dynamic Surfaces

More recently, with the advent of RGB-D cameras, a few works explore the advantage of having an additional depth camera, albeit noisy and low-latency, in handling dynamic rigid objects. Use of PDC units (i.e. a unit made of a Projector, a Depth camera and a RGB Camera) became common in such systems. A large body of work focus on a single PDC unit and completely avoid calibration or shape recovery using a coaxial setup with highly specialized high-speed RGB-D devices [71, 53, 128, 114] that can additionally alleviate both the noise and latency issues. [139, 2, 94, 27, 95] calibrate a single PDC unit with prior knowledge of a precise 3D mesh of an object that serves as the calibration object. [110, 63, 59, 109, 62, 80] use multiple projectors to illuminate a moving rigid object. [110, 63, 59, 109, 62] use a single depth camera to tie all the projectors together via an object with known 3D shape. [80] uses several motion capture cameras to track markers on the target object. These works focus on illuminating small 3D objects (e.g. a bust) that can be contained within the field of view of a single RGB-D camera.

Kurth et al. [60] then advance to systems with multiple RGB-D cameras to allow 360 degree surround illumination of rigid moving objects. Multiple RGB-D cameras are required for covering the surround field of view created by all the projectors. This work proposes a calibration method that supports multiple RGB-D cameras and projectors, however still uses a single RGB-D camera for tracking. Since the rigid object does not change shape, as it moves the part visible to the RGB-D camera can be matched to the known rigid geometry via iterative closest points (ICP) based techniques to find the pose and orientation.

In a dynamic system, the overlapping regions of the projected imagery from multiple projectors change continuously due to the movement of the projection surface. Therefore, blend masks for each projector need to be recomputed in real-time. [66] proposes a method to compute blend masks for dynamic rigid objects using the dot product between the surface normal at a point on the object and the unit-rays from each projector. However, they do not take the 3D distance into account. [80] introduces a blending method that considers the surface geometry, distance to the surface as well as the pixel size on the surface to compute blend weights across projectors. By constructing a light transport matrix, [110, 111] solve a global optimization problem using a GPU-based solver to achieve real-time *content dependent* projector blending on a dynamic, rigid object. However, this solution does not scale as the light transport matrix becomes too large with more projectors. [60] proposes a high-performance distributed solver to achieve real-time performance in small systems. Although blending techniques that respect surface discontinuities have been proposed [4], they are not real-time and used for a different purpose of illuminating an animatronic head with limited local motions to render different expressions and manipulating the projection images on it to enhance its expressiveness.

5.2.3 Deformable Dynamic Surfaces

The rigid 3D object of known shape provides the critical anchor for all the devices in a system that illuminates rigid dynamic objects. The biggest challenge when moving to deformable objects is the constantly changing shape of the object that removes any way to anchor the system. The changing shape of the surface has to be tracked continuously and the projection has to adapt to these changes in real-time. This challenge becomes multi-fold when considering geometrically stitching images projected from multiple PDCs to achieve a seamless display, especially when the PDCs are not synchronized. Therefore, most prior work skirts around the problem of device calibration and surface recovery by using coaxial pro-cam units [87, 54, 10, 8]. Some works embed retroreflective markers [18, 73] or markers painted with IR ink [84, 86] on the deformable surface to track it and recover its shape. [77, 78] annotate the display surface with a precisely printed dot cluster marker grid using IR ink.

Some highly specialized systems like the MIDAS system [72] use a specialized, high-speed projector, three high speed cameras and three near-IR light sources of different bands, all setup in a co-axial arrangement using mirrors so that they have the same center of projection. Each of them is equipped with color filters to sense each of the three NIR light sources. This setup enables them to use photometric stereo to compute the object normals at every camera pixel. With this information, they render their projection image to change the appearance of the object to a different material. Due to the use of coaxial high-speed setup, this system can achieve dynamic projection mapping without much perceptible lag.

Ahmed et al. [1] implement a two-projector superimposed display on a deformable surface. Each projector is coaxially aligned with an IR camera. Although they calibrate each coaxial pair separately, they do not calibrate all the devices together and hence, do not reconstruct

the 3D surface geometry. Instead, each camera tracks an IR grid painted on the surface and the display content is warped separately for each projector.

The method proposed in Chapter-3 is the only method that calibrates a consumer-grade *non-coaxial* single PDC unit on a dynamic, deformable surface without using any embedded markers. The projector projects a grid of ArUCo markers to establish projector-camera pixel correspondences on a moving surface before calibrating the PDC unit. However, that method cannot be used to calibrate multiple PDCs.

5.2.4 Comparison

The work in this chapter is closest to [1], but is significantly different from their work in the following respects. **(i)** We calibrate all our devices and align them in world space instead of using coaxial pro-cam pairs that are not calibrated with respect to each other. This allows us to **(ii)** place our cameras and projectors so each device need not view/illuminate the entire surface, enabling large-scale displays unlike [1]. Additionally, **(iii)** unlike [1], we reconstruct the entire 3D surface geometry, **(iv)** without requiring temporal synchronization of the cameras. Finally, **(v)** our display requires real-time blending based on the reconstructed surface shape, unlike [1] who implement a two-projector superimposed setup and hence, do not require it.

Multi-PDC systems have not been explored in the context of deformable dynamic objects. Therefore, blending techniques are non-existent for this scenario. The normal-based blending applied for rigid dynamic system are not applicable since surface normals change continuously in a deformable dynamic object. Instead, we propose a novel blending technique dependent on distances between the devices, their view frusta and the changing projection surface. Finally, since no single RGB-D camera sees the entire projection surface, we devise a new tracking method that combines the information from multiple unsynchronized RGB-D

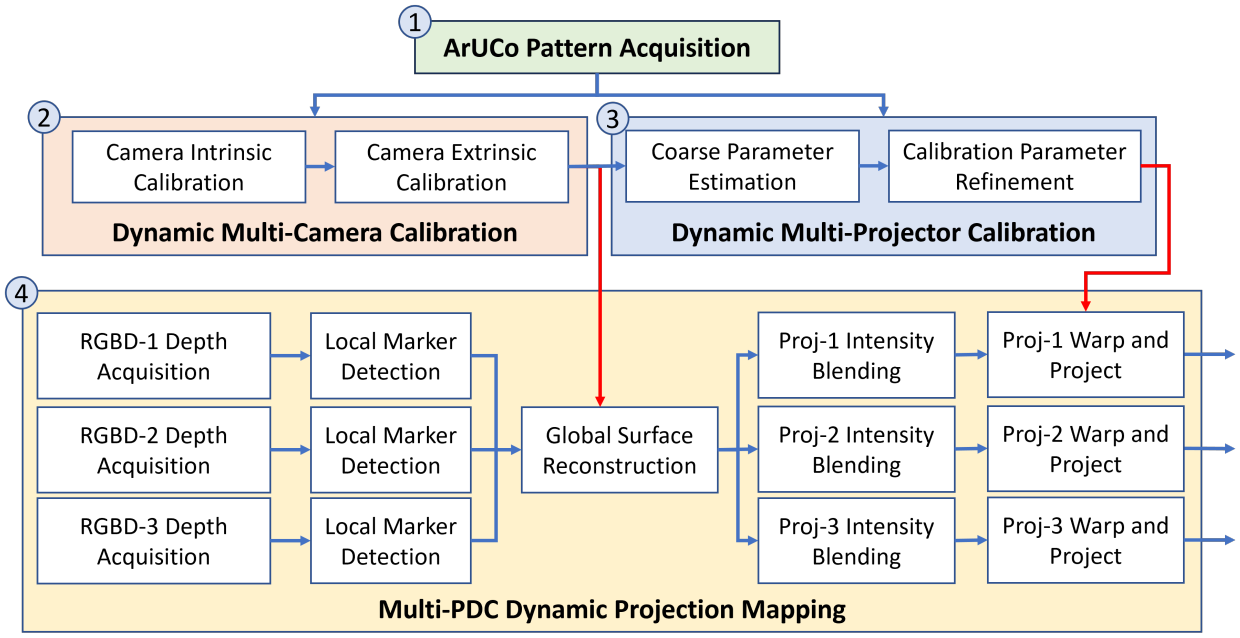


Figure 5.2: The flowchart of the proposed system.

cameras to provide acceptable tracking updates. However, for calibration, our method starts from the solutions proposed for single PDC units in Chapters 2-4 and design more comprehensive methods based on ray or plane-based constraints imposed by the pin-hole camera model to achieve accurate device calibration for multiple PDC units.

5.3 Proposed System

The system setup consists of N PDCs projecting on a large, continuous and smooth deformable surface in a tiled manner such that the projections overlap (see Figure 5.1 (a)-(b)). The boundary of the display region is marked on the surface by black dots placed equidistant along each edge, resulting in a final display. Each projector has a corresponding RGB-D camera that observes part of the surface, such that the camera fields of view also overlap. We assume that the color camera and the depth camera in each RGB-D camera are temporally synchronized and geometrically registered, which is common for almost all consumer RGB-D

cameras. Further, we do not require any knowledge of the projector or camera intrinsics or extrinsics.

Our method begins by each projector projecting ArUCo patterns on the moving deformable surface while all RGB-D cameras capture those patterns and decode them to generate pixel correspondences. Then, we proceed to camera calibration, followed by projector calibration. Using the calibration parameters, we track the surface shape and perform intensity blending in real-time to adapt to the changing surface shape and create a stable seamless multi-projector display on a dynamic, deformable surface.

The goal of a single-projector projection mapping system is to determine a texture-mapping function $\Omega(\cdot)$ that warps the source image I_{src} to a target image I_{tgt} such that I_{tgt} conforms to the surface shape when it gets projected out. The texture mapping function is defined as: $\Omega(p) = s$, which maps a projector pixel $p \in \mathbf{R}^2$ in I_{tgt} to a texture coordinate $s \in \mathbf{R}^2$ in I_{src}, i.e. $I_{tgt}(p) = I_{src}(\Omega(p))$. In our multi-projector system, each projector j has its own texture mapping function, denoted by $\Omega^j(p) = s$ which must be computed based on the current surface shape. Therefore, in order to compute the correct $\Omega^j(p)$ and render a *registered* multi-projector display, we need the following information: the texture coordinate $s = (u, v)$, the 3D point d on the surface that it maps to and the projector pixel p^j in projector j that will illuminate the surface at that point. Since there are multiple cameras, each observing a part of the surface, we require accurate camera calibration in order to merge their depth in a unified world space. In order to map the 3D point d to its correct projector pixel p^j in projector j, we require accurate projector calibration.

Finally, to render a *seamless* display, we must blend the projectors in their overlapping regions. This is achieved by computing a per-projector intensity mask, denoted by $\alpha_j(p), 0 \leq \alpha^j(\cdot) \leq 1$. Therefore, for the i-th point on the surface illuminated by projector j, we require the following tuple of correspondences: $f_i^j = (s_i, d_i, p_i^j, \alpha_i^j)$, where $s_i = \Omega(p_j^i)$, and d_i is the 3D point on which pixel p_i^j gets projected on. The final image that is rendered by

projector j in a multi-projector setup is $I^j_{tgt}(p) = \alpha^j(p) I_{src}(\Omega^j(p))$. Note that for systems handling deformable dynamic surfaces that continuously change in shape, $\alpha^j(\cdot)$ and $\Omega^j(\cdot)$ are additionally functions of time. However, we omit that for the sake of brevity.

Figure 5.2 gives an overview of the complete pipeline of our system. First, we perform calibration, which recovers accurate projector and camera calibration parameters. Next, we use the cameras to capture and track the surface shape. With the camera calibration parameters, we align the surface depth from each camera to the world space and reconstruct the surface. Using the projector parameters, we compute the blend weights $\alpha^j(\cdot)$ for each sampled point on the surface and reproject it to the projector to determine its corresponding projector pixel. This enables us to compute the texture mapping functions $\Omega^j(\cdot)$. Finally, we render the display by passing all this information to a shader. In subsequent sections, we describe in detail our calibration, surface tracking and intensity blending methods that come together in the above pipeline to create a seamless, real-time display on a dynamic, deformable surface.

5.4 Calibration

The first step in achieving a registered multi-projector display on a dynamic, deformable surface is to calibrate every projector and camera in the system. As noted in Chapter-3, we model the projectors and cameras with a pinhole camera model extended with radial and tangential distortion, where a 2d point p (of a projector or camera) corresponds to a 3D point d on the surface. The calibration process needs to determine the intrinsics that include the intrinsic matrix $K \in \mathbf{R}^{3\times3}$ and distortion coefficients $D \in \mathbf{R}^{5\times1}$, and the extrinsics that include the rotation $R \in \mathbf{R}^{3\times3}$ and translation $T \in \mathbf{R}^{3\times1}$. Thus, for each device, we have nine intrinsic parameters (two for focal length, two for principal point and five distortion coefficients) and six extrinsic parameters (three each for rotation and translation) resulting

in a total of fifteen parameters per device. We denote the projection function M^i that takes a 3D point d and maps it to a pixel p using the calibration parameters of device i as $p = M^i(d; \{K^i, D^i, R^i, T^i\})$.

In order to calibrate a multi-PDC system, we require correspondences between the cameras and projectors. In static projection mapping systems, this is achieved by structured light scanning, where each projector projects known patterns, while all the cameras capture them. The patterns are decoded to establish pixel correspondences between the projectors and cameras, and are triangulated using multi-view geometry.

However, in a moving deformable surface that changes shape continuously, this approach is inapplicable, especially when the devices are not synchronized. There is no guarantee that the surface shape remained the same when computing camera pixels corresponding to the same projector pixel across multiple cameras. In fact, in all likelihood, the correspondences belong to two different surface shapes. This lack of synchronization implies that the surface cannot be triangulated even if they are correctly decoded. Therefore, the system cannot be accurately calibrated to match the physical setup, a critical requirement for creating a registered display.

In the absence of inter-camera correspondences, we devise a method that leverages the depth from each camera via ray and plane based constraints offered by the pinhole model to achieve accurate calibration despite the surface motion. For this we need to acquire images of some projected patterns. We project a grid of ArUCo patterns as in the method proposed in Chapter-3. Note that the movement of the deformable surface is important for this step. Each projector projects several sequences of ArUCo grids one at a time for a few camera frames, while all cameras capture it. The captured images are then decoded to determine the camera pixels of the ArUCo pattern corners, while the corresponding 3D point is obtained from the depth map. Thus, for the k-th ArUCo pattern m_k^i detected in camera i, we have a 3D point d_k^i, the camera pixel c_k^i and the projector pixel p_k^i.

5.4.1 Camera Intrinsic Calibration

The first step in achieving full calibration is to estimate the camera intrinsics. For each camera, we fix the extrinsics to be at the origin, with the camera looking down the positive Z-axis. Thus, for each camera i, we optimize its intrinsic parameters denoted by $\{K_c^i, D_c^i\}$ using the correspondences (c_k^i, d_k^i) from all markers m_k^i detected in that camera. We minimize the reprojection error E_c^i, as:

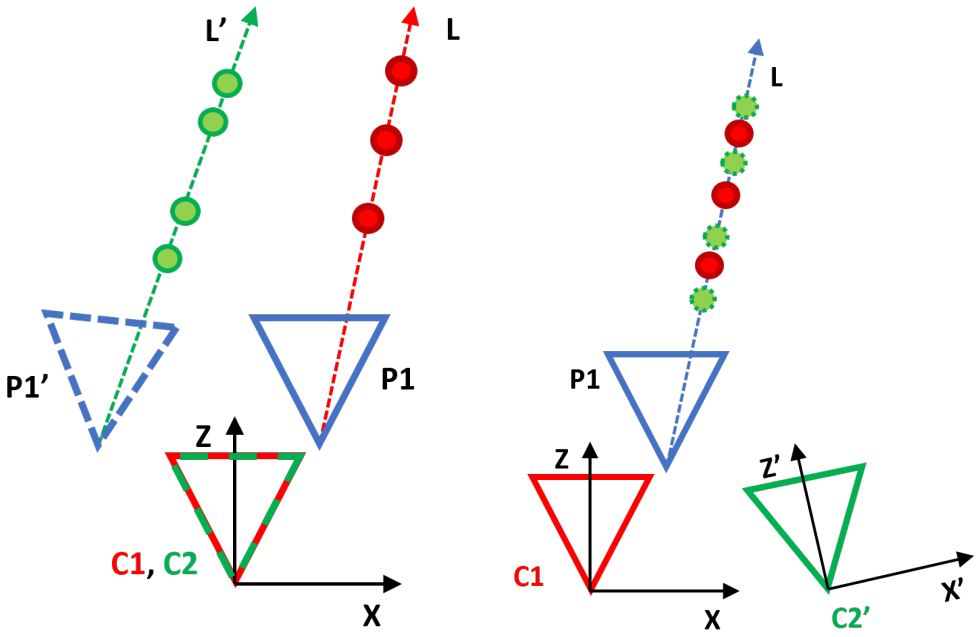
$$\arg\min_{K_c^i, D_c^i} E_c^i = \frac{1}{2}\sum_k (|c_k^i - M_c^i(d_k^i; \{K_c^i, D_c^i, \mathbf{0}, \mathbf{0}\})|)^2. \tag{5.1}$$

We initialize the optimization assuming the distortion coefficients to be zero, and compute the intrinsic matrix K_c^i directly from the 2D-3D correspondences. Each camera's intrinsics are optimized separately. If the camera API provides the camera intrinsics, we use them to initialize the optimization instead. We continue optimizing the parameters until convergence.

5.4.2 Camera Extrinsic Calibration

The largest impact of the absence of inter camera correspondences in an unsynchronized system is estimating the camera extrinsic parameters. Traditional multi-view geometry methods that assume a static surface become inapplicable. We present a new method to estimate the camera extrinsics that uses 3D depth captured by each camera instead of the 2D pixel correspondences.

Consider only the set of pattern corners that lie on the same row of pixels in the projector image when it is displaying a grid of ArUCo patterns on the moving deformable surface. This row of 2D pixels forms a plane in 3D. Each camera captures this ArUCo grid and records the 3D points corresponding to the pixel row of pattern corners. If we fit a plane through



(a) Camera Local Space (b) World Space

Figure 5.3: Illustration of camera extrinsic calibration. Cameras C1 and C2 both capture the world from their local origin and view the same projector at different locations (P1 and P1'). **(a)** They capture 3D points corresponding to the same projector pixel in their local spaces, giving points along ray L (red) and L' (green). However, in world space, points along ray L' must align with points along ray L. **(b)** Our camera extrinsic calibration finds a rigid transform that aligns points on ray L' along the ray L to determine the camera pose.

each set of 3D points acquired by each camera, we will get two different plane parameters, even though they are the same 3D plane in world space. Similarly, consider only 3D points captured by two cameras that correspond to the same projector pixel i.e. ArUCo marker corner. If we fit a line through each of these sets of 3D points, we will get two different line parameters, even though they correspond to the same 3D ray in world space.

Therefore, we need to transform the point clouds such that 3D points from different cameras corresponding to the same projector row/column pixels lie on the same 3D plane, and 3D points from different cameras corresponding to the same projector pixel lie on the same 3D ray. This transform implicitly gives us the camera extrinsics. Figure 5.3 illustrates this concept using points along a ray.

Thus, we perform an optimization that computes a rigid transform by minimizing the error between corresponding 3D lines and 3D planes in the point clouds captured by multiple cameras. We minimize the error, $E^i_{ray}, E^i_{row}, E^i_{col}$, between corresponding lines, row planes and column planes respectively in the 3D point cloud captured by the reference camera and the point cloud captured by the i-th camera transformed by the camera pose R^i_c, T^i_c given by:

$$\begin{aligned} \arg\min_{R^i_c, T^i_c} E^i_t &= E^i_{ray}(d^i, d^{ref}; \{R^i_c, T^i_c\}) \\ +E^i_{row}(d^i, d^{ref}; \{R^i_c, T^i_c\}) &+ E^i_{col}(d^i, d^{ref}; \{R^i_c, T^i_c\}) \end{aligned} \tag{5.2}$$

The error is computed as the distance between the points in the transformed point cloud and their projection on the corresponding ray/plane in the reference point cloud. Once we have each camera's extrinsics, we transform the point cloud from each camera to the world space.

Note that standard 3D point cloud alignment algorithms like ICP (Iterative Closest Point) cannot be used here. Due to the dynamic nature of the surface, the 3D points corresponding to the same pattern corner lie on a straight line. One camera may have captured more 3D points of that pattern than another. Since ICP computes point matches every iteration, it may converge to the incorrect solution because of incorrect matches. Therefore, while ICP serves as a good initialization to the optimization, it does not result in a precise alignment of the point clouds.

5.4.3 Coarse Projector Calibration

At this point, the cameras in the system are fully calibrated. Next, we perform projector calibration. We use the method in Chapter-3 that uses the point cloud to perform a coarse

projector calibration. This is used as an initialization to an optimization that refines the projector parameters.

To calibrate a particular projector, we use all the 3D points corresponding to it that have been captured by all cameras and transform them to world space using the camera extrinsics. Next, we again use the plane based constraint imposed by the pinhole camera model to fit a plane through each set of 3D points that correspond to a row of pixels in the projector image. This gives us a set of row planes. As dictated by the pin hole camera model, we intersect all the row planes to form a 3D line that contains the projector location in world space and find the direction of this 3D line to estimate the projector's X-axis in world space. We repeat this procedure for the set of 3D points that correspond to the same column pixels in the projector image, fitting planes and intersecting them to get the projector's Y-axis direction. Taking a cross product of the X and Y-axes gives the projector's Z-axis. Finally, we intersect the 3D lines corresponding to the X- and Y-axes to get an estimate of the projector's center of projection (aka location).

We also estimate the intrinsic matrix of the projector using the pose estimate and the 2D-3D correspondences of that projector. This coarse projector calibration serves as an initialization to the optimization that refines each projector's parameters.

5.4.4 Fine Projector Calibration

Let $(p_k^j, \tilde{d_k}^j)$ denote the k-th correspondence between a 2D projector coordinate p_k^j and its transformed 3D point $\tilde{d_k}^j$ for the j-th projector. Let $\{K_p^j, D_p^j, R_p^j, T_p^j\}$ denote projector parameters. We optimize these parameters and minimize the reprojection error E_p^j:

$$\arg \min_{K_p^j, D_p^j, R_p^j, T_p^j} E_p^j = \frac{1}{2} \sum_k (|p_k^j - M_p^j(\tilde{d_k}^j; \{K_p^j, D_p^j, R_p^j, T_p^j\}))^2 \tag{5.3}$$

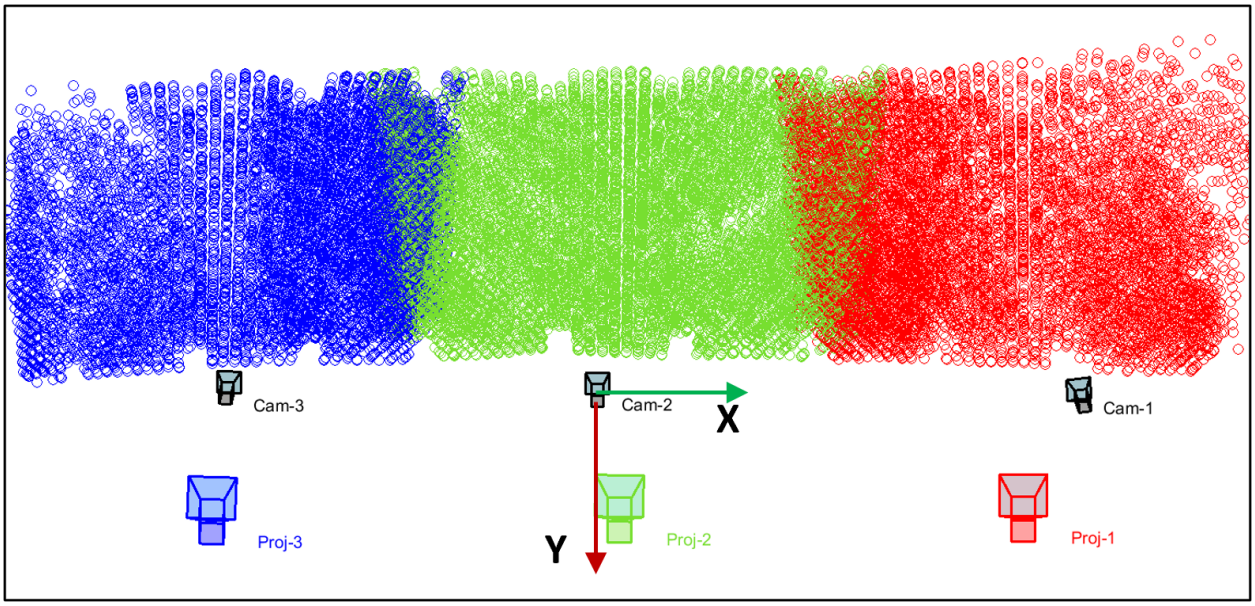


Figure 5.4: The calibration results from our method. The cameras are shown in black, while the three projectors and their corresponding depth data are colored in red, green and blue.

We initialize the projector parameters from the coarse projector calibration in the previous step, assuming distortion coefficients are zero. The optimization is then performed until convergence. Figure 5.4 shows the final, fully calibrated system.

5.5 Multi-camera Surface Tracking

Surface tracking is achieved by a realtime 3D reconstruction of the continuously changing surface shape. The goal of the surface reconstruction step is to determine 3D points on the surface that correspond to the uniformly sampled texture grid, where each texture coordinate is denoted by $s_k = (u_k, v_k)$. Each black marker on the surface corresponds to a known texture coordinate. However, because they are all identical (to keep the system simple for deployment for lay users), we must first determine the (u, v) coordinate of each marker in every camera frame. We employ the method in Chapter-2 to track the surface, but adapt it to handle multiple depth cameras, each observing only part of the surface.

We start by detecting the border markers in the IR frame of each camera and determine the corresponding 3D point from the depth map. Since each 3D point is in the local space of the camera that detected it, we use the corresponding extrinsics to transform each 3D point to world space. However, because the camera FOVs overlap, some markers are detected by multiple cameras. In order to reconstruct the surface accurately, we must identify these duplicate detections and merge them.

Note that while all the 3D points are in world space, they are randomly ordered. To remove this random ordering, we compute the average 3D point of all the points in world space and then sort them all in a clockwise fashion around the average 3D point. This arranges each 3D point with its adjacent neighbor around the display boundary. Next, we proceed to merging markers detected by multiple cameras. For each 3D point, we compare the distances with its left and right neighbors. If the distance between a pair of points is less than a threshold ($\leq 50mm$), we mark those points as duplicates. Each set of duplicate detections is assigned a unique ID and after all duplicates have been detected, the 3D points in each set are merged by averaging them.

Now that we have one 3D point for each marker in world space, we identify the four corners of the display region by computing the angle each marker makes with its two adjacent neighbors. Four markers with angles close to 90° are determined as the display corners. From these four corners, we determine the top-left corner and assign its texture coordinate $(u, v) = (0, 0)$. Since all the markers are sorted in a clockwise fashion, we can assign each marker its corresponding (u, v) starting from the top-left corner, going clockwise around the display boundary.

Next, we compute correspondences inside the display region. Instead of interpolating normals like the method in Chapter-2, we use bilinear interpolation to estimate the internal correspondences. These interpolated 3D points are not guaranteed to be on the surface and using them for rendering will result in visible misregistrations. Assuming a calibrated

multi-PDC rig, we reproject the internal 3D points back into every camera and re-sample the depth at that pixel location if it is valid (see Figure 5.5). We transform these re-sampled 3D points back into world space and average any points that were seen by multiple cameras.

Thus, by tracking the markers from each camera, resampling the surface shape and combining all 3D points in world space, we are able to achieve multi-camera surface tracking. At this point, the entire texture grid has a corresponding 3D point. Finally, we use the projector calibration parameters to determine the corresponding 2D pixels in every projector and compute the function $\Omega^j(\cdot)$.

Note that calibration errors could introduce misregistration artifacts when we transform the points back into world space. However, misregistrations will only be visible in the projector overlapping regions. Since the overlapping regions are seen by multiple cameras and we average the 3D points seen by each camera, this mitigates the misregistration artifacts caused by imperfect calibration.

5.5.1 Extension to Marker-less Display

The proposed method in this chapter can be extended to remove the border markers completely for a marker-less multi-projector display. Note that the border markers have two functions: **(i)** they demarcate the displayable boundary, and **(ii)** they assist with parameterizing the surface. Therefore, any marker-less algorithm must provide these two functionalities for the system to work.

For computing the display boundary, we determine the bounding box of the 3D points of the ArUCo marker corners acquired during calibration. The limits of this bounding box are set to the boundaries of the (u, v) parameter space. Then, each depth camera samples the 3D surface in world space. The (u, v) values corresponding to the these 3D points are computed.

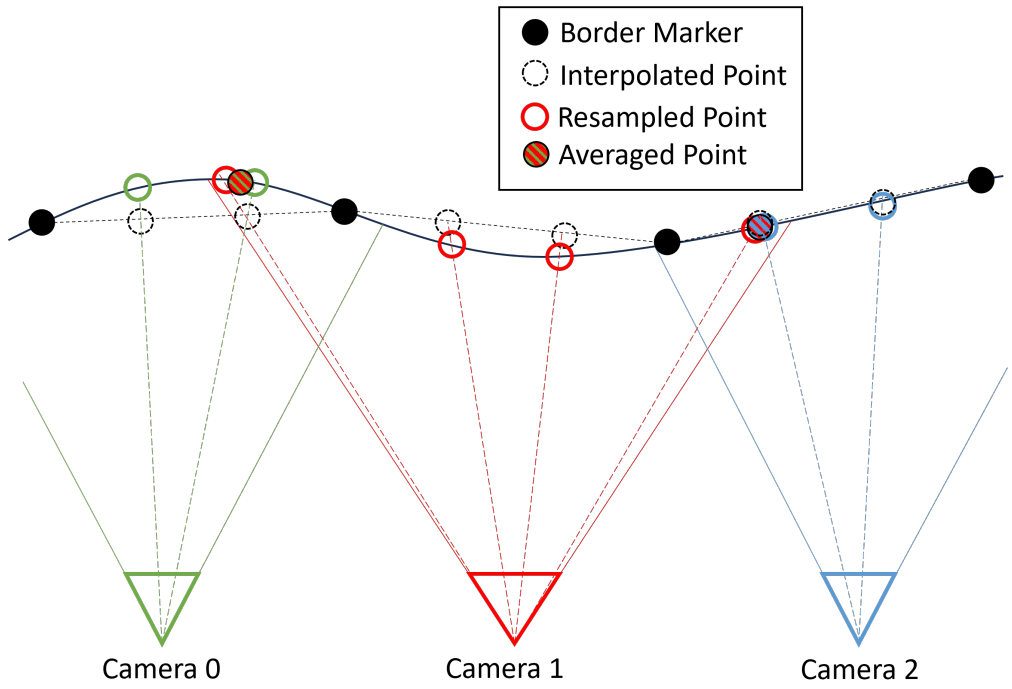


Figure 5.5: Surface sampling by the multi-camera rig. The filled black markers represent the surface border. Points are estimated by linear interpolation between the markers (empty black circles). These points are reprojected into the cameras, which measure the 3D surface at that pixel (colored circles). Points seen by multiple cameras are averaged (striped circles).

Figure 5.6: The projector blend masks in (u, v) space. Note that each mask represents the entire display region.

Then, the (u, v) space is uniformly sampled to compute the 3D points on the surface. These 3D points are then used to warp the projector images accordingly.

5.6 Projector Intensity Blending

To blend the contributions from different projectors in the overlapping region, we need to determine the contribution by each projector to all 3D points. The challenge in a system with dynamic deformable projection surface stems from the fact that these regions are continuously changing. Therefore, the projector blend masks need to be computed every frame efficiently to maintain a real-time FPS (frames per second) of the display.

Prior works on multi-projector dynamic systems only handle rigid objects where the projector light completely floods the target object. In such a rigid object, the surface normals at each 3D point are known apriori. This allows such methods to use the angle of the surface normal with respect to the projector locations (by computing the dot product) to determine the blend weights for each projector. Thus, if the surface normal is at a grazing angle with respect to a projector, the blend weight is close to zero. During runtime, the normals are transformed based on the current object pose to determine the blend weights for each projector contributing to a 3D point. The 3D distance of contributing projectors is also included in the blend weights to account for distance attenuation. Such methods are well suited to handle sharp edges and occlusions of rigid object with respect to projectors.

On the other hand, projector intensity blending on deformable surfaces cannot assume static surface normals and hence, have to be recomputed every frame. However, surface normals are not well suited for blending in our case. Aside from the added latency to compute them every frame, surface normal based blending weights do not feather off the masks to zero at the edges of a projector since the projector rays do not hit the surface at grazing angles, resulting in visible brightness seams. Additionally, noise in the depth camera can impact surface normal computation, resulting in artifacts in the blend masks. The mandate is for the projection intensity mask to fall off gradually to zero the closer it gets to the projection edge in the overlapping region. At the same time, we want to account for the distance of a

3D point to the projectors as well, with the closer projector having a higher contribution. Therefore, we compute distance-based blending weights using the correspondences generated in Section 5.5.

First, for each projector, we pre-compute the 3D planes forming its view frustum using its calibration parameters. Next, for each 3D point d_k on the surface, we check whether its reprojection in projector j is valid. If not, its blend weight is automatically set to zero. Otherwise, we compute **(a)** the 3D distance between the 3D point and its closest view frustum planes for that projector, and **(b)** the 3D distance of the corresponding 3D point to that projector. Finally, the blend weight at the k-th 3D point in projector j, denoted by α_k^j, is computed as:

$$\alpha_k^j = \frac{w_k^j L(d_k)}{\sum_q^N w_k^q L(d_k)}, \quad L(d) = \frac{G(d)^2}{H(d)^2}, \tag{5.4}$$

where d_k is a 3D point on the reconstructed surface, N is the number of projectors, $G(d)$ computes the 3D distance between the 3D point d_k and its closest view frustum plane in projector j, and $H(d)$ computes its 3D distance to the projector location. w_k^j is 1 if the reprojected 3D point is valid in projector j, and 0 otherwise. In other words, if a 3D point in the overlapping region is closer to a projector, that projector has a higher contribution to that point. At the same time, if that 3D point is close to the projection edge, its contribution is reduced.

The result is a blend mask for each projector where the pixel intensities in the non-overlapping regions are one, but gradually fall off to zero in the overlapping regions as they approach the projector edges, resulting in a smooth display. Figure 5.6 shows the blend masks for the three projectors in (u, v) space. Note that each mask represents the entire display region. Therefore, while the falloffs look small in the image, they span 2' (0.7m) on the 3D surface.

Figure 5.7: The final multi-projector display on a moving, deformable surface.

5.7 Results

Figure 5.1 and Figure 5.7 show the full seamless multi-projector display on a deformable fabric achieved by our method. Figure 5.9 shows the markerless multi-projector display on the same fabric. We used three PDCs that illuminated a display surface of size 18'×4' (5.5m×1.2m). The display surface comprises a loosely hanging white fabric. Ten boundary markers annotate the top and bottom edges across, while four markers annotate the left and right edges down the fabric. The PDCs are placed approximately 4' (1.2m) away from the display surface and 6.5' (2m) apart from each other to have a reasonable overlap while still covering the entire display area. Two powerful fans placed on either side of the surface

Figure 5.8: Results of our calibration, rendering and display method. *Left column*: No calibration. *Middle column*: Calibrated result without blending. Note the accuracy of our registration. *Right column*: Our final blended result.

are used to simulate surface motion of varying intensities. Figure 5.8 shows the seamless registration and blending of overlapping projectors.

We used three Azure Kinect RGB-D cameras. The Azure Kinect is a time-of-flight IR/depth camera that provides registered RGB and depth images. It provides multiple operating resolutions. In our work, we used the Azure Kinect in the wide field-of-view setting, capturing 1280×720 RGB-D images at 30fps. This enables us to cover a large display. At this setting, the depth camera resolution is 512×512, significantly smaller than the RGB camera resolution. However, the camera API provides methods that transform and interpolate the depth

Figure 5.9: The markerless multi-projector display from three PDCs.

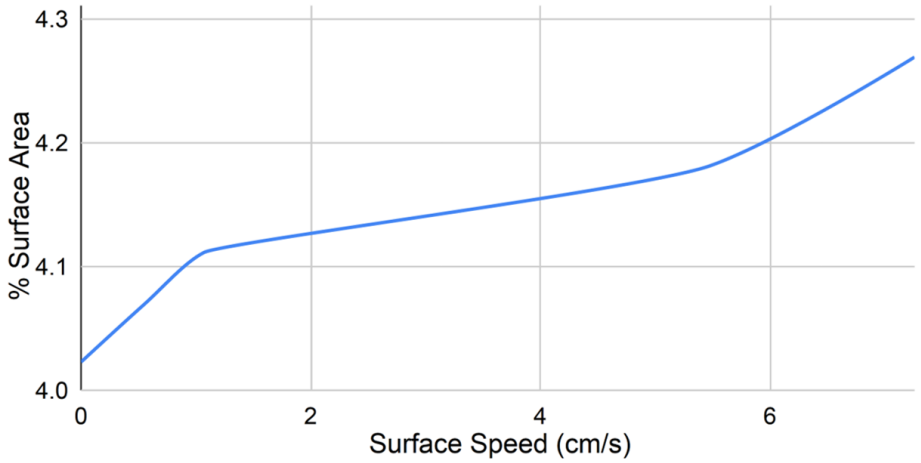
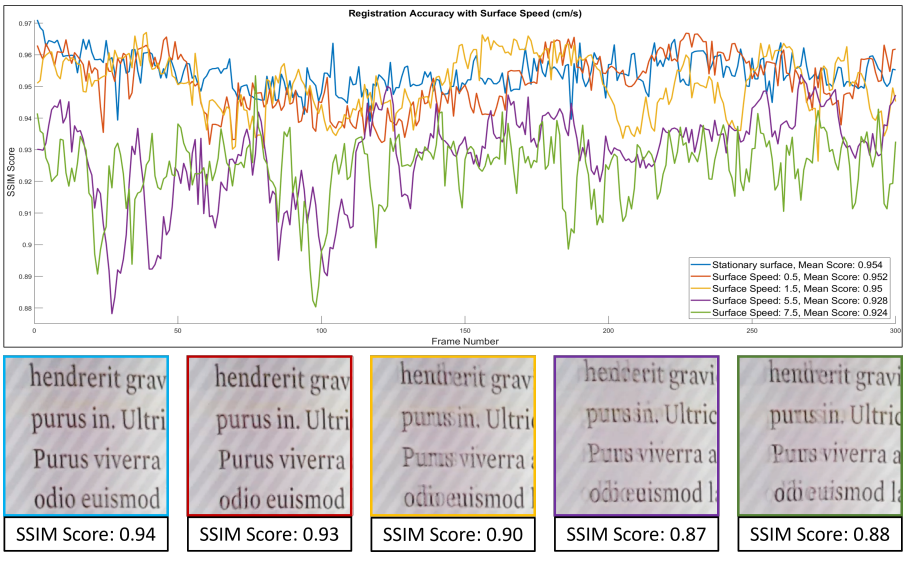
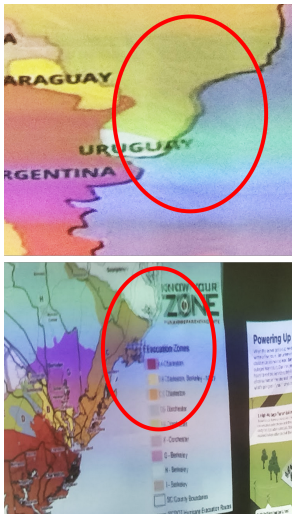


Figure 5.10: Plot of the misregistered surface area (%) vs. surface speed (cm/s), showing the effect of surface motion on the 3D surface reconstruction.

map to the RGB camera space, similar to other consumer-grade depth cameras like the Intel RealSense Depth Cameras [42]. In our work, we use 3D points from this interpolated depth map. For the projectors, we used three Optoma ZH406STx short-throw projectors, operating at 1920×1080 resolution. At a time, only one projector projects an ArUCo marker grid while all cameras capture in parallel. We capture approximately 25 frames per projector. Although the Kinect API provides camera intrinsic and distortion parameters, we have found



(a) Registration Accuracy versus Surface Motion



(b) Momentary misregistrations

Figure 5.11: Effect of surface motion on registration accuracy. **(a)** The graph shows the SSIM score of a video of the overlapping projection compared to the original text image. The images underneath show the projected frames with the lowest SSIM scores at the corresponding surface speeds. **(b)** shows momentary misregistrations for other types of display content.

them to be inaccurate for our application. Therefore, we always refine camera intrinsic and distortion parameters. Note that the cameras are not temporally synchronized, either during calibration or during real-time display. We tested the robustness of our system by placing the PDCs in different positions and orientations with respect to the surface, such as tilted upwards or sideways. So long as each border marker is visible to at least one camera and the display area is completely illuminated by the combined projections, our system operates smoothly.

The proposed system was implemented in MATLAB and C++. Our machine had an Intel Core i7 CPU, with 32GB RAM with a nVidia Quadro P4000 GPU. We used OpenCV APIs for detecting ArUCo markers with sub-pixel accuracy. We used Levenberg-Marquadt optimization to implement our calibration routines. All calibration routines were implemented in C++, except the camera extrinsics calibration, which was in MATLAB. Calibration routines run until convergence, which happens when any one of the following two conditions are

Table 5.1: Reprojection error (in pixels) for the 3-PDC system

Camera Reproj. Error	**Projector Reproj. Error**
0.113	1.156
0.126	1.234
0.124	1.249

Table 5.2: Display run-time breakdown for the 3-PDC system to render a single frame.

Step	**Time (ms)**
Border detection	3.78
Surface Reconstruction	3.37
Blending	3.56
Warping	10.65
Total	**21.36**

met: (1) the number of iterations exceeds a user-defined limit (typically 1000), or (2) the absolute difference between the errors in two successive iterations is less than a threshold $\Delta = 10^{-4}$. It takes approximately 20 minutes to fully calibrate a 3-PDC system, of which ArUCo acquisition takes 40 secions, camera extrinsics calibration takes 11 minutes and projector calibration takes 7 minutes. Note that our code is not optimized and calibration is done only once for the system. The calibration parameters are then used to achieve the real-time seamless projection on the deformable surface. We used OpenGL for rendering.

5.7.1 Evaluation

We evaluated four main aspects of our system: **(i)** calibration accuracy, **(ii)** run-time efficiency, **(iii)** accuracy of the 3D surface reconstruction and **(iv)** display registration quality.

Calibration Accuracy: Table 5.1 shows the reprojection error achieved using our calibration method for each camera and projector in our system. The reprojection error for all cameras is less than 0.2 pixels, whereas for projectors, it averages around 1.5 pixels. The reason for the relatively higher reprojection error for the projectors is due to noise in the depth capture by the Kinects.

Existing calibration methods require the surface to be static, rigid and/or of known shape, none of which apply to our case. The calibration parameters provided by such methods are accurate only for a specific surface geometry. If the surface then moves, we will see misregistrations immediately. In contrast, our method can exploit the surface movement to provide a more accurate calibration.

Runtime Efficiency: Table 5.2 shows the runtime breakdown of our system to render a single frame for the display with 3 PDCs after each RGB-D camera has provided a frame for processing. Note that the total time taken by our system to render a single frame for the deformable surface is 22ms, which is less than the camera frame rate (33ms/frame). Therefore, our system is limited by the camera frame rate and theoretically, we can support cameras where each frame is rendered in 22ms or higher (i.e. maximum of 45 fps).

Surface Reconstruction Accuracy: To evaluate the surface reconstruction accuracy, we computed the surface area of the reconstructed surface that where the depth differed from the surface reconstructed from the dense depth maps by more than 1cm. We captured data at various surface speeds, ranging from stationary to motion mimicking strong gusts of wind. Figure 5.10 shows these areas of the reconstructed surface as a percentage of the total surface area. Notice that with increasing surface speed, the error increases only slightly, reaching a maximum of 4.25%. This amounts to an area of 3 sq. ft. distributed across the total 72 sq. ft. display ($0.25m^2$ of $7m^2$) and is imperceptible. Notice also that there is still some error even when the surface is stationary. This is not due to calibration errors (which would result in visible misregistrations of the display) but the small wrinkles in the fabric which our reconstruction method interpolates over.

Display Registration Quality: We captured a video of the overlapping regions of the final display when projecting text at various surface speeds and compared it to the original image using the Structural Similarity Index Measure (SSIM) [129]. The SSIM score ranges from 0 to 1, with lower values corresponding to poorer quality, while a score of 1 indicates an

exact match. Figure 5.11 shows the variation in the SSIM score over 300 video frames. For smaller surface movements, the SSIM score remains generally high, greater than 0.94. However, as expected, with increasing surface speed, the SSIM score decreases, going below 0.9 for a few frames which is when misregistrations become obvious. This is confirmed by a visual analysis of the overlapping projections as well. The images of the text in Figure 5.11a show the display registration in the video frame with lowest SSIM score for each surface speed across the 300 frames. It is important to note that these misregistrations are only momentary as the display continues to adapt to the changing surface geometry.

5.7.2 Discussion and Limitations

Latency: Although our calibration and surface reconstruction is quite accurate even in strong winds, we still see misregistrations in the final display at high surface speeds (Figure 5.11). This is mainly due to the lag caused by our system. It takes 22ms to render a single frame for a 3-PDC system and this lag is enough to see minor misregistrations in the overlapping regions for a brief moment. One possible hardware solution is to use higher speed RGB-D cameras e.g. 45fps to mitigate these misregistrations. A possible algorithmic solution is to use motion prediction. While prior work has used motion prediction for rigid bodies in a multi-projector DPM setup [110], these techniques are not applicable in our case where the surface is deformable and the movements are random and complex.

Depth Camera Noise: A problem with using commercial depth cameras such as the Azure Kinect is noise in the captured depth map. Using noisy depth for calibrating the system can result in calibration errors. However, our novel calibration method uses ray and plane -based constraints to mitigate those effects. During display runtime however, noise in the depth map manifests as a jittery display. Although the method in Chapter-3 addresses this jitter in the context of DPM by using the Kalman filter to smooth the points, the Kalman

filter introduces some lag which can manifest as misregistrations in the overlapping regions. Therefore, in the future, we would like to explore other more accurate noise removal methods.

Border Markers: We place border markers on the surface to serve as features demarcating the display boundary. This is necessary for deformable surfaces as the projection areas change based on the surface movement. While we have explained a marker-less method of performing DPM in Chapter-2, it relies on a clear depth discontinuity between the surface and the background. This is not guaranteed in our case and we would like to explore methods that do not require any boundary features to perform DPM on deformable surfaces in the future.

Display Readability at High Surface Speeds: Our goal was to create a projection display that conforms to the surface shape as it is perceptibly pleasing for viewers. However, with high surface speeds, the display readability may be compromised due to the rapid surface motion. In the future, we would like to study this effect of rapid surface motion on display readability by conducting user studies and exploring algorithms that can stabilize the projection at high surface speeds.

Display Color Quality: As shown in Figure 5.1(d), Figure 5.7 and Figure 5.8, our distance-based blending method is able to blend the overlapping regions to create a seamless display in real-time. Though normal based blending does not work in this scenario, we would like to explore adapting other color calibration methods such as [119, 60] for deformable dynamic surfaces in the future.

Chapter 6

Applications - Surgical Stencils

In this chapter, we propose an application of D-SAR using the previously mentioned work: a system that provides real-time guidance to surgeons by illuminating salient markings (e.g. points, lines and curves) on the physical surgical site using a projector. In addition to the projector, the system uses a RGB-D camera (e.g. Kinect) for feedback and is driven by a PC. The RGB-D camera provides depth information in addition to an image at video frame rates. This unit is called the projector-depth-camera or PDC unit. The goal is to capture a high resolution mesh of the surgical site using the PDC unit initially. During surgery planning or execution, this digital model can be marked by appropriate incision markings on a tablet or monitor using touch based or mouse based interface, locally or remotely. These markings will then be illuminated at high precision via the PDC unit on the surgical site in real time. If the surgical site moves during the process, the movement will be tracked by the same system and updated quickly on the moved surgical site. This system overcomes the limitations of AR/VR headsets which can overlay information through a display, but are obtrusive and provides visualization only for the surgeons excluding others in the room. Overlaying information, at high precision, directly on the physical surgical site that can be

seen by everyone in the operating room can become an useful tool for skill transfer, expert consultation or training.

6.1 Introduction

Augmented Reality (AR) headsets are being used to assist surgeons in overlaying patient scans (like CT, MRI etc) on the human body [56, 55, 57]. In addition to visualization, this overlay can also aid in pre-operative surgical planning. Instead of marking the incisions and blood vessels by referring to CT scans separately, AR/VR headsets allow marking using cues from the overlaid digital information coming from the headset increasing the accuracy of the markings [56]. Though immersive, wearables are cumbersome and become a barrier to accessing the surgical site directly. Further, the overlaid information is only visible to the surgeon and not to other people (e.g. surgical staff, residents etc.) making communication or collaboration difficult amongst the team. In contrast, our digital overlay via the projector appears directly on the patient's body, making the information visible to everyone. Our projection conforms to the shape of the surgical site putting the digital data in physical alignment with the 3D shape of the surgical site providing the best possible context to improve surgical understanding. It can also be used in preoperative planning for iterative revision of the incision marks that is currently achieved by drawing directly on the surgical site using a marker and erasing and redrawing for changes. This process is not only time-consuming and cumbersome, but often leads to a 'dirty' surgical site where legibility of markings are impacted.

The proposed system falls in the broad umbrella of Projection-based Augmented Reality (PAR) where projectors are used to augment real objects to create visualizations that can be seen together by multiple people [119]. Use of display-based AR (DAR) where a user has to look at a tablet, monitor or headsets to surgical assistance, navigation and visualization

have been explored in recent years. However, prior PAR systems have been explored in a very limited manner. They are usually heavily engineered using custom devices, need preoperative specialized scans of the surgical site which are usually registered manually using fiducials or markers in the operating room (OR). In some cases, 3D shapes are not even taken into account resulting in a loosely registered projection. No practical method exists to bind the projection accurately with movement of the surgical site.

We first use the PDC for a structured light scan to create a high-resolution digital model of the surgical area in the pre-processing stage in around 3 minutes. This model is then loaded in an GUI on another machine (desktop, tablet or laptop) being used remotely or locally by an expert. The expert surgeon then marks the 3D model in the GUI using touch-based or pen-based or mouse based interactions in a tablet or computer (Figure-5.1). The high resolution model is registered to the lower-resolution depth camera frame capture throughout the entire session of the interaction. Any markings made on the digital model shows up at a high precision at the surgical site in real time. Movements of the surgical site is common during the surgery. Therefore, we track the moving surgical area and update the projection quickly so that the digital overlay binds to the surgical site accurately even after the movement.

We demonstrate the PDC based surgical guidance capability on a cleft lip clinical model focusing on craniofacial surgery. The choice of craniofacial surgery for initial demonstration and testing is driven by the computational challenges posed by enabling precise 3D projection on a very a small and richly three-dimensional surgical site of infants. Cleft surgery, in particular, is one of the most challenging and impactful surgeries that require precision preoperative planning in order to ensure a correct and balanced repaired lip. Errors in preoperative planning can result in imbalanced and asymmetric repair which requires follow-up corrective surgeries. Due to the sensitive nature of the preoperative planning for cleft repair, surgeons often have to revisit and revise the surgical markings, a process that can

take 20-30% of OR time (e.g. 1.5-2 hours in a 5-7 hour surgery) due to repeated erasing and redrawing the markings. Using optics based marking from the proposed system also eliminates the special care and attention that has to be devoted to the sterility of the markers and the erasers. Figure-6.1 shows the anthropometric landmarks for a unilateral nasolabial repair. Note that a real surgical site on a 1-3 month old infant will be smaller than the size of this image on a A8 letter-sized paper.

Every step of this work has been done in collaboration two renowned reconstructive surgeons, Dr. Raj M. Vyas and Dr. Lohrasb R. Sayadi, who were the motivation for the application of the PAR technology in this domain. Dr. Vyas is the Chief of Plastic Surgery at CHOC Children's Hospital and Professor of Plastic Surgery and Neurosurgery at the School of Medicine, University of California, Irvine. He specializes in adult and pediatric facial reconstruction of congenital, traumatic and oncologic conditions. Dr. Sayadi is a resident physician in the Plastic and Reconstructive Surgery Department at the University of California, Irvine. Both Dr. Vyas and Dr. Sayadi have been working with us by providing us the models for development of our system and in marking the models with accurate real surgical stencils which are used for evaluation. The system we present has undergone multiple rounds of testing and evaluation by the doctors themselves and their comments are documented in the results section.

Main Contributions: The main contributions of the proposed work are as follows.

1. To the best of our knowledge, we have built the first end to end system that allows digital surgical guidance markings on a 3D model to be transferred accurately on a physical object via a PDC unit at high accuracy in real time;
2. We have developed a method that uses a combination of computer vision and optimization techniques to compute the camera pixel to projector pixel correspondence at the precision required in surgical applications;
3. We have developed a method that uses the PDC unit to project precision surgical markings

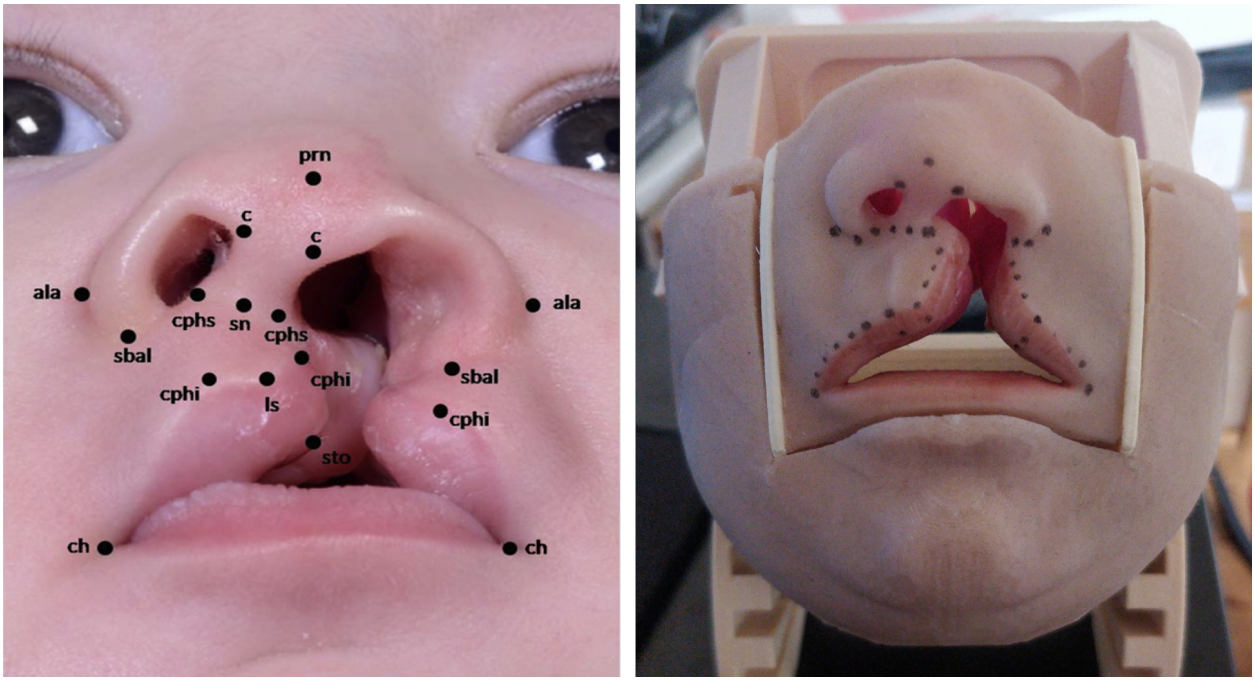


Figure 6.1: (*Left*) Anthropometric landmarks on a unilateral cleft patient. (*Right*) Unilateral cleft model.

that is updated quickly with movement;

4. We have designed a user-friendly GUI that is readily usable by surgeons for skill transfer;

5. We have tested and validated our system using feedback from real surgeons who played with the developed prototype. We have evaluated the system efficiency and functionality using two reputed reconstructive surgeons who anticipate using this system in the OR. Their initial feedback on the accuracy and effectiveness is documented in the results section and is extremely encouraging in terms of scaling the system towards other types of surgeries.

Table 6.1: This table shows the comparison of our method with existing PAR sytems.

Method	Projection based	Automatic Calibration	Handles Movement	3D-shape Reconstruction	Markerless
[117]	✓	N/A			✓
[21]	✓	✓		✓	
[116, 121, 112, 31, 32]	✓				
[130]	✓	✓		✓	✓
[16]	✓			✓	✓
[13]	✓				✓
[19]	✓	✓			✓
Ours	✓	✓	✓	✓	✓

6.2 Related Work

6.2.1 Display-based AR (DAR)

DAR systems overlay information on a digital representation of the surgical site (e.g. image, video, 3D mesh) and show it on a dedicated display device (e.g. a tablet or monitor or 3D display). Wang et.al [127] uses a regular tablet or monitor display to visualize a needle as it is inserted into a patient's teeth during an oral and maxillofacial (OMS) surgery. They use a calibrated stereo camera pair to track the contours of the patient's teeth and reconstruct the 3D model in real time. By tracking the needle using a set of dot markers and registering the reconstructed teeth with a preoperative model, they can visualize the position of the needle tip as it is inserted into the patient on the display. In [124] and [125], Wang et.al extend their work by using 3D display device (i.e. a lens array monitor) through which the patient can be seen. The 3D display is used to overlay and visualize the teeth model on the patient seen through the display. The teeth contours are tracked every frame by a stereo camera setup so that the teeth model realigns with patient movement. However, the visualizations are only visible when viewed through the 3D display. In [126], Wang et.al improve the tracking system by using a single monochrome camera (instead of stereo

cameras) and a novel image registration technique [128] to achieve the digital overlay. In these systems, since the digital representation and the physical object are not overlaid, error in geometry recovery or registration between them, especially with patient movement, cannot be evaluated robustly. Further, while DAR systems are useful to visualize 3D models, on a surgical area, they require users to look into a display. This constant context switch from surgical site to the display impedes the hand eye coordination increasing the time for surgery [**?**].

6.2.2 Projection-based Augmented Reality (PAR)

The limitations of DAR systems can be overcome by projection-based augment reality (PAR) systems. When describing PAR systems we don't consider systems that use projectors only for structured light scanning purposes and not for augmenting physical objects. For example, Edgecumbe et.al [16] develop a small device using a pico projector, called the PicoLantern, for use in laproscopic surgeries. Using the camera in an endoscope and the pic projector, they can perform structured light scanning of organs inside the body. However, the projector is not used to add information on to the 3D surface.

When considering information augmentation by projectors on surgical sites, Tardif et.al [117] propose a rudimentary system to project content on a patient. Using a camera atop the surgical site, they first obtain projector-camera pixel correspondences using structured light patterns. These correspondences are used to warp the content to look correct from the camera's viewpoint. Since the 3D shape of the surgical site is not considered, it can only be useful for projecting on roughly flat areas of the body (e.g. abdomen) and will look correct only from the single view of the camera. Also, since they do not calibrate the projector and camera pair, they cannot bind the projection with patient movement when they repeat the process to find new projector-camera correspondences. Chae et.al [13] proposes another basic

system to project CT scan data on the patient using a pico projector. They assume that the projector is looking directly down on the patient and adjust the height manually until the projection aligns correctly. Fukuhara et.al [19] use a pre-calibrated projector-camera pair to project blood vessels on the abdomen. They compute the camera and projector pose by matching the outline of the abdomen model in the camera image to the CT scan of the abdomen.

Gavaghan et.al [21] design a handheld projector device called an Image Overlay Device (IOD) which they use to project onto a static surgical area. The 3D shape of the surgical site is scanned apriori using a custom high resolution depth camera. The scan is also registered with a camera atop the surgical site apriori. The IOD has a 3D tracker attached to it which is tracked by the aforementioned camera. The same camera is also used to detect markers on the surgical site to align to the digital 3D shape with the physical surgical site. Using the tracked projector pose and orientation and the marker-based alignment of the digital shape with the physical, correct projections can be achieved on the surgical site. Since the camera atop the surgical site, the 3D model and the markers are all tightly calibrated apriori, the system must be recalibrated if the camera, surgical site or the markers move.

Wen et.al [130] propose a projection-based visual guidance system to assist with radiofrequency (RF) ablation needle insertion. The surgical site, i.e. the abdomen, is surrounded by a rig of pre-calibrated stereo cameras and a projector. Structured light is used to find the 3D shape of the abdomen from the calibrated rig. Then, they project the critical structures of the inside of the abdomen on the abdomen itself, including the tumor and the needle trajectory in a view-dependent manner by tracking the viewer's position. In case of movement of the abdomen, the system must be re-calibrated. Other similar systems include Tabrizi et.al [116] for neuronavigation and image-guided neurosurgery. They compute the warping for the projector image by manually registering the projection using five fiducial markers placed on the patient head.

Other works [121, 112] project slices of a CT scan on the body to visualize the internal body structures. They manually align the projection to match physical features on the body e.g. abdomen or breast, and mark important structures inside the body on the skin using a pen/marker. Hummelink et al. [31] also project CT scans on the human body by manually aligning the projector while [32] make projector registration easier by marking four points on the CT scan and placing markers at the corresponding physical locations on the body.

In contrast, our system does not require any markers or pre-calibration and can bind to the surgical site with movement. We achieve this by designing methods that can leverage the low-resolution depth information provided by consumer RGB-D camera (e.g. Kinect) in addition to high resolution color information at video rates. Table-6.1 shows all the PAR systems compared with our proposed system.

6.3 System Overview

Our setup (see Figure-5.1) consists of the PDC unit driven by a PC arranged in a manner such that the surgical area is covered by the field of view of both devices. We assume that the depth camera provides registered color and depth images, as is available in most consumer devices (e.g. Kinect). The user (local or remote) is anticipated to use the system via a graphical user interface (GUI) on a tablet or desktop i.e. a client connected to the PC server. The GUI shows a high resolution 3D digital twin of the surgical site. The user draws surgical guidance marks or stencils (e.g. points, lines, curves, areas) on this digital twin using a preferred interaction modality (e.g. mouse, pen or touch) which are then illuminated precisely on the physical surgical site using the PDC unit in real-time. Also, if the surgical site moves, the stencils bind accurately to it even after movement. Formally, the main objective of our method is to determine the set of projector pixels Ω at each time step which when turned on, illuminate the intended stencils on the surgical site. *We assume*

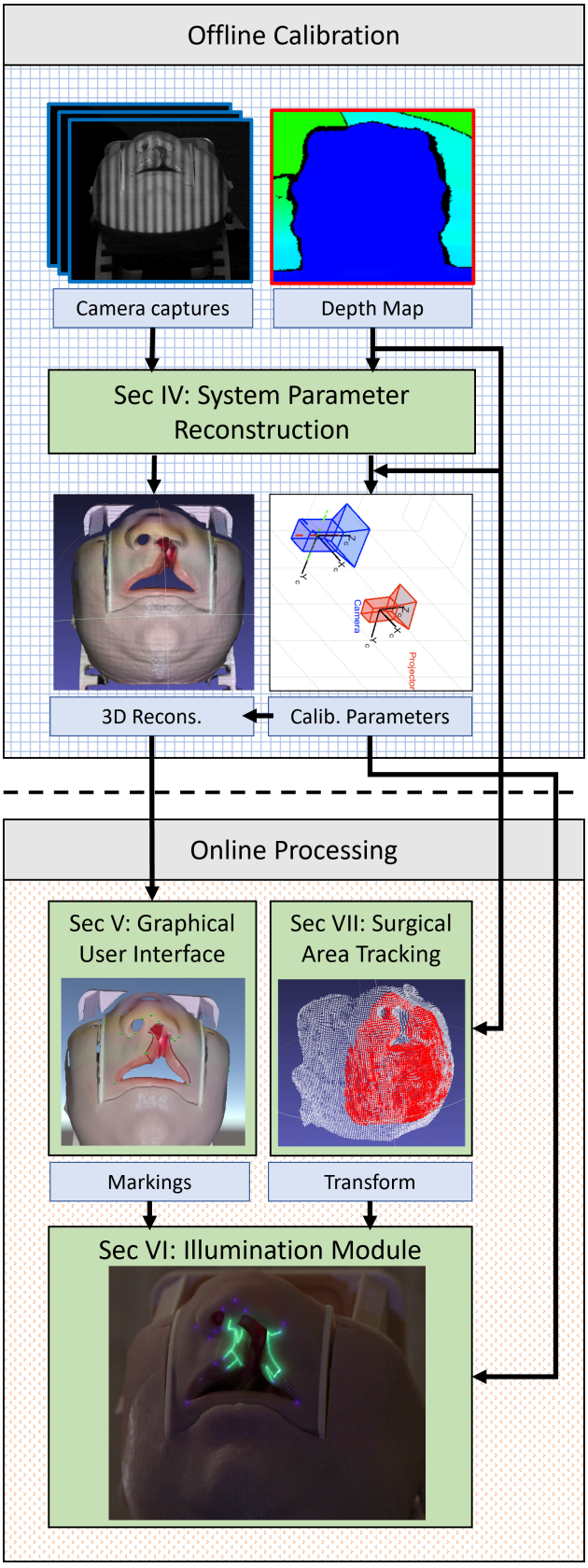


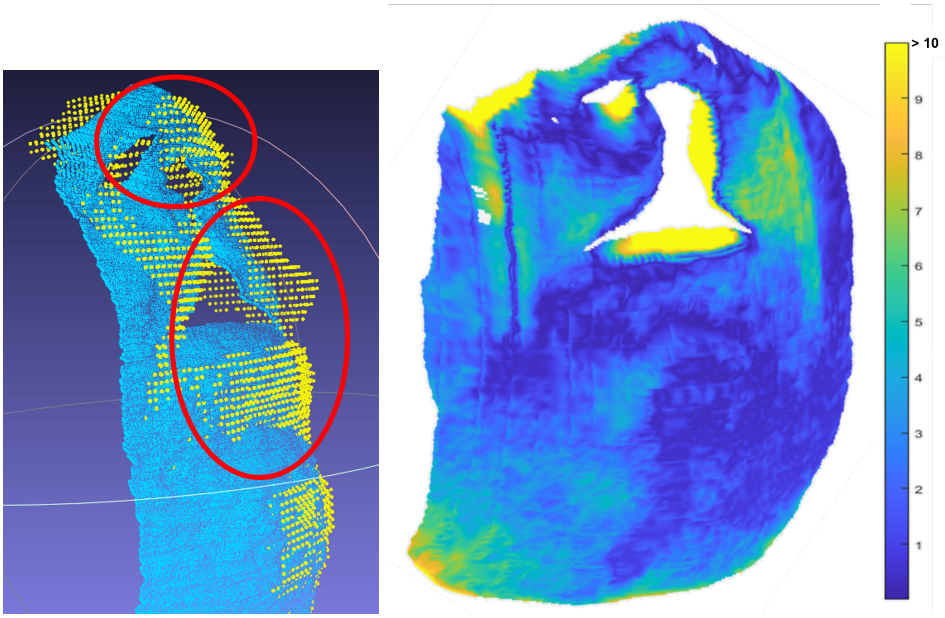
Figure 6.2: The flowchart of our system.

uncalibrated projector and camera with no apriori knowledge of the 3D shape of the surgical site.

Figure-6.2 summarizes our system in a flowchart. It consists of five different modules. First, in the *System Parameter Reconstruction* module, we reconstruct the shape of the surgical site (i.e. digital twin) and recover the parameters of the uncalibrated projector and depth camera in the PDC. The digital twin is then displayed by the *Graphical User Interface* that presents it to the user and allows them to inspect it (via different transformations like rotation and zoom) and mark the surgical stencils on it. The marked stencils and recovered device parameters are fed to the *Illumination Module* that finds the desired Ω in real time and projects it appropriately to illuminate the surgical site accurately by with the appropriate projector image generation module. The surgical area is continuously monitored by the PDC unit. If it moves, the *Tracking Module*, running in parallel with the illumination module, recovers the new location of the surgical site and projects an updated imagery for the illumination module to bind the stencils to the moved surgical site quickly (i.e. within seconds). This is achieved by computing a transformation that takes the structured light reconstruction of the surgical site to its moved location and orientation. The following sections explain each of the above modules in detail.

6.4 System Parameter Reconstruction

The system parameter reconstruction module uses well known structured light reconstruction (SLR) techniques [22, 97, 140] for shape reconstruction. Known patterns are projected from a projector on the surgical site and captured by a camera. The captured patterns are decoded to compute the projector to camera pixel correspondence at every 3D location. These pixel correspondences are triangulated to generate a 3D model (see Figure-6.2). Thus, structured light scanning techniques differ mainly on the pattern design and the decoding algorithm.



(a) (b)

Figure 6.3: Comparison of the depth camera output with the reconstruction using a calibrated PDC unit and structured light. **(a)** The structured light point cloud (cyan) is shown superimposed with the depth camera point cloud (yellow). The red circles highlight some of the mismatched regions. **(b)** The distance (in mm) between the structured light reconstruction and the depth map of the cleft face. Notice how the depth map does not reflect the face geometry accurately, especially in non-planar regions like the lips.

In our work, we use Micro phase-shift (MPS) technique [25] that uses multiple high-frequency sinusoidal patterns in a narrow frequency range. MPS is robust to global illumination effects, inter-reflections and projector defocus, all of which are challenges associated with scanning the human body, especially in a surgical setting. However, it is important to note that our method is independent of the SLR technique used and can be replaced with other methods like Embedded Phase Shift [74].

6.4.1 PDC calibration

The reconstruction of the model can only be as accurate as the SLR pixel correspondence decoding and the projector-camera calibration parameters. Since we start from an uncal-

ibrated PDC unit, we need to calibrate it first. In order to compute the PDC calibration parameters, we again use the pixel correspondence decoding from SLR and approximate the 3D model/depth given by the RGB-D camera available in the PDC unit. We capture the depth map I_D of the surgical area using the RGB-D to get the corresponding 3D point $d_i = I_D(c_i) \in \mathbb{R}^3$ at the camera pixel c_i. Most RGB-D cameras provide this information via their API. Thus, we establish correspondences between the camera, projector and 3D points as a 3-tuple $\{c_i, p_i = L(c_i), d_i = I_D(c_i)\}$ which we use to refine the calibration parameters of the projector-camera pair. The camera calibration parameters, denoted by $\boldsymbol{M^c} = \{\boldsymbol{f^c}, \boldsymbol{o^c}, \boldsymbol{k^c}\}$ include the focal length $\boldsymbol{f^c} = (f_x^c, f_y^c)$, principal point $\boldsymbol{o^c} = (o_x^c, o_y^c)$ and radial distortion coefficients $\boldsymbol{k^c} = (k_1^c, k_2^c)$. For the projector, the calibration parameters include the intrinsics $\boldsymbol{M^p} = \{\boldsymbol{f^p}, \boldsymbol{o^p}, \boldsymbol{k^p}\}$ along with the pose $(\boldsymbol{R^p}, \boldsymbol{T^p})$ relative to the camera.

We assume that the RGB camera is at the origin and looking down the positive Z-axis. Thus, we only need to determine the camera intrinsics. While most RGB-D camera intrinsics are provided by the manufacturer, we noticed that they may not always be accurate and therefore, we optimize for the camera intrinsics using $d_i = I_D(c_i)$ and a standard reprojection solver.

We start with an initial estimate of the projector intrinsics, which can be obtained from [76]. Using the pixel correspondences $\{p_i, d_i\}$ established during the SLR, we get an initial estimate of the extrinsics using the Perspective n-Point algorithm [69]. Note that the extrinsics have the correct scale since we use depth information from the depth camera. Then, we use a standard reprojection solver to optimize the projector parameters. Finally, we perform bundle adjustment using the pixel correspondences and refine the projector extrinsics further by fixing the camera extrinsics. This gives us a good set of calibration parameters which we use for triangulating the SLR pixel correspondences and obtain an accurate 3D digital twin of the surgical area.

Discussion: Despite the presence of the depth camera in the PDC unit, we perform a SLR for the digital twin motivated by the following. (a) While depth cameras are becoming increasingly accurate, they do not capture complex surfaces (e.g. the human face) accurately enough, primarily due to their lower resolution (e.g. the depth camera in Kinect has a resolution of (640×576) that is much lower than the RGB camera at (4096×3072)). Figure-6.3 compares the 3D shape from the structured light scan reconstruction and the depth camera output. Note how the nose is extra pointy in the depth camera, while some parts of the cheek have high error. In order to project precisely onto the surgical area, we need to have an accurate digital twin at the inception to prevent errors due to the inaccurate shape from creeping in. (b) Aside from accuracy, depth cameras also show high quantization errors. The depth at each point is quantized to a whole number instead of a floating point. This loss of precision shows up as inaccuracies in the final projection. SLR solves both these problems. By determining the pixel correspondences and triangulating them, we get an accurate, high resolution 3D shape reconstruction of the surgical site at a high precision.

6.5 Graphical User Interface (GUI)

The high resolution precise digital twin of the surgical area is presented to the user (remote or local) through a GUI, where they can inspect the model by zooming in and rotating it. Additionally, the user can mark points or curves directly on the digital twin. The GUI adds the 3D coordinates corresponding to the marked points and curves in two sets $\boldsymbol{P} \in \mathbb{R}^{N \times 3}$ and $\boldsymbol{Q} \in \mathbb{R}^{L \times Q \times 3}$ respectively, where N is the number of points, L is the number of curves and Q is the number of points per curve.

Marking Points: To mark points, the user hovers the mouse pointer over the 3D surface. We cast a ray into the 3D scene from the rendering camera center through the 2D mouse position. The point where the ray intersects the 3D model is highlighted for the user by

rendering a small sphere. When the user clicks, the sphere is anchored at that point on the 3D model and the 3D coordinates are added to the set $\boldsymbol{P}$. Figure-6.4 shows an example of a user having marked several points (green spheres) on the model. In order to *modify* an existing point, a user clicks on an existing sphere and can place it anywhere else on the model. The corresponding 3D point is updated in $\boldsymbol{P}$. In order to *remove* a point, a user right-clicks on an existing sphere and the corresponding point in $\boldsymbol{P}$ is removed.

Drawing Curves: When drawing curves, our goal is to draw geodesic curves so that it also lies on the surface of the reconstructed digital twin. We use a piecewise linear representation of the geodesic curve and decipher it in two steps. First, the user clicks and drags the mouse across the model surface. The camera shoots rays into the 3D scene at equal intervals between the start and end points of the curve inclusive. Let there be R such rays, and let r_i denotes the 3D point where the i-th ray intersects the 3D model. We render a sphere at each r_i (red and yellow spheres in Figure-6.4) and fit a 3D Bezier curve of degree $(r-1)$ through all r_i's. If we render this Bezier curve as is, there is no guarantee that it will conform to the surface. Parts of the curve may be below or above the 3D model. The former ones will not even be visible to the user. Therefore, we evaluate the Bezier curve for $Q >> R$ points densely spaced along the curve and project each 3D point onto the 3D model. Let q_i denote each projected 3D point. Then, we render a line between each consecutive pair of the projected points $\{q_i, q_{i+1}\}$. This ensures that the final rendered curve is geodesic to the surface geometry. The set of projected points q_i comprising the curve are added to the set $\boldsymbol{Q}$. Figure-6.4 shows an example of a user drawing a curve. In order to *modify* an existing curve, the user can select and move any of the points r_i that the Bezier curve passes through. Upon changing any r_i, all q_i's are updated each frame while the rendered curve is displayed to user. Once satisfied, the user clicks again to finalize the changes. The corresponding curve in $\boldsymbol{Q}$ is also updated. If a user right-click's on an existing curve, the curve is *deleted* from the model as well as from $\boldsymbol{Q}$. ***Please watch the anonymous video at [?] for a demonstration of the proposed system***.

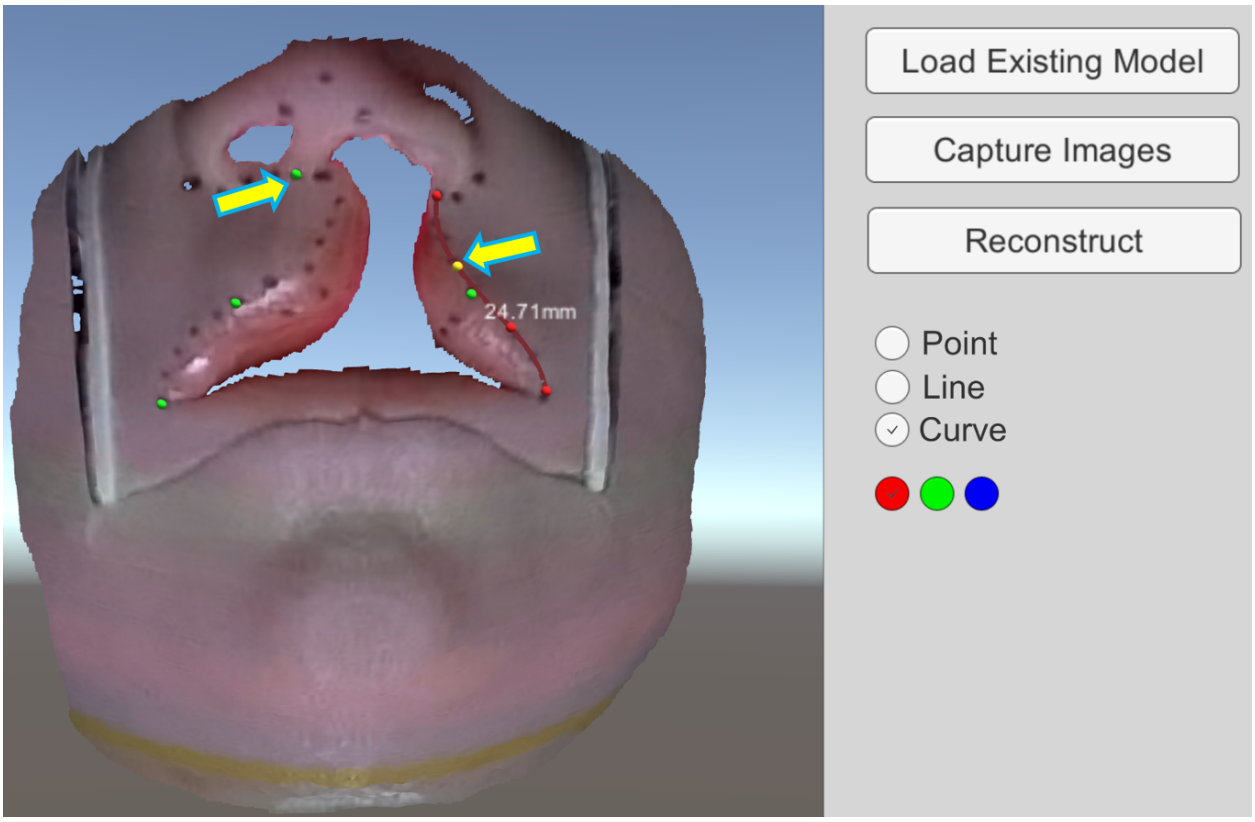


Figure 6.4: A screenshot of our UI. The user can mark points (green spheres) and/or draw curves and lines (red curve).

Table 6.2: Projection accuracy evaluation in mm. θ (in degrees) and δ (in mm) are the rotation and translation of the model from its original position. Nomenclature: *c'*: cleft side, *nc'*: non-cleft side, *m'*: medial, *l'*: lateral.

Keypoint	$\theta = 0$ $\delta = 0$	$\theta = 0$ $\delta = 30$	$\theta = 15$ $\delta = 50$	$\theta = 15$ $\delta = 100$	$\theta = 30$ $\delta = 200$
prn	0	0.16	0.42	0.38	0.19
sn	0	0.26	0.21	0.28	0.25
nc'sbal	0	0.10	0.22	0.12	0.15
c'sbal	0	0.15	0.35	0.13	0.18
m'cphi	0	0.23	0.11	0.25	0.15
l'cphi	0	0.16	0.18	0.40	0.10
c'cphi	0	0.39	0.15	0.40	0.25
nc'ch	0	0.40	0.50	0.58	0.68
c'ch	0	0.47	0.48	0.46	0.48

6.6 Illumination Module

The 3D points marked by the user $\{\boldsymbol{P}, \boldsymbol{Q}\}$ and the projector parameters $\{\boldsymbol{M^p}, \boldsymbol{R^p}, \boldsymbol{T^p}\}$ are used to determine the final set of pixels Ω that must be illuminated to project on the intended regions on the surgical area using

$$\bar{W}_i = \begin{bmatrix} \boldsymbol{R_P} & \boldsymbol{T_P} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} W_i \\ 1 \end{bmatrix} \tag{6.1}$$
$$\omega_i = project(\bar{W}_i; \boldsymbol{M^p})$$

where W_i is a point in $\{\boldsymbol{P}, \boldsymbol{Q}\}$, $\bar{W}_i$ is the same 3D point after applying the projector extrinsics, $\omega_i \in \mathbb{R}^{2\times1}$ is the i-th point in Ω and $\mathbf{0} \in \mathbb{R}^{1\times3}$. All pixels Ω are illuminated in the final image which is then projected onto the surgical area.

6.7 Surgical Area Tracking

During surgery or planning thereof, it is likely that the surgical area will move with respect to the PDC unit. Therefore, we track the surgical area in 3D space using the depth camera. We determine the rigid transformation that minimizes the error between the current depth map from the camera and the depth map captured at the time of SLR. In order to achieve this we use the Iterative Closest Point (ICP) algorithm represented by:

$$\epsilon_{ICP}^{k} = \sum_{i} \sum_{j} v_{ij}^{k} ||d_i^k - \boldsymbol{S}^k \cdot d_j^{k-1}|| \tag{6.2}$$

where d_i^k is a 3D point in the depth map at frame k, v_{ij}^k is a binary variable that is 1 only if ICP matches points d_i^k to d_j^{k-1}, and $\boldsymbol{S}^k \in \mathbb{R}^{3\times4}$ is the rigid transformation computed at frame k. In order to adapt to the movement we modify the projection mapping equation (Equation 6.1) to include $\boldsymbol{S}^k$ as:

$$\begin{aligned} \bar{W}_i &= \begin{bmatrix} \boldsymbol{R_P} & \boldsymbol{T_P} \\ \boldsymbol{0} & 1 \end{bmatrix} \boldsymbol{S}^k \begin{bmatrix} W_i \\ 1 \end{bmatrix} \\ \omega_i &= project(\bar{W}_i; \boldsymbol{M^p}) \end{aligned} \tag{6.3}$$

where W_i is a point in $\{\boldsymbol{P}, \boldsymbol{Q}\}$, $\bar{W}_i$ is the same 3D point after applying the rigid transform and the projector extrinsics, $\omega_i \in \mathbb{R}^{2\times1}$ is the i-th point in Ω and $\boldsymbol{0} \in \mathbb{R}^{1\times3}$. All pixels Ω are illuminated in the final image which is then projected onto the surgical area.

To achieve this efficiently, we use the lower resolution depth map captured during the SLR rather than the high resolution model recovered by the SLR, even though the latter is a more accurate representation of the surgical area. This density of the point cloud from the SLR is much more than the point cloud captured by the depth camera. This results in ICP computing incorrect matches resulting in imprecise S^k as well as becoming inefficient.

Further, when trying to match the depth camera point cloud with the high resolution point cloud from SLR, the noise between consecutive depth camera frame manifests itself in noisy computation of S^k. This results in jittery projection. Using the depth map captured during SLR avoids all these aforementioned issues.

6.8 Implementation and Results

In our setup, we used the Microsoft Azure Kinect RGB-D camera along with an Optoma Technology ML750 DLP projector. The Azure Kinect consists of a depth camera and a RGB camera. It provides different operating modes depending on the capture resolution [70]. Note that the highest RGB camera resolution (4096×3072) is much higher than the corresponding depth camera resolution (640×576). We used the highest resolution (4096×3072) during SLR for a high resolution model. However, for ICP, we use the lowest resolution of the Kinect RGB i.e. (1280×720). The projector resolution is (1920×1080). The surgical models were typically placed around 1.5-2 feet away from the PDC unit to accommodate for the projector focus and Kinect depth sensor limitations, which can measure the depth for objects more than 1.5 feet away.

The intrinsics and extrinsics of the RGB and depth cameras can be accessed through the Kinect API. We use the RGB camera intrinsics to initialize the camera parameter optimization. The Kinect has a depth resolution of upto a millimeter and provides a low resolution raw depth data as well as a higher resolution interpolated depth map rendered from the RGB camera's viewpoint. During calibration, we use the raw depth map since the interpolated depth map has interpolation artifacts like depth quantization. The PDC calibration optimizations were implemented in MATLAB. For computing an initial estimate of the projector intrinsics, we used the projector-camera calibration technique proposed in this thesis i.e. projecting ArUCO markers on a deformable surface to compute calibration parameters. The

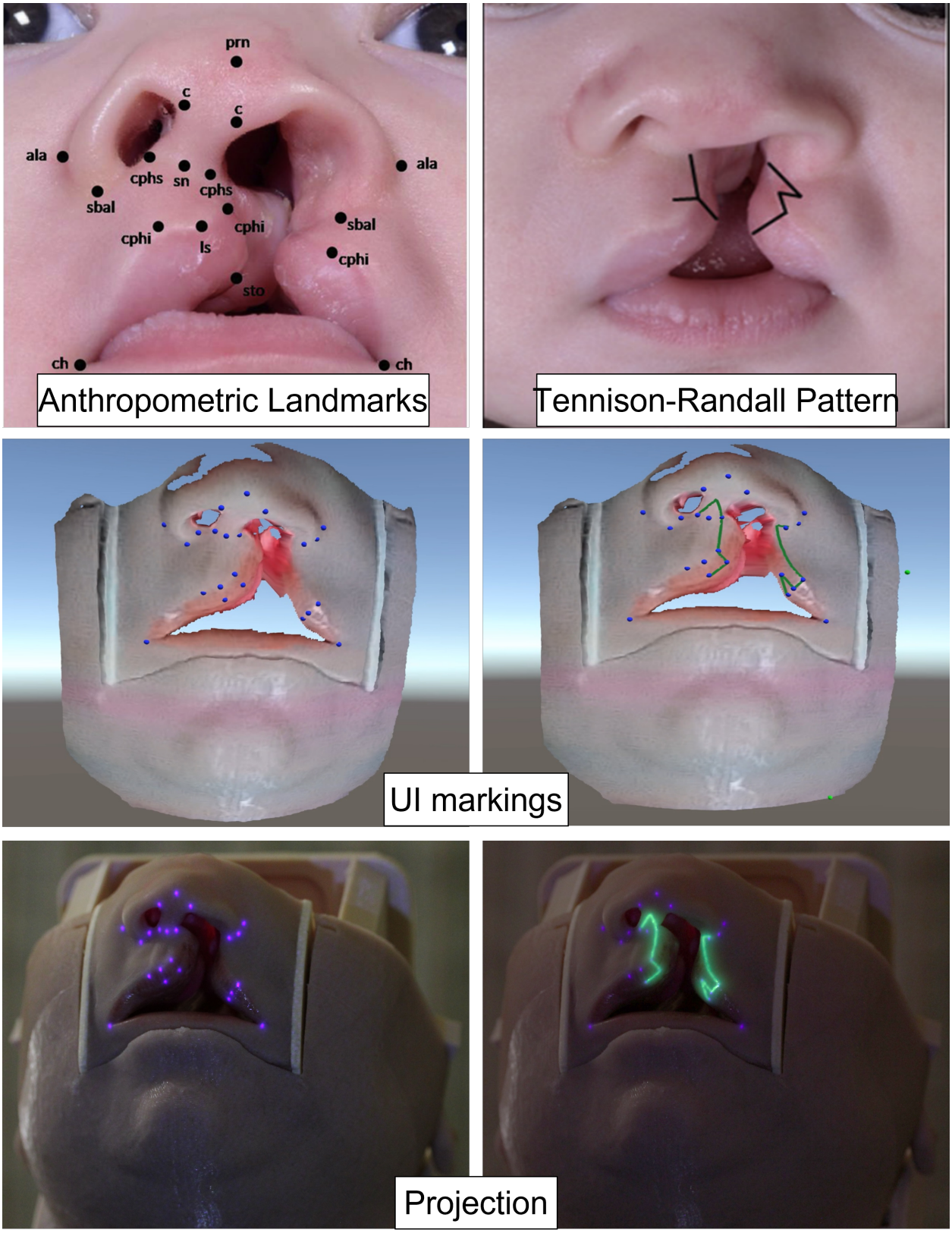


Figure 6.5: Anthropometric landmarks and the Tennison-Randall pattern marked on the SLS through the UI and projected onto the model using our system

GUI and 3D model interaction was developed using Unity rendering engine. For tracking the surgical area, we used OpenCV and the ICP implementation provided by *libpointmatcher* [82, 83].

Figure-6.5 shows the projection of the 21 anthropomorphic landmarks and the Tennison-Randall pattern on the cleft model. The user marks the points and curves on the SLS through the UI. Notice how the projection illuminates the models at the marked points precisely. Figure-6.6 shows the projection of the Mulliken pattern marked through the UI by the user. Even after the model is rotated and translated, the projection realigns to illuminate the correct areas. Figure-6.7 shows the frame-by-frame movement of the projection after the model is moved. ***We strongly encourage the reader to watch the anonymous video at [?] to see the complete system in action***.

The proposed system can also be used for other surgeries. Figure-6.8 shows the proposed system being used to projecting patterns for face reconstruction and breast reduction surgeries. The system reconstructs the surgical area from the SLR, the user marks the points and curves on it and the system projects onto the surgical areas.

6.8.1 Performance and Evaluation

Timing: On average, the image acquisition for SLR took 1 minute while decoding took 3 minutes. The entire calibration and reconstruction process following decoding took 1.5 minutes. Thus, the one-time SLR and calibration took 5-6 minutes. The average time taken to compute a rigid transformation with movement is 0.75 seconds.

Projector Calibration Accuracy: In order to verify the projector-camera calibration accuracy, we performed SLR of a planar board and compared the 3D reconstruction with the depth camera output. The average error was 2.11mm. This was in part due to errors in

the depth camera. The mean reprojection error in the projector after calibration was 0.1821 pixels. This sub-pixel error shows the accuracy of our calibration and 3D reconstruction methods, enabling precise projection on the actual surgical site.

Projection Accuracy: To verify the accuracy of the projection, we marked the 21 anthropomorphic landmarks on the cleft model with a black pen and ran the calibration pipeline. This resulted in the black markers showing up in the digital twin recovered by SLR. Then, we marked those 21 points on the digital twin and measured the distance between the actual marked surface points and their corresponding projections using a digital caliper for different positions of the model. Table-6.2 shows the average error (in mm) for 9 of the 21 landmarks. In general, we noticed that the error for all 21 points was less than 0.5mm, even after rotation and translation. This satisfies the requirements of surgeons, who want sub-millimeter accuracy of the projections and is similar to the accuracy reported by DAR systems as well [125]. However, we did notice a higher error of 0.5-0.6mm for $c'ch$ and $nc'ch$ keypoints. This is because the edges of the lip have more error in the SLR decoding and the depth camera. Note that the error is less than the maximum resolution of 1mm that can be detected by the depth camera. This shows that our method yields extremely accurate results limited by the resolution of the depth camera.

Tracking Accuracy: To verify the accuracy of the model tracking, we attached a board printed with several ArUCo markers to the cleft model. We tracked the ArUCo markers as the model was moved and compared the transforms provided by ICP and the transform computed using the ArUCo markers. The rotation error was 0.1° and the translation error was 0.47 mm.

6.8.2 Domain Validation

We have been working closely with two reconstructive surgeons, Dr. Raj M. Vyas and Dr. Lohrasb Sayadi, who have many years of experience with craniofacial surgery. They have been providing feedback at every level of the development process. All the markings shown in the aforementioned images were performed by the doctors on the computer monitor using our UI and projected on the model by our system. Thus, we received feedback on all salient aspects of the system – ease of use, accuracy of projection and so on. Their comments are as follows:

1. *In the field of plastic and reconstructive surgery, cleft lip repair marking require the greatest degree of precision and accuracy. These markings are defined by both specific anatomical landmarks of the lip as well as geometrical considerations. To make things even more rigorous these points must be placed in the anatomically complex three dimensional space of the child's lip in an area of less than 10 cm^2. The aforementioned technology presented in this manuscript can accurately place these markings with consistency and with minimal error fit by our standards for the operating room. The ability to project on the complex contour of the lip opens up this technology to the whole human body.*

2. *In its current state, this technology is useful for a variety of plastic surgery procedures spanning from the head to the toes. Once we find the ability to project onto surfaces other than the skin (i.e. fat, fluid) in addition to projecting within crevices, then we can expand the technology to intra-thoracic and intra-abdominal surgeries. What sets this technology apart is the lack of headsets which are cumbersome, ergonomically disadvantageous and intrusive to the surgeon. The fact that everyone in the operating room can see the projection allows for better communication between surgical team members. We are looking forward to test this technology is the operating room.*

3. *The accuracy the PAR system achieves is phenomenal. Cleft lip surgery has perhaps the most rigorous surgical markings given the number of markings in such small anatomical confines. The demonstrated ability to achieve low error rates in cleft lip markings bodes well for the clinical translation of this technology to a wide variety of surgical procedures on the surface of the human body. We have not tested the device in the operating room. However, our colleagues who have seen the mannequin projections are amazed by the technology and see a clear room for its application in plastic surgery for a wide variety of surgeries including craniofacial, hand and microsurgery.*

6.9 Limitations and Future Work

The proposed work is just the beginning of an effort to build a device that can be integrated in the OR (for example with the OR lights) to project surgical guidance directly on surgical sites. Such a system can be integrated with AI-based predictive surgical landmark systems such as [12] to completely automate the projection of surgical guidance. However, there are several important technical challenges that have to be overcome which are not addressed by the proposed system.

In the OR, we will have to deal with occlusion of projection due to obstruction by surgeons. Detecting occlusion and compensating for it from other non-obstructing projectors is a possible solution. This builds the motivation to design a system that scales to multiple PDC units. Another limitation of the proposed system is the depth camera, whose limited resolution and noisy reconstruction can cause minor errors in tracking the surgical site. Using noise removal methods and super-resolution techniques to combine depth from multiple PDCs can be a way to alleviate this issue. Finally, the proposed system assumes that the surgical site is semi-rigid and retains its shape. Therefore, it cannot handle soft tissue whose shape can change.

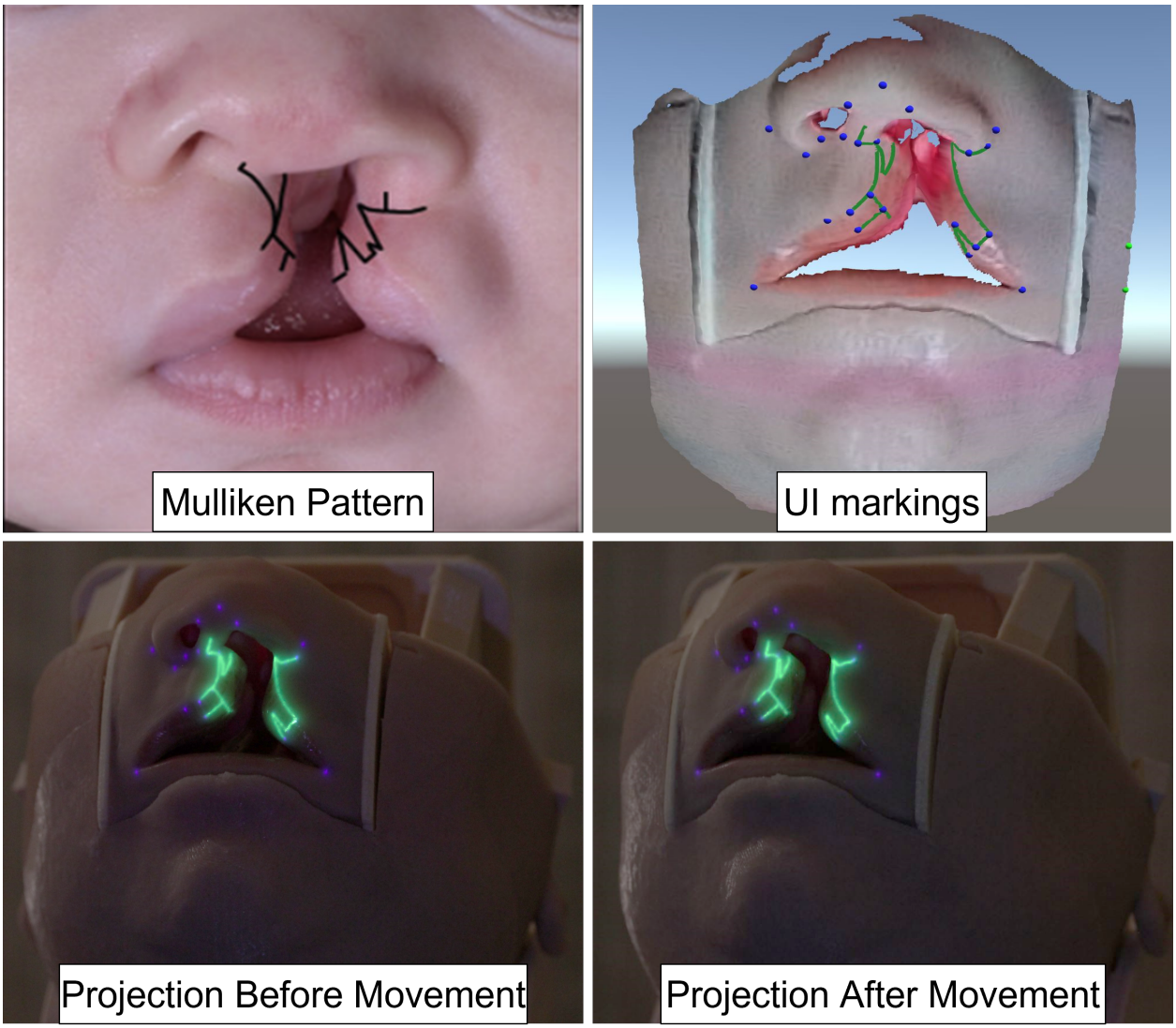


Figure 6.6: Our system can handle movement of the surgical area as well. The projection of the Mulliken pattern marked through the UI and projected onto the model realigns after the model is moved.

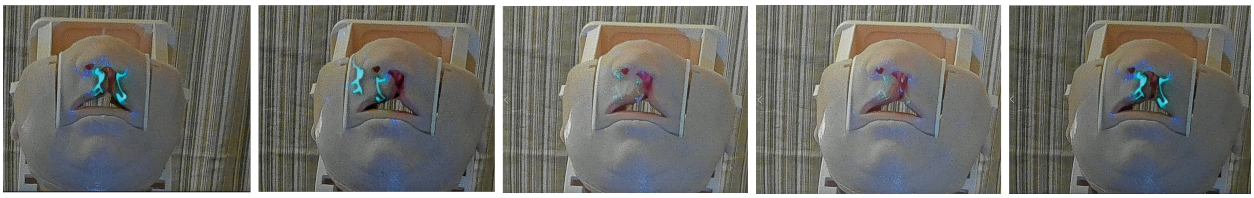


Figure 6.7: This figure shows some frames of the Mulliken projection realigning after it is moved (from left to right).

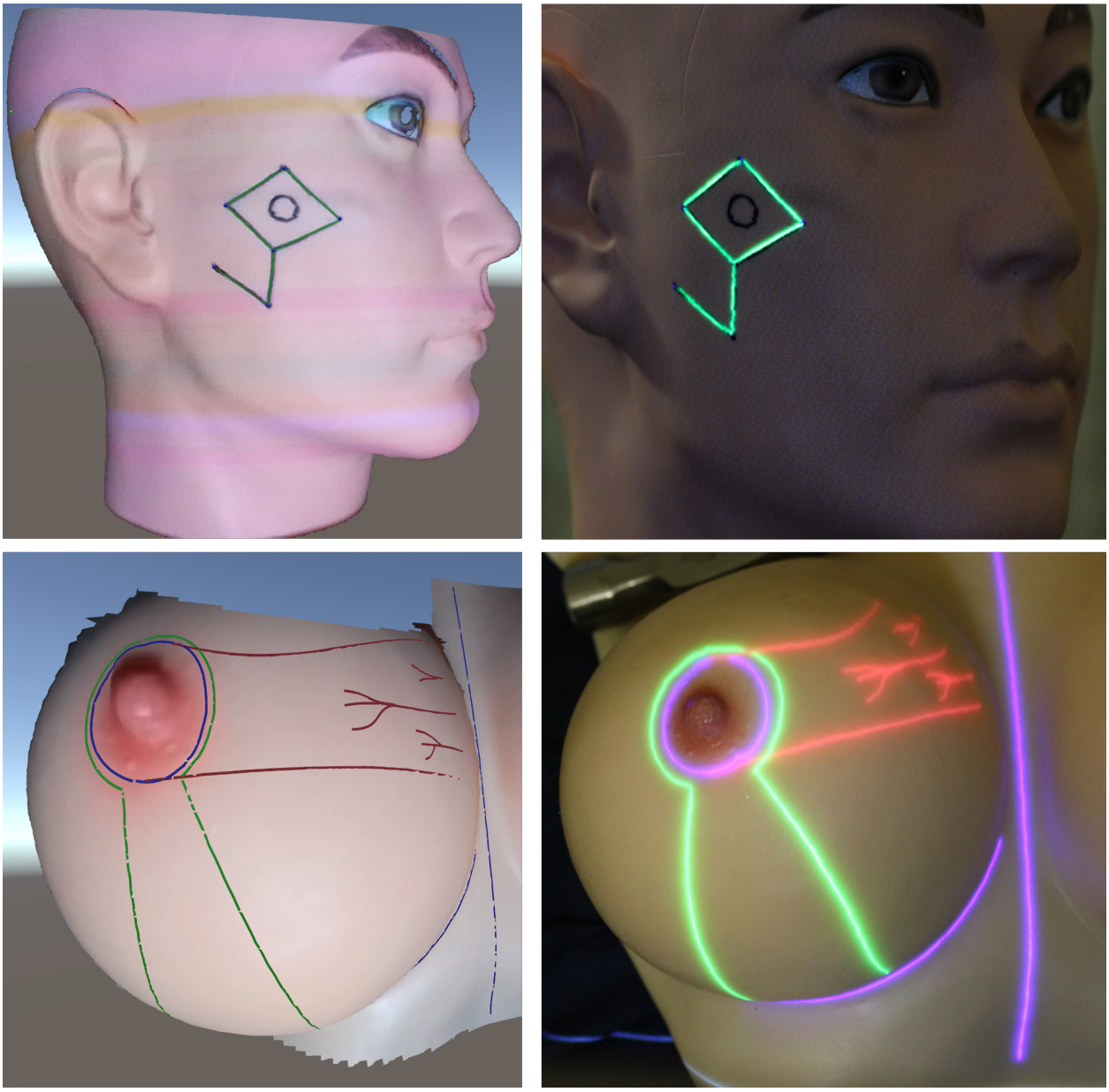


Figure 6.8: The proposed system can be used for other surgical areas as well e.g. the face projected with the rhomboid pattern and a model of a human breast for breast reduction surgery.

Chapter 7

Conclusion

In this thesis, I presented the first end-to-end solution for achieving D-SAR on deformable surfaces. I began by presenting a general B-spline patch-based framework to achieve projection mapping on dynamic deformable stretchable materials using consumer-grade ToF depth camera and the registered IR camera accompanying it. Then, I addressed the problem of accurate geometric calibration and presented the first work on calibration of a projector-RGB-D camera pair using a dynamic and deformable surface. I also addressed important practical challenges in a D-SAR system i.e. a jittery display due to noisy depth cameras and display glitching due to occlusion of the surface. I used a Kalman filter to remove the jitter and handle occluding objects.

Extending this work to multiple projectors, I presented the first a real-time, seamless display on deformable surfaces using mutliple unsychronized projectors without requiring any prior knowledge of the surface or device parameters. The system first accurately calibrates multiple RGB-D cameras and projectors using the deformable display surface itself, and then using those calibrated devices, tracks the continuous changes in the surface shape. Based on the deformation and projector calibration, the system warps and blends the image content in

real-time to create a seamless display on a surface that continuously changes shape. Using multiple projectors and RGB-D cameras, we can provide the much desired aspect of scale to the displays on deformable surfaces.

Bibliography

[1] B. Ahmed and K. H. Lee. Projection mapping onto deformable nonrigid surfaces using adaptive selection of fiducials. *Journal of Electronic Imaging*, 28(6):063008–063008, 2019.

[2] H. Asayama, D. Iwai, and K. Sato. Fabricating diminishable visual markers for geometric registration in projection mapping. *In IEEE Transactions on Visualization and Computer Graphics*, 24:1091–1102, 2018.

[3] S. Audet and M. Okutomi. A user-friendly method to geometrically calibrate projector-camera systems. In *2009 IEEE computer society conference on computer vision and pattern recognition workshops*, pages 47–54. IEEE, 2009.

[4] A. Bermano, P. Brüschweiler, A. Grundhöfer, D. Iwai, B. Bickel, and M. Gross. Augmenting physical avatars using projector-based illumination. *ACM Trans. Graph.*, 32(6), nov 2013.

[5] A. H. Bermano, M. Billeter, D. Iwai, and A. Grundhöfer. Makeup lamps: Live augmentation of human faces via projection. *Comput. Graph. Forum*, pages 311–323, 2017.

[6] E. Bhasker, R. Juang, and A. Majumder. Registration techniques for using imperfect and par tially calibrated devices in planar multi-projector displays. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1368–1375, 2007.

[7] E. S. Bhasker, P. Sinha, and A. Majumder. Asynchronous distributed calibration for scalable and reconfigurable multi-projector displays. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1101–1108, 2006.

[8] W. Bi and B. Xiao. Perceptual constancy of mechanical properties of cloth under variation of external forces. In *Proceedings of the ACM Symposium on Applied Perception*, pages 19–23, 2016.

[9] S. Bougnoux. From projective to euclidean space under any practical situation, a criticism of self-calibration. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 790–796. IEEE, 1998.

[10] K. L. Bouman, B. Xiao, P. Battaglia, and W. T. Freeman. Estimating the material properties of fabric from video. In *Proc. IEEE Int. Conf. Comput. Vis.*, page 1984–1991, 2013.

[11] M. S. Brown and W. B. Seales. A practical and flexible tiled display system. In *10th Pacific Conference on Computer Graphics and Applications*, pages 194–203, 2002.

[12] S. Byun, M. T. Ibrahim, M. Gopi, A. Majumder, L. R. Sayadi, U. S. Hamdan, and R. M. Vyas. Automated landmark detection for ar-based craniofacial surgical assistance system. In *AI Technologies and Virtual Reality*, pages 57–75. Springer Nature Singapore, 2024.

[13] M. P. Chae, D. Ganhewa, D. J. Hunter-Smith, and W. M. Rozen. Direct augmented reality computed tomographic angiography technique (arc): an innovation in preoperative imaging. *European Journal of Plastic Surgery*, 41(4):415–420, 2018.

[14] H. Chen, R. Sukthankar, G. Wallace, and T. J. Cham. Calibrating scalable multi-projector displays using camera homography trees. *CVPR Technical Sketch*, 2001.

[15] J. Choi and I. S. Kweon. Deep iterative frame interpolation for full-frame video stabilization. *ACM Transactions on Graphics (TOG)*, 39(1):1–9, 2020.

[16] P. Edgcumbe, P. Pratt, G.-Z. Yang, C. Nguan, and R. Rohling. Pico lantern: Surface reconstruction and augmented reality in laparoscopic surgery using a pick-up laser projector. *Medical image analysis*, 25(1):95–102, 2015.

[17] Z. Feng, D. Man, and Z. Song. A pattern and calibration method for single-pattern structured light system. *IEEE Transactions on Instrumentation and Measurement*, 69(6):3037–3048, 2019.

[18] Y. Fujimoto, R. T. Smith, T. Taketomi, G. Yamamoto, J. Miyazaki, H. Kato, and B. H. Thomas. Geometrically-correct projection-based texture mapping onto a deformable object. *In IEEE Transactions on Visualization and Computer Graphics*, 20:540–549, 2014.

[19] R. Fukuhara, K. Kaneda, T. Tamaki, B. Raytchev, T. Higaki, S. Nishimoto, and Y. Sotsuka. A projection mapping system onto a human body for medical applications. In *Eurographics (Posters)*, pages 29–30, 2019.

[20] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[21] K. A. Gavaghan, M. Peterhans, T. Oliveira-Santos, and S. Weber. A portable image overlay projection device for computer-aided open liver surgery. *IEEE Transactions on Biomedical Engineering*, 58(6):1855–1864, 2011.

[22] J. Geng. Structured-light 3d surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160, 2011.

[23] E. B. Goldstein and J. Brockmole. *Sensation and Perception*. 2017.

[24] W. Guilluy, L. Oudre, and A. Beghdadi. Video stabilization: Overview, challenges and perspectives. *Signal Processing: Image Communication*, 90:116015, 2021.

[25] M. Gupta and S. K. Nayar. Micro phase shifting. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 813–820. IEEE, 2012.

[26] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[27] N. Hashimoto, R. Koizumi, and D. Kobayashi. Dynamic projection mapping with a single ir camera. *In International Journal of Computer Games Technology*, 2017.

[28] J. Heikkila and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 1106–1112. IEEE, 1997.

[29] B. Huang, S. Ozdemir, Y. Tang, C. Liao, and H. Ling. A single-shot-per-pose camera-projector calibration system for imperfect planar targets. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 15–20. IEEE, 2018.

[30] B. Huang, Y. Tang, S. Ozdemir, and H. Ling. A fast and flexible projector-camera calibration system. *IEEE Transactions on Automation Science and Engineering*, 18(3):1049–1063, 2020.

[31] S. Hummelink, M. Hameeteman, Y. Hoogeveen, C. Slump, D. Ulrich, and L. S. Kool. Preliminary results using a newly developed projection method to visualize vascular anatomy prior to diep flap breast reconstruction. *Journal of plastic, reconstructive & aesthetic surgery*, 68(3):390–394, 2015.

[32] S. Hummelink, L. S. Kool, and D. Ulrich. Displaying inguinal lymph nodes before transplantation in a deep inferior epigastric perforator flap breast reconstruction using an innovative projection method. *Journal of Plastic, Reconstructive & Aesthetic Surgery*, 69(3):376–380, 2016.

[33] M. T. Ibrahim, M. Gopi, and A. Majumder. Auto-calibrating dynamic projection mapping system with jitter and occlusion management. In *2023 22nd IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2023.

[34] M. T. Ibrahim, M. Gopi, and A. Majumder. Projector-camera calibration on dynamic, deformable surfaces. In *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 905–906, 2023.

[35] M. T. Ibrahim, M. Gopi, and A. Majumder. Real-time seamless multi-projector displays on deformable surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 30(5):2527–2537, 2024.

[36] M. T. Ibrahim, M. Gopi, R. Vyas, L. R. Sayadi, and A. Majumder. Projector illuminated precise stencils on surgical sites. In *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 903–904. IEEE, 2023.

[37] M. T. Ibrahim, R. Hafiz, M. M. Khan, and Y. Cho. Automatic selection of color reference image for panoramic stitching. *Multimedia Systems*, 22:379–392, 2016.

[38] M. T. Ibrahim, R. Hafiz, M. M. Khan, Y. Cho, and J. Cha. Automatic reference selection for parametric color correction schemes for panoramic video stitching. In *ISVC (1)*, pages 492–501, 2012.

[39] M. T. Ibrahim, A. Majumder, and M. Gopi. Dynamic projection mapping on deformable stretchable materials using boundary tracking. *Computers & Graphics*, 103:61–74, 2022.

[40] M. T. Ibrahim, A. Majumder, M. Gopi, L. R. Sayadi, and R. M. Vyas. Illuminating precise stencils on surgical sites using projection-based augmented reality. *Smart Health*, 32:100476, 2024.

[41] M. T. Ibrahim, G. Meenakshisundaram, and A. Majumder. Dynamic projection mapping of deformable stretchable materials. In *26th ACM Symposium on Virtual Reality Software and Technology*, pages 1–5, 2020.

[42] Intel. Intel depth camera d435. `https://www.intelrealsense.com/depth-camera-d435/`. Accessed: 2022-09-11.

[43] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *In ACM Transactions on Graphics*, 30, 2011.

[44] H.-S. Jang, M. S. Muhammad, and T.-S. Choi. Removal of jitter noise in 3d shape recovery from image focus by using kalman filter. *Microscopy research and technique*, 81(2):207–213, 2018.

[45] H.-S. Jang, M. S. Muhammad, and T.-S. Choi. Optimal depth estimation using modified kalman filter in the presence of non-gaussian jitter noise. *Microscopy Research and Technique*, 82(3):224–231, 2019.

[46] C. Jaynes, S. Webb, R. Steele, M. Brown, and W. Seales. Dynamic shadow removal from front projection displays. In *Proceedings Visualization, 2001. VIS '01.*, pages 175–555, 2001.

[47] C. Jaynes, S. Webb, and R. M. Steele. Camera-based detection and removal of shadows from interactive multiprojector displays. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):290–301, 2004.

[48] C. Jia and B. L. Evans. Online motion smoothing for video stabilization via constrained multiple-model estimation. *EURASIP Journal on Image and Video Processing*, 2017(1):1–13, 2017.

[49] T. Johnson and H. Fuchs. Real-time projector tracking on complex geometry using ordinary imagery. In *IEEE CVPR Workshop on Projector-Camera Systems (PROCAMS)*, 2007.

[50] T. Johnson, F. Gyarfas, R. Skarbez, H. Towles, and H. Fuchs. A personal surround environment: Projective display with correction for display surface geometry and extreme lens distortion. In *Proceedings of IEEE Virtual Reality Conference*, pages 147–154, 2007.

[51] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen. Real-time hyperlapse creation via optimal frame selection. *ACM Transactions on Graphics (TOG)*, 34(4):1–9, 2015.

[52] R. Juarez-Salazar and V. H. Diaz-Ramirez. Flexible camera-projector calibration using superposed color checkerboards. *Optics and Lasers in Engineering*, 120:59–65, 2019.

[53] S. Kagami and K. Hashimoto. Animated stickies: Fast video projection mapping onto a markerless plane through a direct closed-loop alignment. *In IEEE Transactions on Visualization and Computer Graphics*, 25:3094–3104, 2019.

[54] T. Kawabe, T. Fukiage, M. Sawayama, and S. Nishida. Deformation lamps: A projection technique to make static objects perceptually dynamic. *In ACM Trans. Appl. Percept.*, 2016.

[55] T. Khan, E. G. Andrews, P. A. Gardner, A. N. Mallela, J. R. Head, J. C. Maroon, G. A. Zenonos, D. Babichenko, and J. T. Biehl. Ar in the or: exploring use of augmented reality to support endoscopic surgery. In *Proceedings of the 2022 ACM International Conference on Interactive Media Experiences*, pages 267–270, 2022.

[56] T. Khan, J. T. Biehl, E. G. Andrews, and D. Babichenko. A systematic comparison of the accuracy of monocular rgb tracking and lidar for neuronavigation. *Healthcare Technology Letters*, 9(6):91–101, 2022.

[57] T. Khan, T. S. Zhu, T. Downes, L. Cheng, N. M. Kass, E. G. Andrews, and J. T. Biehl. Understanding effects of visual feedback delay in ar on fine motor surgical tasks. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[58] P. Kurth, V. Lange, C. Siegl, M. Stamminger, and F. Bauer. Auto-calibration for dynamic multi-projection mapping on arbitrary surfaces. *In IEEE Transactions on Visualization and Computer Graphics*, 24:2886–2894, 2018.

[59] P. Kurth, V. Lange, C. Siegl, M. Stamminger, and F. Bauer. Auto-calibration for dynamic multi-projection mapping on arbitrary surfaces. *IEEE transactions on visualization and computer graphics*, 24(11):2886–2894, 2018.

[60] P. Kurth, M. Leuschner, M. Stamminger, and F. Bauer. Content-aware brightness solving and error mitigation in large-scale multi-projection mapping. *IEEE Transactions on Visualization and Computer Graphics*, pages 3607–3617, 2022.

[61] O. Kwon, J. Shin, and J. Paik. Video stabilization using kalman filter and phase correlation matching. In *ICIAR*, pages 141–148, 2005.

[62] V. Lange, C. Siegl, M. Colaianni, P. Kurth, M. Stamminger, and F. Bauer. Interactive painting and lighting in dynamic multi-projection mapping. *In Augmented Reality, Virtual Reality, and Computer Graphics*, 9769, 2016.

[63] V. Lange, C. Siegl, M. Colaianni, M. Stamminger, and F. Bauer. Robust blending and occlusion compensation in dynamic multi-projection mapping. In *Proceedings of the European Association for Computer Graphics: Short Papers*, page 1–4. Eurographics Association, 2017.

[64] I. Leizea, H. Álvarez, I. Aguinaga, and D. Borro. Real-time deformation, registration and tracking of solids based on physical simulation. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 165–170, 2014.

[65] F. Li, H. Sekkati, J. Deglint, C. Scharfenberger, M. Lamm, D. Clausi, J. Zelek, and A. Wong. Simultaneous projector-camera self-calibration for three-dimensional reconstruction and projection mapping. *IEEE Transactions on Computational Imaging*, 3(1):74–83, 2017.

[66] P. Lincoln, G. Welch, and H. Fuchs. Continual surface-based multi-projector blending for moving objects. In *2011 IEEE Virtual Reality Conference*, pages 115–118. IEEE, 2011.

[67] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[68] A. Majumder and R. Stevens. Perceptual photometric seamlessness in projection-based tiled displays. *ACM Trans. Graph.*, 24:118–139, jan 2005.

[69] E. Marchand, H. Uchiyama, and F. Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015.

[70] Microsoft. Microsoft azure kinect. `https://docs.microsoft.com/en-us/azure/Kinect-dk/hardware-specification`. Accessed: 2022-09-11.

[71] Y. Mikawa, T. Sueishi, Y. Watanabe, and M. Ishikawa. Variolight: Hybrid dynamic projection mapping using high-speed projector and optical axis controller. In *SIGGRAPH Asia 2018 Emerging Technologies*, pages 1–2. 2018.

[72] L. Miyashita, Y. Watanabe, and M. Ishikawa. Midas projection: markerless and modelless dynamic projection mapping for material representation. *ACM Trans. Graph.*, 2018.

[73] D. Miyazaki and N. Hashimoto. Dynamic projection mapping onto non-rigid objects with dot markers. In *2018 International Workshop on Advanced Image Technology (IWAIT)*, pages 1–4. IEEE, 2018.

[74] D. Moreno, K. Son, and G. Taubin. Embedded phase shifting: Robust phase shifting with embedded signals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2301–2309, 2015.

[75] D. Moreno and G. Taubin. Projector-camera calibration / 3d scanning software. `http://mesh.brown.edu/calibration/software.html`, 2012.

[76] D. Moreno and G. 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.

[77] G. Narita, Y. Watanabe, and M. Ishikawa. Dynamic projection mapping onto a deformable object with occlusion based on high-speed tracking of dot marker array. In *Proceedings of the 21st ACM symposium on virtual reality software and technology*, pages 149–152, 2015.

[78] G. Narita, Y. Watanabe, and M. Ishikawa. Dynamic projection mapping onto deforming non-rigid surface using deformable dot cluster marker. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1235–1248, 2017.

[79] D. T. Ngo, J. Östlund, and P. Fua. Template-based monocular 3d shape recovery using laplacian meshes. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:172–187, 2016.

[80] T. Nomoto, W. Li, H. Peng, and Y. Watanabe. Dynamic multi-projection mapping based on parallel intensity control. *IEEE Transactions on Visualization and Computer Graphics*, pages 2125–2134, 2022.

[81] J. Pilet, V. Lepetit, and P. Fua. Real-time nonrigid surface detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 822–828, 2005.

[82] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[83] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart. Tracking a depth camera: Parameter exploration for fast icp. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3824–3829. IEEE, 2011.

[84] P. Punpongsanon, D. Iwai, and K. Sato. Deforme: projection-based visualization of deformable surfaces using invisible textures. In *SIGGRAPH Asia 2013 Emerging Technologies*, pages 1–3, 2013.

[85] P. Punpongsanon, D. Iwai, and K. Sato. Projection-based visualization of tangential deformation of nonrigid surface by deformation estimation using infrared texture. *In Virtual Reality*, 19, 2015.

[86] P. Punpongsanon, D. Iwai, and K. Sato. Projection-based visualization of tangential deformation of nonrigid surface by deformation estimation using infrared texture. *In Virtual Reality*, 19, 2015.

[87] P. Punpongsanon, D. Iwai, and K. Sato. Flexeen: Visually manipulating perceived fabric bending stiffness in spatial augmented reality. *In IEEE Transactions on Visualization and Computer Graphics*, pages 1433–1439, 2020.

[88] R. Raskar. Immersive planar displays using roughly aligned projectors. In *Proc. of IEEE VR*, 2000.

[89] R. Raskar, J. v. Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. Ilamps: geometrically aware and self-configuring projectors. In *ACM SIGGRAPH 2006 Courses (SIGGRAPH '06)*, 2006.

[90] R. Raskar, M. Brown, R. Yang, W. Chen, G. Welch, H. Towles, B. Scales, and H. Fuchs. Multi-projector displays using camera-based registration. *IEEE Proceedings Visualization*, pages 161–522, 1999.

[91] R. Raskar, M. Cutts, G. Welch, and W. Stuerzlinger. Efficient image generation for multiprojector and multisurface displays. In *Eurographics Workshop on Rendering Techniques*, pages 139–144. Springer, 1998.

[92] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 179–188, 1998.

[93] R. Raskar, G. Welch, K.-L. Low, and D. Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Rendering Techniques 2001: Proceedings of the Eurographics Workshop in London, United Kingdom, June 25–27, 2001 12*, pages 89–102. Springer, 2001.

[94] C. Resch, P. Keitler, and G. Klinker. Sticky projections — a new approach to interactive shader lamp tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 151–156, 2014.

[95] C. Resch, H. Naik, P. Keitler, S. Benkhardt, and G. Klinker. On-site semi-automatic calibration and registration of a projector-camera system using arbitrary objects with known geometry. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1211–1220, 2015.

[96] P. Roman, M. Lazarov, and A. Majumder. A scalable distributed paradigm for multi-user interaction with tiled rear projection display walls. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1623–1632, 2010.

[97] H. Rubinsztein-Dunlop, A. Forbes, M. V. Berry, M. R. Dennis, D. L. Andrews, M. Mansuripur, C. Denz, C. Alpmann, P. Banzer, T. Bauer, et al. Roadmap on structured light. *Journal of Optics*, 19(1):013001, 2016.

[98] D. Saakes, H.-S. Yeo, S.-T. Noh, G. Han, and W. Woo. Mirror mirror: An on-body t-shirt design system. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016.

[99] B. Sajadi and A. Majumder. Markerless view independent geometric registration of multiple distorted projectors on vertically extruded surfaces using a single uncalibrated camera. *IEEE Transactions on Visualization and Computer Graphics*, 10, 2009.

[100] B. Sajadi and A. Majumder. Auto-calibration of cylindrical multi-projector systems. In *2010 IEEE virtual reality conference (VR)*, pages 155–162. IEEE, 2010.

[101] B. Sajadi and A. Majumder. Automatic registration of multiple projectors on swept surfaces. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, pages 159–166, 2010.

[102] B. Sajadi and A. Majumder. Auto-calibrating projectors for tiled displays on piecewise smooth vertically extruded surfaces. *In IEEE Transactions on Visualization and Computer Graphics*, 2011.

[103] B. Sajadi and A. Majumder. Autocalibrating tiled projectors on piecewise smooth vertically extruded surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 17(9):1209–1222, 2011.

[104] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *In ACM Transactions on Graphics*, 25, 2006.

[105] P. Sen, B. Chen, G. Garg, S. Marschner, M. Horowitz, M. Levoy, and H. Lensch. Dual photography. *ACM SIGGRAPH 2005 Papers*, 2005.

[106] M. Shahpaski, L. Ricardo Sapaico, G. Chevassus, and S. Susstrunk. Simultaneous geometric and radiometric calibration of a projector-camera pair. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4885–4893, 2017.

[107] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[108] S. Shimazu, D. Iwai, and K. Sato. 3d high dynamic range display system. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 235–236, 2011.

[109] C. Siegl, M. Colaianni, M. Stamminger, and F. Bauer. Stray-light compensation in dynamic projection mapping. In *SIGGRAPH ASIA 2016 Technical Briefs*, 2016.

[110] C. Siegl, M. Colaianni, L. Thies, J. Thies, M. Zollhöfer, S. Izadi, M. Stamminger, and F. Bauer. Real-time pixel luminance optimization for dynamic multi-projection mapping. *In ACM Trans. Graph.*, 2015.

[111] C. Siegl, V. Lange, M. Stamminger, F. Bauer, and J. Thies. Faceforge: Markerless non-rigid face multi-projection mapping. *In IEEE Transactions on Visualization and Computer Graphics*, 23:2440–2446, 2017.

[112] Y. Sotsuka, K. Matsuda, K. Fujita, T. Fujiwara, and M. Kakibuchi. Image overlay of deep inferior epigastric artery in breast reconstruction. *Plastic and Reconstructive Surgery Global Open*, 2(10), 2014.

[113] J. Steimle, A. Jordt, and P. Maes. Flexpad: highly flexible bending interactions for projected handheld displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 237–246, 2013.

[114] T. Sueishi, H. Oku, and M. Ishikawa. Robust high-speed tracking against illumination changes for dynamic projection mapping. In *2015 IEEE Virtual Reality (VR)*, pages 97–104. IEEE, 2015.

[115] R. Sukthankar, T.-J. Cham, and G. Sukthankar. Dynamic shadow elimination for multi-projector displays. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, pages II–II. IEEE, 2001.

[116] L. B. Tabrizi and M. Mahvash. Augmented reality–guided neurosurgery: accuracy and intraoperative application of an image projection technique. *Journal of neurosurgery*, 123(1):206–211, 2015.

[117] J.-P. Tardif, S. Roy, and J. Meunier. Projector-based augmented reality in surgery without calibration. In *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No.03CH37439)*, volume 1, pages 548–551 Vol.1, 2003.

[118] M. A. Tehrani, M. Gopi, and A. Majumder. Automated geometric registration for multi-projector displays on arbitrary 3d shapes using uncalibrated devices. *IEEE Transactions on Visualization and Computer Graphics*, 27(4):2265–2279, 2019.

[119] M. A. Tehrani, M. T. Ibrahim, A. Majumder, and M. Gopi. 3d gamut morphing for non-rectangular multi-projector displays. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–15, 2023.

[120] C. Tomasi and T. Kanade. Detection and tracking of point features. *In Carnegie Mellon University Technical Report CMU-CS-91-132*, 1991.

[121] F. Volonté, F. Pugin, P. Bucher, M. Sugimoto, O. Ratib, and P. Morel. Augmented reality and image overlay navigation with osirix in laparoscopic and robotic surgery: not only a matter of fashion. *Journal of Hepato-biliary-pancreatic Sciences*, 18(4):506–509, 2011.

[122] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. 2007.

[123] M. Waldner, C. Pirchheim, and D. Schmalstieg. Multi projector displays using a 3d compositing window manager. *Emerging display technologies*, pages 1–4, 2008.

[124] J. Wang, H. Suenaga, K. Hoshi, L. Yang, E. Kobayashi, I. Sakuma, and H. Liao. Augmented reality navigation with automatic marker-free image registration using 3-d image overlay for dental surgery. *IEEE transactions on biomedical engineering*, 61(4):1295–1304, 2014.

[125] J. Wang, H. Suenaga, H. Liao, K. Hoshi, L. Yang, E. Kobayashi, and I. Sakuma. Real-time computer-generated integral imaging and 3d image calibration for augmented reality surgical navigation. *Computerized Medical Imaging and Graphics*, 40:147–159, 2015.

[126] J. Wang, H. Suenaga, L. Yang, H. Liao, T. Ando, E. Kobayashi, and I. Sakuma. 3d surgical overlay with markerless image registration using a single camera. In *Workshop on Augmented Environments for Computer-Assisted Interventions*, pages 124–133. Springer, 2015.

[127] J. Wang, H. Suenaga, L. Yang, H. Liao, E. Kobayashi, T. Takato, and I. Sakuma. Real-time marker-free patient registration and image-based navigation using stereovision for dental surgery. In *Augmented reality environments for medical imaging and computer-assisted interventions*, pages 9–18. Springer, 2013.

[128] L. Wang, H. Xu, Y. Hu, S. Tabata, and M. Ishikawa. Dynamic depth-of-field projection for 3d projection mapping. In *Extended abstracts of the 2019 CHI conference on human factors in computing systems*, pages 1–4, 2019.

[129] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[130] R. Wen, C.-K. Chui, S.-H. Ong, K.-B. Lim, and S. K.-Y. Chang. Projection-based visual guidance for robot-aided rf needle insertion. *International journal of computer assisted radiology and surgery*, 8(6):1015–1025, 2013.

[131] S. Willi and A. Grundhöfer. Robust geometric self-calibration of generic multi-projector camera systems. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 42–51. IEEE, 2017.

[132] Z. Wu and M. T. N. Ibrahim. Increasing spatial resolution of panoramic video captured by a camera array, 2018. US Patent 10,003,740.

[133] E. Yaman and S. Ertürk. Image stabilization by kalman filtering using a constant velocity camera model with adaptive process noise. *Proc. of ELECO*, pages 152–155, 2001.

[134] S. Yamazaki, M. Mochimaru, and T. Kanade. Simultaneous self-calibration of a projector and a camera using structured light. In *CVPR 2011 WORKSHOPS*, pages 60–67. IEEE, 2011.

[135] L. Yang, J.-M. Normand, and G. Moreau. Practical and precise projector-camera calibration. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 63–70. IEEE, 2016.

[136] R. Yang, D. Gotz, J. Hensley, H. Towles, and M. S. Brown. Pixelflex: A reconfigurable multi-projector display system. In *Proceedings Visualization, 2001. VIS'01.*, pages 167–554. IEEE, 2001.

[137] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[138] M. Zhao and Q. Ling. Pwstablenet: Learning pixel-wise warping maps for video stabilization. *IEEE Transactions on Image Processing*, 29:3582–3595, 2020.

[139] Y. Zhou, S. Xiao, N. Tang, Z. Wei, and X. Chen. Pmomo: Projection mapping on movable 3d object. In *Proceedings Of The 2016 CHI conference on human factors in computing systems*, pages 781–790, 2016.

[140] C. Zuo, S. Feng, L. Huang, T. Tao, W. Yin, and Q. Chen. Phase shifting algorithms for fringe projection profilometry: A review. *Optics and Lasers in Engineering*, 109:23–59, 2018.