MULTI-VIEW VIDEO STITCHING WITH MAXIMIZED FIELD OF VIEW FOR ENHANCED LAPAROSCOPIC VISUALIZATION

by

Alex J. Watras

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2020

Acknowledgments

This work is the culmination of many years of work and would not have been possible without the help and support I have received throughout those years. There are several people to whom I would like to express my sincerest gratitude for their help in this process.

First, my advisor, Professor Yu Hen Hu, your guidance and wisdom has been invaluable in getting me to this point. I am so grateful to have had the opportunity to be a part of your lab and work on this project. Thank you to the rest of my committee. Working with Professor Jiang on this project has been a pleasure and an honor, Professor Gupta may have taught me more about computer vision than anyone else, and Professor Velten's insightful comments and expertise made him a delight to work with.

I thank my parents. My father instilled me with the drive and passion for my work and gave me my first taste of research at the Trout Lake Research Station. My mother's support has given me the confidence in myself that kept me going.

Finally, I would like to thank Sarah Denning. Through long days and tight deadlines, you kept me sane, and I could not have done this without you.

Contents

Contents ii

List of Tables iv

List of Figures v

Abstract ix

1 Introduction 1
1.1 Contributions 4

2 Motivation 8

3 Related Work 10
3.1 Image Stitching 10
3.2 Video Stitching 11
3.3 Optimal Camera Placement 13

4 Multi-Planar Parallax Mitigation 15
4.1 Introduction 15
4.2 Parallax Analysis 17
4.3 Stitching Simple Scenes 18
4.4 Results 26

5 Stitching Complex Scenes 30
5.1 Overview 30
5.2 Multi-View, Multi-Planar Stitching 31
5.3 Application to Surgical Scenes 41

5.4 Discussion 43
5.5 Conclusion 45

6 Optimal Camera Placement for Maximal Field of View Stitching 47
6.1 Problem Formulation 47
6.2 Camera Spaces 59
6.3 Results 64
6.4 Discussion 68
6.5 Future Work 69
6.6 Conclusions 71

7 Conclusion 74

Bibliography 76

List of Tables

5.1 Comparison of Fitting error 43
5.2 Mean Squared Alignment Error (Pixels) on MICCAI 2017 Dataset . . . 44

6.1 Results of Maximal Area Optimization Methods on Surgical Array . . 65
6.2 Results of Maximal Area Optimization on Grid Array 66
6.3 Results of Maximal Convex Region Optimization on Surgical Array . . 68

List of Figures

1.1 Distance from the plane of stitching can cause misalignment such as shown in the red box. These alignments occur because it is impossible to determine how far an object will move from images without knowledge of that object's depth. 2
1.2 The trocar-based camera array offers improved FoV over traditional laparoscopes by utilizing multiple visual sensors. The expanded FoV minimizes the need for camera adjustments during surgery. 6

2.1 A trocar camera array (TCA) consists of several cameras built into a surgical trocar (a). When the trocar is installed (b), the cameras are deployed in such a way that they spread away from the central port of the trocar and begin to collect data from the scene. When the trocar is removed, the cameras fold up inside the trocar so that they will fit through the incision. 9

4.1 A scene consisting of a foreground (dark) object with varying depth and a plain white background. Global homographies are estimated to align background feature points. Discontinuities across the seam line due to parallax is clearly visible. 16
4.2 MPPM Pipeline 19
4.3 Static Homography Alignment Pipeline 20
4.4 Parallax anomaly manifests as discontinuity of the object image along the view boundary 25
4.5 Four video frames consisting of a single surgical grasping instrument against the plain white background of a surgical trainer box. 26

4.6 The red lines in (a) denote the scan lines used for edge detection. Each line will be treated as a one dimensional slice of the image which will be checked for object location. (b) denotes the original difference image that will be sliced by the scanlines. 27
4.7 Each scanline can be displayed as a slice of the image. In each graph, the blue represents the pixel values of a slice of a difference image, and the red line represents a thresholding function used to determine which pixels contain the foreground object. (a) shows a slice that contains an object and (b) shows a slice that does not. 28
4.8 Applying the proof of concept implementation of the MPPM algorithm to the same scene as Fig. 4.1, we are now able to identify the dominant line edges that define the foreground object and compute a transformation that will align that object without changing the background. 28

5.1 An example of a stitched mosaic using (a) traditional single planar stitching and (b) Multi-Planar stitching. The multi-planar stitching has corrected the misalignment of the grasping tools and increased accuracy of the background alignment. 39
5.2 A comparison of the (a) left and (b) right frames from the MICCAI 2017 Dataset along with the stitching results from (c) Single planar stitching, (d) As-Projective-As-Possible Stitching, and (e) Multi-planar stitching 45

6.1 The camera viewing cone is defined by its four corner vectors (B_ij) and the camera center (t_i). 49
6.2 The general flow of our optimization algorithm. For each possible array, we model the coverage region, check to see that the result is a simple polygon which satisfies the overlap constraints, then find the area and compare that to the previously maximum area array. 51

6.3 To determine whether stitching is possible, we need to check that there is sufficient overlap between the views. To do this, we create a graph where each node represents a camera and the existence of an edge specifies that there is enough overlap between the two cameras that stitching could occur. If the resulting graph is connected, then we will be able to create a mosaic from the camera array. 53
6.4 The placement of cameras in the array following the greedy algorithm for various regions of interest (ROI). Blue shapes are the total coverage of cameras in the array, and red indicates the region of interest used. The plot in the lower right of each figure shows the resulting field of view without the region of interest. 55
6.5 The placement of cameras for the greedy algorithm. The blue region denotes what area the camera can see, and the red region denotes the region of interest we wish to cover. 59
6.6 The trocar-based camera array offers improved FoV over traditional laparoscopes by utilizing multiple visual sensors. The expanded FoV minimizes the need for camera adjustments during surgery. 60
6.7 Applying the model to a set of five cameras placed symmetrically on the trocar camera array shows the following: (**a**) the area of the union of the fields of view; (**b**) the area of overlap between Cameras 1 and 2; and (**c**) the cost function that results from thresholding the overlap between each pair of cameras. 63
6.8 Applying symmetric optimization to create camera arrays for use in a laparoscopic trainer box can give significant improvements in total visibility for tasks in the box: (**a**) a frame from the original naive camera system; and (**b**) a similar frame from our symmetrically optimized camera system. The symmetric optimization of camera arrays improves visibility and task performance in the trainer box. No blending techniques were applied so that the individual camera views can be easily picked out from the mosaic. 64

6.9 The resulting views of the four approaches applied to the restrictions of the trocar camera array. 65
6.10 The resulting field of view of the three approaches used for the grid based camera array. 67
6.11 The resulting field of view when optimizing for maximal convex region rather than maximum total area. Blue represents the camera FoV and red represents the found convex region. 67

Abstract

The miniaturization of camera sensors has enabled the replacement of single-camera systems with multi-camera arrays. Small sensors can be mass-produced cheaply and combined into a small package that has the potential to provide multiple viewpoints of a given scene. However, whereas single-camera systems function well as standalone units, multi-camera arrays require extra processing to translate information into a human palatable format. This dissertation will address two major challenges when utilizing multi-camera arrays in place of single-camera systems: parallax mitigation and camera placement.

Existing video stitching techniques allow multiple video feeds to be combined into a single video mosaic, but they require assumptions about scene geometry. Using traditional image stitching algorithms, parallax may occur when objects reside in different planes, violating the co-planar assumption of image stitching. In this work, a multi-planar, parallax mitigation (MPPM) algorithm is proposed to alleviate the parallax anomaly. Multiple planes are used to model the scene and stitching is performed separately on each plane. The separate stitching of multiple planar objects can yield a panoramic image without parallax anomaly, and linear shapes remains unchanged before and after stitching is completed. However, deriving a multi-planar model requires a significant amount of computation. In the proposed MPPM algorithm, prior knowledge of the object shapes is used to develop a fast parallax mitigate procedure. The proposed formulation directly predicts the shape of the multi-planar stitching outcome without explicitly segmenting the object boundaries between video streams. As a result, computation overhead is reduced while visual quality is preserved. This approach can be applied to many other real-time multi-view video surveillance or visualization applications. Here the application is to facilitate laparoscopic surgical vision within a patient's abdominal cavity.

The second challenge is that naive placement of cameras can significantly reduce the total Field of View (FoV) or make it impossible to stitch the videos together. Optimal camera placement offers a potential solution by applying optimization techniques to maximize the total FoV of the camera array. The proposed optimization pipeline attempts to leverage the specific requirements of image stitching to compute camera poses tailored to the desired end goal.

CHAPTER 1
Introduction

From cell phones to automobiles, cameras have become a pervasive part of modern life. Cameras fulfill two very important roles: Cameras fulfill two important roles: first, they offer a way to record moments, freezing them in time and preserving them to be revisited in the future; second, they offer a way to view things that would not otherwise be visible. Driven by these factors, digital cameras have advanced rapidly since the development of the digital camera sensor in 1975. The miniaturization of camera sensors has recently made the replacement of single-camera systems with multi-camera arrays a possibility. Small sensors can be mass-produced cheaply and when bundled into multi-camera arrays can offer a much larger field of view in a small package while offering extra information from different viewpoints.

Cameras allow us to remotely view scenes that we wouldn't normally be able to interact with. When attached to tools, they allow us to interact with scenes even when we aren't physically present. Unmanned vehicles utilize cameras to allow us to explore hazardous terrain. Surgical cameras enable operations that are safer and less invasive. Web-cameras connect us to people and places that are far away. However, these cameras are not perfect. They have limited field of view (FoV), limited spatial and temporal resolutions, and in many cases are required to be stationary, unable to explore their surroundings or move to prevent occlusion. The view that cameras give us of the world is a noisy, 2D projection of a 3D space.

Many of these drawbacks can be alleviated by the introduction of more sensors. FoV can be expanded and blind spots, occlusion, or viewing angles can all be fixed by careful placement of sensors. With two or more cameras viewing the same scene, we begin to recover the 3D information that was lost during the projection process. Extracting, compiling, and conveying the information from these extra cameras

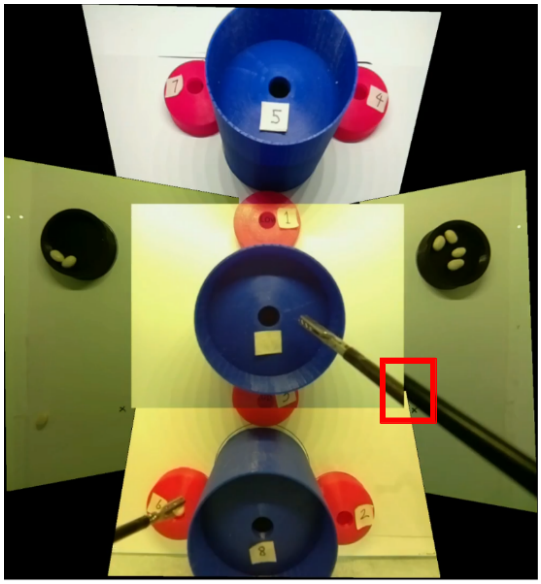


Figure 1.1: Distance from the plane of stitching can cause misalignment such as shown in the red box. These alignments occur because it is impossible to determine how far an object will move from images without knowledge of that object's depth.

pose challenges.

Image stitching algorithms [1, 2, 3] attempt to combine images from multiple cameras into a single composite image called a mosaic. This is done by computing point correspondences between the images and using those to fit a projective transformation (homography) to the image. This homography will align the images based on those detected point correspondences and then blend the images to hide the seams between images. This approach allows for a single unified image to be generated by a camera array.

However, image stitching is heavily reliant on assumptions about scene geometry and camera placement. Either the cameras must all share the same camera center, or the scene that they are recording must lie on a single plane in 3D space. In some cases, these assumptions are valid. Image stitching is often used in settings where the cameras are located far from the scene they wish to record; and although the cameras may not share the same camera center and despite the scene depth, the restricted model used by image stitching may still be a good approximation.

However, the homography used for image stitching will not properly align the images in many cases (See Fig. 1.1). This will occur in situations where the scene depth and camera translation are no longer small relative to the distance from the camera to the scene. As these mi-salignments arise from the inability of the transformation to fit the scene, they can not be addressed by current image stitching methods.

Current approaches to correct these misalignment artifacts can be divided into local warping methods and seam selection methods. Local warping methods [4, 5, 6, 7, 8] replace the homography with new transforms that are able to align any set of feature points but do not accurately reflect the 3D geometry of the scene. Seam selection approaches [9, 10, 7] note that misalignments are only visible across the seam between the two images and focus on placing the seam in such a way as to minimize the possibility of parallax artifacts. The most effective approaches to stitching leverage both seam selection and local warping to generate mosaics that are both locally consistent in their alignment and have any inconsistencies hidden by smart seam selection [11, 10].

Previous parallax correction approaches have focused on generating plausible images for aesthetic purposes. This dissertation focuses on image mosaics created for remote scene interaction. Motivated by the development of a surgical camera system [12], the proposed image stitching method attempts to leverage the 3D properties of the scene to create a mosaic with spatio-temporal consistency that can be relied upon even when a human operator is attempting to interact with a scene that they can sense only through the provided visualization. The near-field nature of the surgical setting means that camera translation will be large relative to the distance from the camera to the scene, and the surgical tools typically lie in planes nearly orthogonal to the scene on which they are operating. As such, it is a setting that exemplifies many of the challenges of remote scene interaction. Furthermore, in a surgical setting, there is no room for error as such methods that attempt to hide inconsistencies can also lead to undesirable consequences.

1.1 Contributions

There are two fundamental research challenge to the problem identified above: (1) The computer vision challenges of generating video mosaics from a camera array under near field conditions, and (2) the challenge of optimizing camera placement into an array that maximizes spatial coverage of the image. Here, new algorithms and techniques in computer vision that address these challenges are presented.

1.1.1 Multi-Planar Stitching

Multi-view video stitching is a process combining synchronous video frames from multiple video streams to produce a panoramic video with larger FoV than any of the individual videos. Parallax is a visual anomaly that manifests itself as image ghosts, discontinuities, and distortions in the stitched panoramic video due to the non-planar scene in the FoV.

In this work, we present novel parallax mitigation algorithms for multi-view video stitching with an application to a novel medical device called a trocar-camera assembly (TCA) for real-time visualization to guide minimally invasive surgeries [12]. TCA can deploy multiple miniature cameras inside a patient's abdominal cavity forming a camera array to construct laparoscopic surgery visualization. Videos captured from these micro-cameras are transmitted to an outside server to be stitched in real-time to provide a video with an enlarged field of view (FoV) which facilitates navigation of surgical instruments within the patient's abdomen.

State of the art multi-view video stitching algorithm pipelines consist of three steps: (a) feature detection at individual views using a common feature detector such as SIFT [13] or SURF [14], (b) feature matching and homography estimation using RANSAC [15], and (c) view warping and pixel value blending. An implicit assumption of image stitching is that all the feature points are co-planar in the real world. As such, the homogeneous image coordinates of the same physical world feature at two different views are described by a linear transformation (homography). To implement such a standard image stitching algorithm on the TCA, two challenges

must be addressed: First, feature extraction and matching requires a significant amount of computation, making real-time implementation impractical. Second, the shallow depth of the abdominal cavity invalidates the co-planar assumption of the feature points. Consequently, parallax may introduce large misalignments in the stitched image.

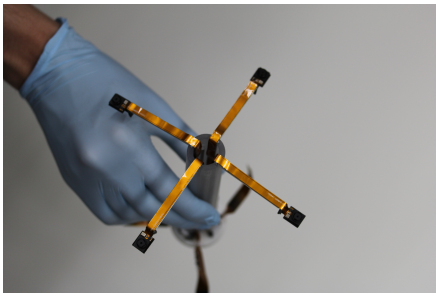
The problem of parallax correction in panoramic images has been studied previously [4, 5, 6, 9, 10]. However, these parallax mitigation solutions are often computationally expensive, and the parallax corrections vary from frame to frame causing temporal visual anomalies. The approach developed here minimizes parallax anomalies while supporting real-time, low latency video stitching. It differs from existing solutions in several ways: (a) the image is segmented into separate planar objects, each of which is fitted with its own homography (b) information from past frames is used to fill in gaps that may occur due to parallax correction (c) the spatial and temporal correlations exhibited in the multi-view video streams are leveraged to reduce the frequency of updates for homography matrices and regions of stitched video frames where no activities are detected. By doing so, a video mosaic can be generated at 16 frames per second (FPS) which is free of parallax discontinuities and does not introduce new spatial distortions.

1.1.2 Optimal Camera Placement for Camera Arrays

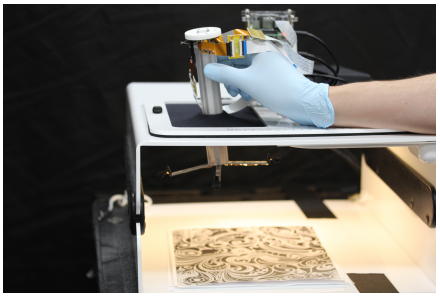
Camera arrays have applications in surveillance [16], robotics [17], virtual reality [18, 19], surgery [20], and more. Multiple cameras placed in proper locations and poses may offer a wider field of view (FoV) by aggregating (stitching) individual images into a coherent, extended mosaic beyond what a single camera can provide.

The traditional camera placement problem has been investigated in the context of video surveillance [21, 22, 23] where cameras were placed in a three-dimensional space to cover a two-dimensional plane [1]. Optimal camera placement under these restrictions has been previously investigated [24, 25].

There are two types of camera placement problem formulations: MIN and FIX



(**a**) Trocar-based Camera Array



(**b**) Array and Laparoscopic Trainer Box

Figure 1.2: The trocar-based camera array offers improved FoV over traditional laparoscopes by utilizing multiple visual sensors. The expanded FoV minimizes the need for camera adjustments during surgery.

[23]. The goal of MIN formulation is to minimize the number of cameras needed to cover a given area. The goal of the FIX formulation is to maximize the coverage area for a fixed number of cameras. The work presented in this paper falls into the FIX formulation category.

The camera placement problem is formulated in the context of building a camera array to enhance the visual quality of a laparoscopic surgery [20]. In current laparoscopic surgery, a single-camera with lighting is inserted through a trocar to provide a view for surgeons to carry out the surgery. Alternatively, a camera array hanging on the trocar assembly would provide a bigger FoV while freeing the surgeon's hand from holding the laparoscopic camera. The cameras are mounted on four arms extending from the housing of a circular trocar, as shown in Fig. 1.2. In this figure, the trocar assembly is mounted on a surgical training box to emulate the surgical setting.

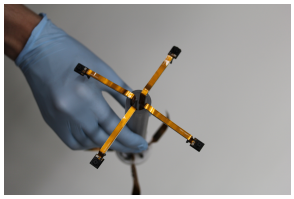
To simplify the hardware design, the cameras are fixed on pre-selected positions on each arm. At each position, the camera has a specific pose selected from a fixed number of choices. This multi-camera system enhances visual coverage during surgery because FoVs from multiple cameras are stitched together. The task is to create an algorithm for optimal camera placement plan that maximizes the stitched FoV subject to several visual quality-related constraints. These constraints make the

camera placement problem formulation distinct from those in the existing literature.

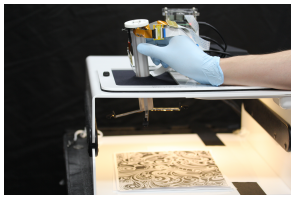
CHAPTER 2
Motivation

The laparoscopic surgical setting can confer substantial health benefits to patients, but it also poses one of the most difficult challenges for remote scene interaction. It is a high stakes environment where small mistakes can have large consequences. Even so, the possible benefits of improving surgical visualization are significant. Minimally invasive surgery can offer significantly reduced recovery time for patients and fewer long term consequences for the surgery. However, the visualization systems currently employed during laparoscopic procedures have a limited field of view and must be navigated through the surgical field to provide adequate coverage. Navigating the camera through the scene has several downsides. First, in order to navigate the camera through the scene, a human operator must use one hand on the camera which means that hand is not available for surgical tools. Second, the moving camera causes a disconnect in spatial awareness. Operators must track the orientation of the camera to know how the movement of the surgical tools will affect the scene that they are viewing. Finally, the limited FoV limits situational awareness of the region near the surgical area.

To address these limitations, a trocar camera array (TCA) utilizes multiple cameras to survey the entirety of the surgical scene in a single view, as in open-cavity surgery (Fig 2.1). While not applicable in all surgical operations, this approach could decrease the barrier to entry for surgeons learning laparoscopic techniques in the abdominal area. The TCA is an array of cameras built into a single surgical trocar. When the trocar is inserted into the patient, these cameras unfold from the body of the trocar to their final positions in the body cavity. The cameras are placed away from the surgical field to reduce the possibility of splatter from the scene and lower the effect of possible occlusion. The multiple cameras of



(**a**) Trocar Camera Array



(**b**) TCA installed in trainer box

Figure 2.1: A trocar camera array (TCA) consists of several cameras built into a surgical trocar (a). When the trocar is installed (b), the cameras are deployed in such a way that they spread away from the central port of the trocar and begin to collect data from the scene. When the trocar is removed, the cameras fold up inside the trocar so that they will fit through the incision.

the TCA mean that the surgeon no longer needs to navigate the scene manually, freeing up that hand for other tasks or allowing the operation to be performed with fewer personnel. However, the effectiveness of the TCA is heavily dependent on the quality of the visualization. The resulting visualization needs to provide the surgeons with the same level of detail about the scene that they can currently gather with a single camera endoscope.

The ideal visualization would behave the same as an open window into the body cavity. The result should be a single video feed provided at >30 frames per second (fps) that accurately shows the spatial relationships between objects in the entirety of the surgical field. This visualization should be overlayed on top of the patient's body as in VTEI [26, 27] so that all movements of the surgical tools behave the same in the visualization as they do in the real world. Generating such a visualization is a difficult challenge, but it's important to keep in mind the ideal while developing more realistic visualizations.

CHAPTER 3
Related Work

3.1 Image Stitching

Modern image stitching is based on the work of Richard Szeliski [1, 28, 29, 30]. The traditional stitching model uses feature matches to compute a 2D projective transformation (homography) which aligns those features between the views of disparate cameras in the array. These transformations are then used to project the images onto a single plane, cylinder or sphere.

The cylindrical and spherical models are often used in virtual reality applications [19, 31, 32, 33, 34, 35, 36, 37, 38] to generate images with a full 360° FoV to surround the user. Planar stitching is more often used to generate composite images from aerial footage [39, 40] or for any scenes that can be assumed to be flat such as whiteboards [29], microscope slides, or any scene where the depth difference in the scene is very small relative to the distance between the cameras or from the cameras to the scene [41]. However, the planar model is generally more useful for any scene when viewing the mosaic on a flat monitor as it creates an image that is much more similar to a traditional camera.

Each model relies on a set of assumptions to function correctly. The cylindrical and spherical models rely on the assumption that the camera is subject to purely rotational motion, and the planar model assumes that the scene being recorded lies on a single plane in 3D space. When these assumptions do not hold, the resulting mosaic can be subject to parallax artifacts where structures become broken across the seams between individual images.

Dual homography stitching [4] broke from Szeliski's global alignment stitching by allowing multiple transformations to be fit to different planar regions of a scene.

This method drastically improved image stitching for landscape imagery where there are typically two dominant planes, one for the ground, and one for the objects on the horizon. This model was further expanded with the introduction of spatially varying warps for parallax stitching [5, 6] which replaced the traditional homography with a new type of transformation that can align even non-planar scenes. As-projective-as-possible (APAP) [3] stitching also attempts to solve the parallax problem by replacing the traditional 2D projective transform with a non-projective transformation that instead seeks to be "as projective as possible" by solving a mixed optimization to simultaneously minimize the number of outlier features and the distance of the transformation from the family of projective transformations. However, these spatially varying transformations are computationally slow and can lead to an unintuitive warping of space which is detrimental to the use of the mosaic for real-time interaction with the physical world.

An alternative approach to parallax correction is optimal seam selection [9, 10]. Rather than changing the transformation model, the seam between different viewpoints is instead chosen in such a way that parallax artifacts are minimalized. These methods seek to find a path across the overlap region that contains only pixels that are similar in both viewpoints. As such there should be no noticeable discrepancies created by the seam. However, simply hiding parallax errors could lead to dangerous inaccuracies in the location of a surgical instrument.

3.2 Video Stitching

Most video stitching approaches have closely followed traditional image stitching. However, the video stitching problem introduces several new challenges and constraints. While a frame by frame image stitching alignment can be used to generate a mosaic video, the resulting mosaic will likely suffer from a visible shaking caused by temporal discontinuities in the stitching alignment. Furthermore, most image stitching approaches rely on feature detection methods which are too slow to run in real-time video stitching applications.

One of the earliest attempts to solve the unique problems posed by video stitching was to use a time-invariant set of homographies to generate the image mosaic [42]. This method allows almost all computation to be offloaded into a calibration step rather than being performed with each incoming frame. It also removes any temporal discontinuities (camera jitter) from the stitching process. However, this method fails to account for changing scene geometry and can perform poorly when the objects in the scene are mobile. Another approach to video stitching has been to focus on using feature tracking to speed up computation and introduce temporal continuity [43]. These methods offer increased robustness to scene motion and allow the implementation of more advanced image stitching techniques such as seam cutting [7, 9, 10] or content preserving warps [4, 5, 6, 7, 8]. However, feature tracking is not a completely solved problem, and various feature tracking algorithms can be fooled by regions with low texture or by occlusion. In addition, once a feature tracking algorithm loses track of a feature, it can be difficult for the algorithm to self-correct.

When using stitched video to interact remotely with a true 3D space, we run into several new issues. Counter-intuitive object motion in the mosaic may reduce the usability of the video feed. As such, these video mosaics need to be approached with a utility first, aesthetics second mindset where any obfuscation of true spatial relations should be weighed against the amount of useful information it may hide. Each mosaic should be considered as an attempt to model the image received by a wide-angle camera placed at a virtual viewpoint location using only the pixel information from our provided cameras. To measure whether or not we have a good model, we must identify what features of the scene are important for navigating the scene and ensure that as many of these features as possible remain in the final mosaic.

The most important scene feature to retain for remote scene interaction is the spatial relationship between the tip of the tool and the nearby scene features. The next most important feature is that apparent motion of the grasping tool must be temporally and spatially consistent with the real world for any given input motion. Any movement the operator wishes to perform should require the same input

regardless of the video frame or tool location within the video frame. Finally, the grasping tool should not be subject to warping across image seams.

3D scene reconstruction [44, 45, 46, 47] can be used to generate a full scene model which can then be projected into any virtual viewpoint. This reconstruction can then be projected using 3D rendering techniques [48] to generate a wide field of view image from many virtual viewpoints. As such, it is a natural option to consider for remote interaction with a three-dimensional scene. It also has the additional benefits of being able to generate mosaics from any number of arbitrarily chosen viewpoints, and the ability to provide the full three dimensions of scene geometry to a user rather than requiring depth be inferred from context clues in an image. It does, however, come with its own set of challenges. Real-time 3D modeling is currently a heavily studied field and still has challenges with generating robust noise-free models in real-time. Current 3D modeling methods also require significant overlap between cameras [44, 45, 46, 47] which leads to a sacrifice in mosaic field of view, a known feature pattern projected onto the scene [49] which can interfere with the task of visualization, or specialized time of flight cameras[50] which can be prohibitively expensive. While solving the problem of real-time 3D reconstruction is one possible solution to remote scene interaction, this paper will focus on finding solutions under situations where full 3D reconstruction is infeasible.

3.3 Optimal Camera Placement

One of the earliest surveillance coverage problem formulations is the Art Gallery Problem (AGP) [51]. The goal was to minimize the number of visual sensors (cameras) required to monitor the floor plan of an art gallery (region of interest, ROI). However, the sensor model in the AGP expected sensors to see any object within line-of-sight of the sensor and limited the camera and surveillance regions to a two-dimensional space. Many existing solutions to the AGP problem do not work well for camera sensors as they assume the sensors have a 360° FoV. As such,

the AGP needs to be reformulated to better reflect real camera models and spatial placement constraints [21, 22, 23].

Some AGP-derived problem formulations attempt to use a two-dimensional floor plan and two-dimensional camera models [52, 53, 23, 54, 55, 56]. These models benefit from their simplicity as camera FoV could be treated as a simple, static two-dimensional shape which enabled many optimization techniques. However, they relied heavily on assumptions about camera placement and scene shape. As such, their usefulness is limited and not applicable to the surgical setting considered in this work.

Several works expand the two-dimensional world view of the AGP into a more realistic three-dimensional case [21, 52, 53, 57, 58, 59, 60, 61, 62]. While many of these models can be simplified when you care about a planar scene, these models require extra computation to be used for occlusion checking which is unnecessary when the scene is planar. These methods typically revolve around discretizing the scene space into a grid and seeking to maximize the number of grid points visible to the camera array. This scene discretization leads to a source of possible error that is not necessary because our scene lies on a plane.

The restriction of 3D scene geometry to a two-dimensional plane has been previously proposed to maximize coverage of a floor plan by surveillance equipment [24, 25, 63] . Fu et al. [24] proposed a two-dimensional coverage model for the placement of cameras in three-dimensional space that is very similar to ours. They used a particle swarm optimization method to perform an optimization that attempts to simultaneously optimize for both maximum coverage of the surveillance space and minimum number of used cameras. However, look at continuity constraints, or look at ensuring overlap as we need to for image stitching. Piciarelli et al. [25] also looked at placing camera sensors in three-dimensional spaces to record scenes on a two-dimensional plane. However, their camera model does not approximate the real-world behavior of a camera as well as the model used by Fu et al. [24].

CHAPTER 4
Multi-Planar Parallax Mitigation

4.1 Introduction

Multi-view image or video stitching combines multiple overlapped images or video frames to form a stitched panoramic image/video frame with a larger field-of-view ([28]). Traditionally, a single homography transformation geometrically aligns key points of one view to those of a reference view in the overlapped region, and warps the remainder of the image accordingly ([1]). By exploiting the temporal correlation between successive video frames, the homography transformations need not be estimated for each video frame. Thus, the majority of computation in a multi-view video stitching algorithm would be applying the homography transformation to each video frame and rendering the stitched panorama video frame. The objective of a real-time multi-view video stitching algorithm is to render the panorama video frames at a desirable frame rate.

The homography matrix enforces geometric consistency across an adjacent pair of overlapping images (views). The purpose of feature detection is to identify distinct feature points in each image. Then, the correspondence of feature points in both views will be established using a robust, iterative RANSAC algorithm. In each iteration, a small subset of matching feature points will be identified, and a candidate homography matrix will be estimated from this set of matching feature points. The estimated homography matrix will then be verified with the remaining feature points. The homography matrix that results in the highest number of correctly aligned feature points will be adopted as the estimated homography matrix between these two views. When there is an insufficient number of feature points available, unless additional visual cues such as structural light are applied,

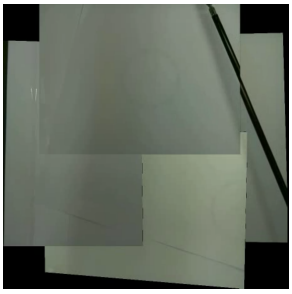


Figure 4.1: A scene consisting of a foreground (dark) object with varying depth and a plain white background. Global homographies are estimated to align background feature points. Discontinuities across the seam line due to parallax is clearly visible.

the stitching cannot be accomplished.

In this work, a real-time stitching of multi-view videos with dynamically moving foreground objects is considered. A foreground object is typically much closer to the camera than objects in the background. The parallax due to foreground objects may cause unsightly visual anomalies in the stitched video frames ([64]). An example of a parallax anomaly is shown in Fig. 4.1. In this figure, four views are stitched together. The image of a dark foreground object crossing two adjacent views exhibits a broken edge anomaly when it crosses the seam line dividing two views. Such parallax anomalies degrade the quality and diminish the utility of the resulting panoramic image.

This dissertation proposes a novel *multi-planar stitching* (MPS) algorithm that can effectively mitigate the parallax anomaly. With the MPS algorithm, the underlying scene is modeled as the composition of a background planar surface and one or more foreground planar surfaces. Different geometric transformations (homographies) will be estimated and applied separately on each surface. The transformed images will be pieced together to form the final panoramic image. In some cases, the homography of some foreground surfaces may not be computable due to the lack of a sufficient number of distinguishable feature points on the surface. The MPS algorithm utilizes 3D line alignment that can effectively estimate the geographically transformed line orientation without relying on feature points. A version of the MPS algorithm was developed for a prototype laparoscopic surgical visualization

system and demonstrate the feasibility of such an algorithm for real-time multi-view stitching of a dynamic scene. When the viewpoint of the stitched multi-view image coincides with a reference image, the proposed line algorithms can be significantly simplified to support time-sensitive multi-view video stitching. Since the reference view is part of the stitched image, the movement of the edge in the reference view can be tracked and the edge can be extended over the seam line without re-estimation of the edge positions each frame. This algorithm was implemented on a desktop computer and observed a multi-view video stitching rate of 16 frames per second with 4 simultaneous video streams at 640 by 480 resolution each.

4.2 Parallax Analysis

Parallax is a phenomenon wherein the velocity of the 2D object created by projecting a moving object in 3D space onto the 2D image plane is dependent on the distance of that object from the camera center. Its importance to the field of image stitching arises from the artifacts that the phenomenon can introduce when a planar homography is used to align non-planar scenes. Objects which do not lie on the plane of stitching will be misaligned by the homography resulting in these objects appearing in incorrect locations. The magnitude of this misalignment will vary based on distance from the stitching plane, and thus objects may appear stretched or warped by the effects of parallax with regions closer to the plane being subject to less misalignment than regions further away. These errors are most noticeable in the regions of overlap between views in image mosaics where the misalignment can cause objects to appear broken or duplicated.

There are two major approaches when attempting to correct for the parallax effect on image mosaics. Seam selection approaches [9, 10, 11, 65] leverage the fact that parallax artifacts can only occur in across the seam line in regions with texture or other measurable features. They attempt to cleverly place the seam lines in locations that do not contain any features so that parallax will be undetectable. Local warping methods [4, 6, 60, 66] attempt to use a spatially varying transformation to account for parallax in the scene. These approaches warp the scene to better fit the detected

feature points without a full understanding of the scene geometry. Due to the fact that seam cutting approaches modify the image blending phase of stitching and local warping approaches modify the warping phase, some methods attempt to combine both approaches to further increase the quality of image stitching [5, 41].

These approaches all sacrifice spatial relationships within the scene for the sake of increased aesthetic appeal. They also are not designed to run on real-time video, with even stripped-down versions of the algorithms taking approximately .5 seconds per frame to complete and more robust methods usually taking on the order of 5-40 seconds per frame. Additional challenges for using video stitching in real-time applications are discussed in section 3.2

4.3 Stitching Simple Scenes

Image stitching attempts to model the viewpoint of a single camera within the array. As such, no data from the viewpoint camera should be modified. Instead, the other images are warped to match the viewpoint camera coordinate system using a dual homography approach where one static homography aligns the scene background, and a second feature tracked homography aligns the remote grasping tool. As the grasping tool is feature-poor, it is a poor candidate for the standard feature-based alignment. Instead, an important Lemma about the homography transformation can be utilized:

Lemma 1: The homography transformation maps straight lines to straight lines.

Proof: Recall that any straight line L can be expressed in the form of $\mathbf{p} + \mathbf{q}\alpha$ where $\mathbf{p}$ and $\mathbf{q}$ are points and α is a real number. After applying a homography transformation $\mathbf{H}$, the points $\mathbf{Hp} + \mathbf{Hq}\alpha$ can still define a straight line.

This implies the following corollary:

Corollary: A homography transformation applied to a polygon will result in a polygon.

Proof: Let $x_1, \ldots, x_n$ be the vertices of a polygon P and let $\bar{x}_{i,i+1}$ be the line segment connecting x_i to x_{i+1}. As seen previously, applying H to P will result in a new set of line segments $\bar{x}'_{i,i+1}$. Due to the linear nature of the homography

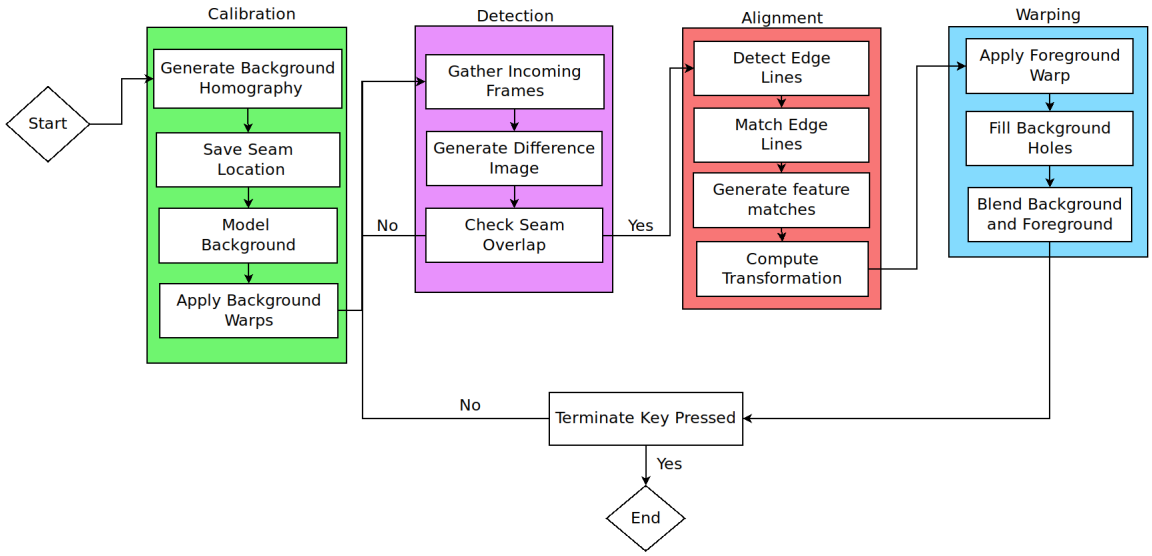


Figure 4.2: MPPM Pipeline

transformation, each vertex will get mapped to exactly one new point x'_i, therefore the end point of the line $\bar{x}'_{i,i+1}$ will be the same as the starting point of the line $\bar{x}'_{i+1,i+2}$ and thus we have a closed polygon.

Hence, without explicitly estimating the homography matrix of the object, one may simply extend the pair of parallel straight lines defining the foreground image of the surgical instrument in the main view to the rectified image of the side view.

With this method, the algorithm requires very low computational complexity while retaining spatial validity wherever there is enough information to do so. Leveraging prior knowledge of the geometry of the remote grasper can simplify computation and make the method more robust to the noise of feature tracking.

4.3.1 Parallax Mitigation

In the preliminary design of the TCA, one camera is designated as the main (reference) view. The other cameras in the array are designated the side view cameras and provide supplementary information to enlarge the FoV of the stitched panoramic video frame. As shown in Fig. 4.2, the algorithm contains four major phases. The

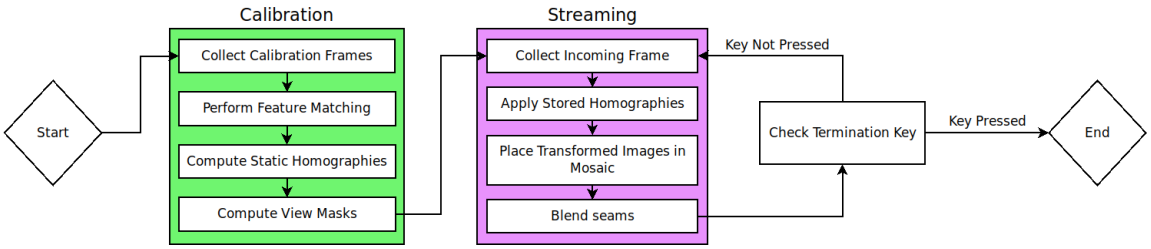


Figure 4.3: Static Homography Alignment Pipeline

initial calibration phase serves to compute static homographies, detect image seams, and generate a background model for each frame. After completing calibration, we enter the main streaming loop. The detection phase involves detecting when a foreground object has entered the frame and needs to be corrected for parallax, the alignment phase computes the appropriate transformation to align the foreground object across the image seam, and the warping phase applies the computed transformation to the foreground object to generate the final panorama.

The quality of the MPPM pipeline was compared to a static homography based real-time video stitching pipeline shown in Fig 4.3. This pipeline was designed based on the work of Zheng et al. [42] to provide fast and reliable video stitching without any form of parallax correction.

4.3.2 Calibration Phase

An initial camera calibration phase is performed to estimate (a) intrinsic and extrinsic parameters of each camera with respect to the reference camera (main view); (b) a static homography matrix between each side view and the main view; (c) a panoramic stationary background image without the presence of any moving surgical instruments. During these steps, seam lines between each side view and the boundary of the main view will also be recorded in both the coordinates of the stitched background image as well as those of each corresponding side view. Assuming that the background has a relatively simple texture and background color, this assumption can be generalized to try and fill in any blind spots the camera array may have. This can be done by simply taking the average intensity value from

the background model and setting any pixel in the panorama which is not set by one of the camera frames to this average color. This calibration phase is performed before video streaming starts, so it can take a bit more time to produce accurate estimates.

Leveraging the application-specific assumption of a stationary background, a moving foreground object can be detected by subtracting the pixel values of the current frame by those of the reference background. Then a threshold is applied to convert the results into a binary image.

To check whether the detected foreground object may cause parallax anomalies, the intersection between the detected foreground object mask and the saved seam lines between a side view and the main view will be examined. If the result is an empty set - which means that the object does not cross the seam line - the stitching operation will proceed using the static homography matrix derived during the initiation phase. If the detected object region overlaps with the seam line along the view boundary, the alignment phase will be executed to mitigate potential parallax anomalies.

4.3.3 Aligning Lines in 3D space

Consider the following two questions: (a) Given the projected images of a straight line in 3D space taken by two or more calibrated cameras, what is the algebraic description of the 3D line? (b) What is the algebraic description of the projected images of a straight line in 3D space from a given viewpoint?

Denote P_1 and P_2 to be two distinct points on a straight-line $L \in \mathbb{R}^3$. Let O_i be the optical center of the i^{th} view (camera). Let O_iP_1 and O_iP_2 be rays from O_i to P_1 and P_2 respectively. These rays intersect the image plane of the i^{th} view at Q_{i1} and Q_{i2}. Hence, the projection of L onto the image plane of the i^{th} view is a line passing through Q_{i1} and Q_{i2}. Furthermore, P_1, P_2, Q_{i1}, Q_{i2}, and O_i shall all be co-planar on a plane denoted by Π_i. The normal vector of Π_i can be found as

$$\mathbf{n}_i = \overrightarrow{O_iP_1} \times \overrightarrow{O_iP_2} = \overrightarrow{O_iQ_{i1}} \times \overrightarrow{O_iQ_{i2}} \tag{4.1}$$

where $\times$ is the vector cross product operator. Note that $\mathbf{n}_i$ is perpendicular to L and the line $Q_{i1}Q_{i2}$, the projection of L on the i^{th} view. That is,

$$\langle \mathbf{n}_i, \overrightarrow{P_1P_2} \rangle = \langle \mathbf{n}_i, \overrightarrow{Q_{i1}Q_{i2}} \rangle = 0 \tag{4.2}$$

where $< \mathbf{x}, \mathbf{y} >= \mathbf{x}^T\mathbf{y}$ is the inner product of two vectors. Π_i may be described as

$$\Pi_i : \{\mathbf{x}; \mathbf{n}_i^T\mathbf{x} + c_i = 0, \mathbf{x} \in \mathbb{R}^3\} \tag{4.3}$$

If the image of L appears in N views ($1 \leqslant i \leqslant N$), then any point $\mathbf{x} \in L$ implies $\mathbf{x} \in \Pi_i$, $1 \leqslant i \leqslant N$. In other words,

$$\begin{bmatrix} \mathbf{n}_1 & \mathbf{n}_2 & \cdots & \mathbf{n}_N \end{bmatrix}^T \mathbf{x} + \begin{bmatrix} c_1 & c_2 & \cdots & c_N \end{bmatrix}^T = \mathbf{0}_{N\times 1}$$

or

$$\begin{bmatrix} \mathbf{n}_1^T & c_1 \\ \cdots & \vdots \\ \mathbf{n}_N^T & c_N \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{0}_{N\times 1} \tag{4.4}$$

where $\mathbf{A}$ is an $N \times 4$ matrix, and the i^{th} row of the $\mathbf{A}$ matrix is $\left[\mathbf{n}_i^T, c_i\right]$ Since a line in $\mathbb{R}^3$ is uniquely defined by the intersection of two planes, $rank(\mathbf{A}) = 2$. Let the singular value decomposition (SVD) of the $\mathbf{A}$ matrix be

$$\mathbf{A} = \sum_{k=1}^{4} \sigma_k \mathbf{u}_k \mathbf{v}_k^T \tag{4.5}$$

where σ_i, $\mathbf{u}_i$, and $\mathbf{v}_i$ are the i^{th} singular value, left singular vector and right singular vector of $\mathbf{A}$. Now pre-multiplying $\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix}^T$ to both sides of eq. (4.4), one has

$$\begin{bmatrix} \sigma_1\mathbf{v}_1^T \\ \sigma_2\mathbf{v}_2^T \end{bmatrix}_{2\times 4} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}_{4\times 1} = \begin{bmatrix} \mathbf{n}_a^T & c_a \\ \mathbf{n}_b^T & c_b \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.6}$$

Eq. (4.6) gives the description of L as the intersection of two planes. Here $\mathbf{n}_a$ and $\mathbf{n}_b$ are normal vectors of the two intersecting planes. Note that although $\mathbf{v}_1^T\mathbf{v}_2 = 0$,

it does not imply that $\mathbf{n}_a^T \mathbf{n}_b = 0$. The direction of L can be found as

$$\mathbf{n}_L = \mathbf{n}_a \times \mathbf{n}_b \tag{4.7}$$

which is perpendicular to $\mathbf{n}_a$ and $\mathbf{n}_b$.

Let O_r and $\mathbf{n}_r$ be the optical center and the direction of the optical axis of a new reference view. The image plane Π_r then can be expressed as

$$\mathbf{n}_r^T \mathbf{x} + c_r = 0 \tag{4.8}$$

If L is not parallel to $\mathbf{n}_r$, one may find a plane Π_L that is perpendicular to L and passes through the optical center of the image plane at the reference viewpoint O_r. Denote the 3D coordinate of O_r by $\mathbf{t}_r = O_r$, then Π_L can be described by

$$\mathbf{n}_L^T \mathbf{x} - \mathbf{n}_L^T \mathbf{t}_r = \mathbf{n}_L^T \mathbf{x} + c_L = 0 \tag{4.9}$$

where $c_L = -\mathbf{n}_L^T \mathbf{t}_r$. Denote $\mathbf{x}_L$ to be the 3D coordinate of the intersection of L and Π_L. It is the solution of the linear system of equations

$$\begin{bmatrix} \mathbf{n}_a & \mathbf{n}_b & \mathbf{n}_L \end{bmatrix}^T \mathbf{x} = -\begin{bmatrix} c_a & c_b & c_L \end{bmatrix}^T \tag{4.10}$$

If L is parallel to $\mathbf{n}_r$, then $\mathbf{x}_L$ will be the projection of L on the reference image view. Otherwise, one may find the projection of $\mathbf{x}_L$ on the reference image plane Π_r, denoted by $\mathbf{x}_r$ using the pin-hole camera equation. Given $\mathbf{x}_r$, the projected image of line L on Π_r can be found as

$$\mathbf{x}_r + \alpha \cdot (\mathbf{I} - \mathbf{n}_r \mathbf{n}_r^T) \cdot \mathbf{u}_L \quad \alpha \in \mathbb{R} \tag{4.11}$$

The line alignment algorithm is summarized in Algorithm 1.

Algorithm 1 Line Alignment

procedure STITCH IMAGES(Main View Video M,Side View Video S)
  for viewpoint $i = 1, 2, /ldots, N$ **do**
    Detect lines in image S_i
    Match lines with image M
    Identify Line of interest L_i
    Compute n_i from L_i, L_M using eq. (4.1)
  end for
  Form the matrix **A** in eq. (4.4).
  Apply SVD to compute σ_1 and σ_2.
  Compute $\mathbf{n}_a$, $\mathbf{n}_b$, and c_a, c_b according to eq. (4.6).
  Compute $\mathbf{n}_L$ and c_L according to eq. (4.9).
  Compute $\mathbf{x}_r$ by solving $\mathbf{x}_L$ from eq. (4.10)
  Find the projected image of L on Π_r from eq. (4.11)
end procedure

Special Case

If the desired viewpoint coincides with one of the given views, one has $r = i$ for some $1 \leqslant i \leqslant N$. Then above procedure can be saved since the projected line image is already obtained at the i^{th} view image.

4.3.4 Alignment Phase

When the moving foreground object crosses the boundary of the main view, part of it will be visible from an adjacent side view. In the current surgical application, the surfaces of this moving object lie on a different plane from that of the background. Using the viewpoint of the main view as a reference, the image of the moving object in the main view shall remain unchanged after stitching. However, the portion of its image in the side view will need to be transformed using a different homography matrix. This would traditionally require applying feature detection inside the object's image in the overlapped region between the main view and the side view. Then a homography may be estimated. However, leveraging the shape of the object in this application, this work develops a novel, fast stitching method.

Figure 4.4: Parallax anomaly manifests as discontinuity of the object image along the view boundary

In laparoscopic surgery, all surgical instruments have a long, cylindrical shape to be inserted into the patient's abdomen through the narrow trocar assembly. Therefore, as shown in Fig. 4.4, parallax anomalies can be detected as discontinuities of the object boundaries across the view boundary seam lines. Moreover, since the object boundary is dominated by the pair of parallel straight lines, we may correct the parallax anomalies without finding the corresponding homography matrix. Instead, we can leverage Lemma 1 to use any homography which correctly matches the object boundary lines from the side view to the main view.

In MPPM, we first detect the dominant parallel straight-line edges of the moving foreground object in the main view as well as in the side view. Then these pairs of features (straight lines) are matched between the main view and the side view. The intersections of the view boundary seam line and these matched object edges can thus be computed. After the homography transformation, the object image in the side view will be bounded by the pair of rectified parallel edges and the view boundary seam line. Therefore, the stitched object image in the side view can easily be obtained by warping the side view object image into the new coordinates. By

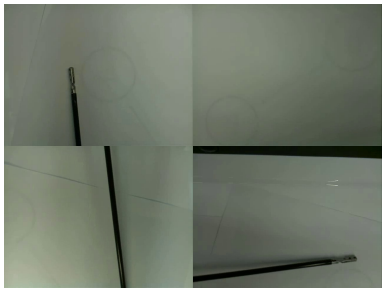


Figure 4.5: Four video frames consisting of a single surgical grasping instrument against the plain white background of a surgical trainer box.

doing so, the efforts of feature extraction and homography estimation can be saved.

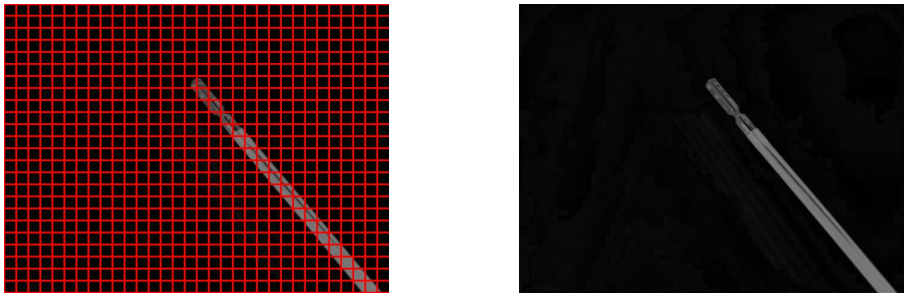
4.4 Results

4.4.1 A Validation Example

Four cameras were used to capture a video feed consisting of a calibration pattern for the first several frames.

A textured pattern is added to the scene to generate an initial set of background homographies. Then the pattern is removed revealing the plain white background of a laparoscopic trainer box. A moving surgical grasping instrument (the foreground object) is inserted into the scene and its image extends across several scene boundaries. An example of video frames from four cameras (views) are shown in Fig. 4.5. When a traditional image stitching algorithm ([42]) is applied to stitch these four views using camera 0 as a reference view, the image of the surgical tool exhibits broken edges across the scene boundary as shown in Fig. 4.1. In this implementation, the overlapped region of the main view (reference view) covers that of a side view.

One solution to this parallax anomaly is to model the surface of the foreground object as a separate plane from that of the background. The long, straight edges are used to infer the orientation of the object plane. Image stitching is performed on the background and the plane separately and combined to make the parallax-free



(a) Scanlines (b) Original

Figure 4.6: The red lines in (a) denote the scan lines used for edge detection. Each line will be treated as a one dimensional slice of the image which will be checked for object location. (b) denotes the original difference image that will be sliced by the scanlines.

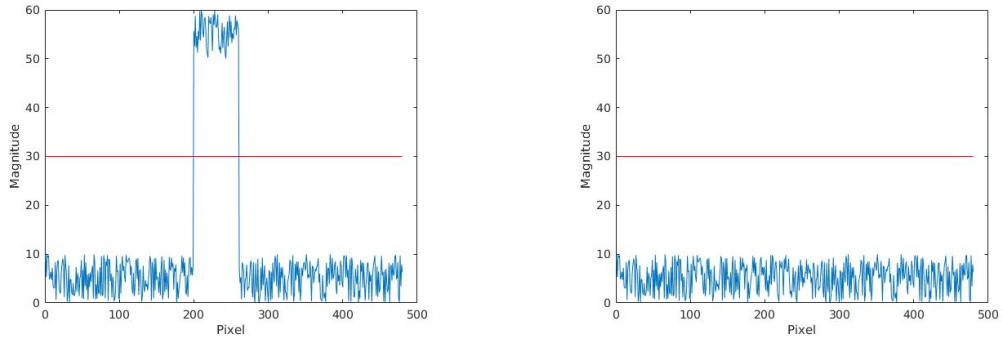
stitched image. Note that in this example, very few point-wise feature points are available for depth estimation using stereo matching.

For each camera, the foreground object is detected using background subtraction. A background model is computed initially by averaging several frames in the absence of the moving foreground object. When the object moves into a view, it is detected as the difference between the current video frame and the background.

$$F_{obj} = I(|F_{fg}(x, y) - F_{bg}(x, y)| - \tau) \tag{4.12}$$

where $I(x)$ is the indicator function and $\tau \in \mathbb{R}$ is a noise threshold.

Once the region of the foreground object is detected in each view, its boundary will be determined. Due to the lack of the point-wise features, we focus on tracking the movement of the foreground object near the scene boundaries where the parallax anomaly is visible. Due to the regular boundary of the object (straight edges), its movement along the scene boundary is tracked using a scanline as seen in Fig. 4.6 and Fig. 4.7. Along each scanline, the crossing points where the difference image crossed the threshold τ are detected. The vertical scanlines thus give us candidate points for the top and bottom edge of the object. Likewise, the horizontal scanlines will give us candidate points for the left and right edges. Applying a linear fitting



(a) Object Detected (b) Object Not Detected

Figure 4.7: Each scanline can be displayed as a slice of the image. In each graph, the blue represents the pixel values of a slice of a difference image, and the red line represents a thresholding function used to determine which pixels contain the foreground object. (a) shows a slice that contains an object and (b) shows a slice that does not.

(a) Single Homography (b) MPPM

Figure 4.8: Applying the proof of concept implementation of the MPPM algorithm to the same scene as Fig. 4.1, we are now able to identify the dominant line edges that define the foreground object and compute a transformation that will align that object without changing the background.

algorithm gives us four candidate lines for the two objects. The zero crossings are utilized to determine correspondence between the candidate lines offered by the vertical scanlines and the candidate lines offered by the horizontal scanlines. Combining the candidates allows for the identification of the final edge lines.

In this example, the reference view is used as the desired viewpoint for the

stitched image. As discussed in section 4.3.3, the line image of the object boundary in the stitched image shall be the same as that in the main view. Hence, the task is to align the line image of the side view to that of the main view.

To align the two detected lines across adjacent views, correspondence between these line pairs must be established. Furthermore, to ensure that the line is extended all the way to the image edge, we compute a mapping that aligns the endpoints of the line in the secondary view to two points in the reference view. The first point is mapped to the edge of the original image in the reference view, then the second is mapped to the intersection of the line and the edge of the reference view mosaic.

The results of this algorithm on a simple scene can be seen in Fig. 4.8. The portions of the object which had broken across seam lines in the static homography alignment (Fig. 4.1) have now been correctly aligned such that the object appears unbroken. The alignment is performed successfully even when the object crosses into and out of 3 viewpoints.

All computation was performed on an Intel Core i7-5500U processor without any multithreading enabled for parallel portions of the processing. Seeking to compare the traditional image stitching approach to the proposed algorithm, both the quality of the image and the time to compute were compared. The static homography method took an average of 94 ms to complete, which resulted in a framerate of 10 FPS. The proposed algorithm took an average of 61 ms, resulting in a framerate of 16 FPS.

CHAPTER 5
Stitching Complex Scenes

5.1 Overview

In Chapter 4, we developed a multi-planar parallax mitigation approach by aligning the boundaries of a moving foreground object in successive frames. The underlying assumption is that the moving foreground object has a planar, feature-less surface with a slender shape and long, parallel straight edges. The parallax anomaly manifests itself as broken edges across the seam-line between adjacent views. By segmenting the foreground object at the side view and applying a homography derived from the corresponding line segments in the main view and the side view, this parallax anomaly can be eliminated in the stitched video frame.

In this chapter, we consider scenarios of complex dynamic scenes with multiple moving foreground objects. In a complex dynamic scene, there may be two or more moving foreground objects of different shapes. These objects may have varying depths observed from a given viewpoint and may appear across the seam-line between the main view and a side view. As such, the ad hoc stitching approach developed in Chapter 4 needs to be generalized to provide multi-view, multi-planar video stitching of complex dynamic scenes.

The system setting is similar to that described in Chapter 4: two or more video cameras mounted on a rigid frame capturing multiple synchronous video streams. The cameras are placed to maximize the overall field of view after video stitching. Sufficient overlap between adjacent views will be maintained to establish feature correspondence between these views. Choosing one of the camera's viewpoints as the viewpoint used for stitching (the main view), our objective is to stitch available side views to maximize the field of view of the stitched video frame while

minimizing parallax anomalies in the overlapped region. The types of parallax anomalies include discontinuities of visible edges across the overlapped region, ghost images of objects inside the overlapped region, and distortion or blurring of images due to parallax mitigating procedures. Our approach is to model the stitched scene with multiple planar surfaces. For each view, foreground objects will be separated from the background. If a foreground object's image has no intersection with any overlapped region, it will be treated as part of the background because it cannot cause any parallax anomaly. It is further assumed that the background scene and background objects lie on a background plane. Once the relative camera pose of a side view camera with respect to that of the main view camera is estimated, a projective transformation will be performed to align the background of a side-view with respect to that of the main view.

If an object is designated as a foreground object, its image must intersect with an overlapped region between adjacent views. For each foreground object, we assume its visible surface is planar. We will develop a separate projective transformation for each foreground object, implicitly modeling the visible surface of it as a plane. In other words, we adopt a 3D model with multiple planes to approximate the surface of the complex scene based on the overlapped regions between the main view and each side view.

5.2 Multi-View, Multi-Planar Stitching

As previously discussed in Chapter 3, planar image stitching as originally proposed by Szeliski et. al. ([28]) involves modeling the scene as a single plane. However, in reality, such scenes are rarely planar. Gao et al. ([4]) identified two major planes dominating landscape scenes, the ground and the horizon. While this meant that these scenes could not be modeled well using the traditional method, it could instead be split up into two separate images, one corresponding to each plane. Each of these images could then use the traditional method of image stitching to generate an image mosaic.

It may then make sense to consider a more general model for image stitching where the scene is broken up into piece-wise planar sections which can then be used to generate components of the final mosaic. In the general case where the desired number of planes is unknown, this approach would become equivalent to 3D modeling and rendering, with each feature match contributing to a polygonal model of the scene. However, prior knowledge of our recorded scene can allow us to simplify down to a set number of planes. If the scene is planar, then this becomes Szeliski's original method and with two dominant planes (located at the top and bottom of an image), we have Gao's method.

The piecewise planar scene model has been used in other areas of computer vision. Many different papers have used the piecewise planar model as a basis for the purpose of refining or improving 3D reconstruction algorithms [67, 68, 69, 70] . Fraundorfer et. al. used the piecewise planar assumption to allow for the reconstruction of a 3D scene from sparse feature points [71]. Sinha et. al. used the model to perform image-based rendering, by constructing a piecewise planar representation of the scene in 3D space and then reprojecting that scene back to novel viewpoints [72].

However, 3D information either relies on specialized hardware or is only available in the overlap area between views. There is no depth information that can be captured outside of the overlap region. The image stitching paradigm automatically encodes the assumption that all pixels which share a homography will be co-planar. This assumption allows the extrapolation of the aligning transformation to pixels that do not fall in the overlap region.

In considering the validity of the multi-planar model for image stitching, let us consider a scene made of disjoint planes Π_i for $i = 1, 2, \ldots, n$. Let I_s and I_d be the source and destination images respectively. Then our scene $S = \cup_i \Pi_i$. Therefore, any point $p \in S$, p must belong to at least one plane Π_i and so there exists some normal N and distance d such that $Np = d$. If the source and destination cameras are related by a rotation matrix R and a translation T, then we can find a homography $H = R + \frac{1}{d}TN^T$ which aligns the point from the coordinate system of I_S to the coordinate system of I_d. Therefore, there must exist some homography H_i such

that $x_d = \lambda H x_s$.Thus we can see that for each pixel in the mosaic image, the value of the pixel must belong to the corresponding pixel in the transformed image created by applying H_i to I_s.

The general multi-planar model of image stitching can be outlined in the following way. For a given pair of images, we let I_s be the source image and I_d be the destination image. The destination image serves as the coordinate system to which the source image will be aligned. We define the planes that exist in the shared regions between the images as $\{P_i : i = 1, 2, \ldots, n\}$, Each plane P_i denotes the pixels in each image that contain objects that lie on that plane. Every pixel should belong to a single one of these planes such that we could generate a pair of masks $M_s(x, y) = i : I_s(x, y) \in P_i$ and $M_d(x, y) = i : I_d(x, y) \in P_i$. These masks allow us to split each image into a set of images (I_{xi}, I_{di}) with pixels pulled only from a single plane.

$$I_{si}(x, y) = \begin{cases} I_s(x, y) & \text{if } M_s(x, y) = i \\ 0 & \text{if else} \end{cases} \tag{5.1}$$

$$I_{di}(x, y) = \begin{cases} I_d(x, y) & \text{if } M_d(x, y) = i \\ 0 & \text{if else} \end{cases} \tag{5.2}$$

Each pair of source and destination images can be stitched without parallax using the traditional planar method ([1]) to generate a slice of the final mosaic. However, when compositing these slices, the parallax effect may have created gaps or overlap, so occlusion handling and gap-filling must be taken into account when generating the final panorama.

The previous example in the surgical training box showcased how this method could be used to create spatially valid mosaics. However, the simplicity of the example scene leaves a lot of questions about how well the method could translate to real-world scenes. The background subtraction and thresholding methods used in our previous example are not robust to minor camera movements nor do they allow for the segmentation of more than a single foreground object.

Algorithm 2 Multi-Planar Stitching

1: **procedure** MULTI-PLANAR STITCHING(Main View Frame F_m,Side View Frames F_s)
2: $k_m = \text{DetectFeatures}(F_m)$
3: **for** $I_j \in F_s$ **do**
4: $k_j = \text{DetectFeatures}(I_j)$
5: $M = \text{MatchFeatures}(k_m, k_j)$
6: $S = \text{SegmentImage}(I_j)$
7: $\hat{M} = \text{SegmentFeatures}(M, k_m, k_j)$
8: **for** $S_i \in S$ **do**
9: $H_i = \text{ComputeH}(\hat{M}_i)$
10: $T_{ij} = \text{ApplyHomography}(S_i, H_i)$
11: **end for**
12: **end for**
13: $F_t = \text{CompositeImages}(F_m, T)$
14: **end procedure**

To address these challenges, a new multi-planar stitching algorithm was developed using object segmentation to apply multi-planar stitching using a local warping method where each pixel is assigned a homography based on segmentation results (Alg. 2).

In the case that more than one side view is available, it is assumed that each of the side views overlaps with the main view. This allows each pair of main and side views to be treated as independent from one another, allowing lines 4-10 of Alg. 2 to be performed simultaneously for each pair of images.

5.2.1 Feature Detection and Matching

In line 2 and 4 of Algorithm 2, each image is subjected to a feature detection method. SURF was chosen for feature detection due to its performanc in several sample scenes [14]. However, there are many feature detection algorithms that may be used in place of SURF. Different feature detection methods may offer different benefits and drawbacks and may be applicable to different scenes or use cases[73], but they all share a similar purpose. These methods are designed to detect easily identifiable

regions that can be used to compute a sparse point-wise correspondence between the images.

The feature detection algorithms provide us with a list of key points and corresponding feature descriptors for each image. In line 5, we utilize the brute-force L1-norm based matcher provided by OpenCV to compare the descriptors and compute correspondence between the detected features in the main and side view. After performing this feature matching, we have a paired list of key points that provide a pointwise correspondence between images. This correspondence serves as the basis of the alignment problem posed by image stitching.

5.2.2 Scene Segmentation

In line 5 of Algorithm 2, the scene is segmented to facilitate the use of semantic scene information to correct for parallax. There are many different approaches to image segmentation. Super-pixel segmentation methods such as SLIC [74] combine pixels into larger regions based on color and texture similarity. Previous work in image stitching has used SLIC segmentation to adjust spatially varying warps and produce higher quality stitched images[75]. However, super-pixels are not quite large enough to encompass a whole object and thus the segmentation provided by SLIC is too fine-grained and does not truly leverage an understanding of semantic scene information. In a video segmentation setting, we may use optical flow and motion segmentation [76, 77] to identify foreground objects and to track objects which can be identified by regions of shared motion.

The current state of the art in image segmentation are deep-learning based approaches[78, 79]. However, these approaches are data-driven and require large quantities of annotated images in order to perform well.

In many cases, the segmentation algorithm will break planar objects into multiple segments. This subdivision will increase the number of objects that do not fall within the overlap region and can introduce new discontinuities along the seams between segments. As it is only possible to align segments that contain features in both the main and side views, any segments which do not meet this criteria

must be removed. To do this, segments can be combined until each can be stitched independently of all other segments. A naive approach to combining segments would be to only combine segments with the background. Without information about how to align these segments, it makes sense to revert back to the single homography case.

Any error in segmentation could propagate through the rest of our algorithm to create new stitching artifacts. Each homography will preserve the straightness of lines, but those lines will not be preserved from segment to segment. As such it may be desirable to refine the segmentation results using multiple segmentation approaches.

As mentioned previously, foreground objects can be identified and tracked via shared motion parameters using motion segmentation [76]. Similarly, motion information can be utilized to refine the segmentation results. Segments that share motion from frame to frame can be grouped into a single object.

Likewise, a calibrated camera array allows an easy method to correct any incorrect segments. If the camera array is calibrated, structure from motion [47, 80, 81] can be utilized to collect a 3D point cloud of features in the overlap region between cameras. When combined with a basic segmentation method, this allows 3D points to be linked with image regions. The 3D point cloud can in turn be refined into a polygon mesh [82, 83, 84, 85]. Each of these polygons can be clustered into co-planar regions. Finally, the segmented regions which contribute to the same co-planar region can be grouped into the same object mask, as they will share the same homography.

The output of our segmentation method should provide a segmentation mask S where

$$S(x,y) = \begin{cases} i & \text{if } I(x,y) \text{is part of object } i \\ 0 & \text{if else} \end{cases} \tag{5.3}$$

Under the assumption that each object can be approximated with a single plane, the planar mask in eq 5.1 can be replaced with the segmentation mask S. This modification of equation 5.1 allows the segmentation of the side view image into a

set of object images S_i where

$$S_i(x,y) = \begin{cases} I(x,y) & \text{if } S(x,y) = i \\ 0 & \text{if else} \end{cases} \tag{5.4}$$

In line 6 of Algorithm 2, the set of feature matches is partitioned based on the results of the segmentation. N sets of feature matches $\hat{M}_i$ for $i = 1, 2, \ldots, N$ are generated where N is the number of segments in the segmentation mask. Defining a set of points $s_i = \{(x,y) \in \mathbb{R}^2 \text{s.t.} S_i(x,y) > 0\}$, the segment feature matches can be defined as the intersection of the feature matches and s_i, that is to say $\hat{M}_i = s_i \cap M$.

5.2.3 Stitching with Multiple Homographies

Four matched points are required to compute a homography[64]. However, more feature points allow for the use of RANSAC [15] to significantly increase the robustness to noise in the feature set. In order to fit a homography for our camera system that correctly aligns all points, the set of points must all lie on a single plane. As shown in Section 5.3.1, it is more accurate to fit a plane to each object for scenes in the MICCAI dataset than to fit a single plane to the entire scene. So long as $\hat{M}_i$ contains at least four feature matches, we will be able to compute an aligning homography for S_i.

In steps 7 and 8 of Algorithm 2, the segmented feature matches $\hat{M}_i$ and object images S_i are used to generate a set of transformed object images T. For each object i, the four-point algorithm is used with RANSAC to compute the optimal aligning homography H_i. H_i is then used to transform S_i using inverse warping. The shape of T_{ij} are computed by applying

$$\lambda \begin{bmatrix} c'_{xk} \\ c'_{yk} \\ 1 \end{bmatrix} = H \begin{bmatrix} c_{xk} \\ c_{yk} \\ 1 \end{bmatrix} \tag{5.5}$$

to each corner (i.e. $k = 0, 1, 2, 3$) of S_i. We define the bounds on T_i to be

$$x_{min} = \min_k c'_{xk}, \quad x_{max} = \max_k c'_{xk}$$

$$y_{min} = \min_k c'_{yk}, \quad y_{max} = \max_k c'_{yk}$$

Then, for $x_{min} \leqslant x' \leqslant x_{max}$ and $y_{min} \leqslant y' \leqslant y_{max}$ we define the corresponding coordinates (x, y) to be

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{5.6}$$

Then we define the value of each pixel in T_i to be

$$T_i(x', y') = \begin{cases} S(x, y) & \text{if } (x, y) \in S \\ 0 & \text{if else} \end{cases} \tag{5.7}$$

By using inverse warping rather than forward warping, we ensure that each pixel is assigned a value and prevent holes that may occur when adjacent pixels are warped to non-adjacent locations.

5.2.4 Compositing

In line 11 of Algorithm 2, we use compositing to address how to select which pixels contribute to the final mosaic and how these pixels should be blended together. Compositing can have a major effect on the appearance of the final mosaic and a proper choice of compositing approach can hide misalignment or create new artifacts.Because of this, and because of the different computational requirements for compositing approaches, the choice of compositing may depend on the desired use of the mosaic.

To easily compare the quality of image stitching results, it is desirable to select a compositing method that will highlight misalignment without introducing new

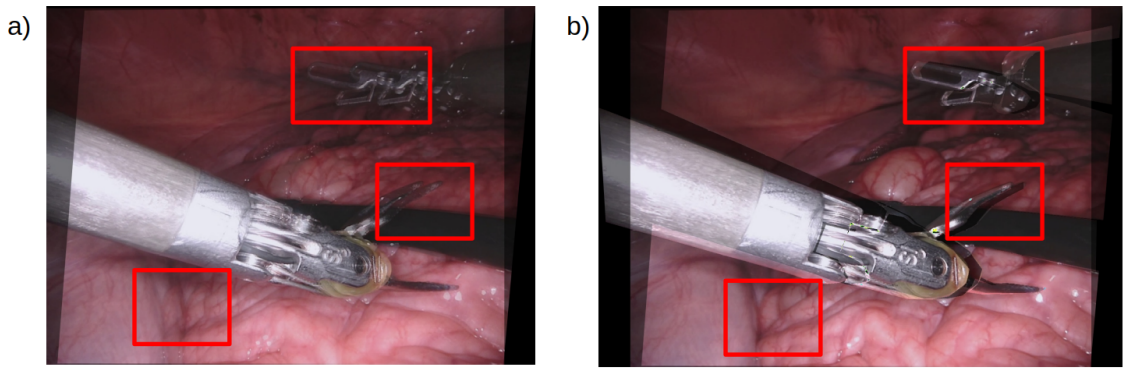


Figure 5.1: An example of a stitched mosaic using (a) traditional single planar stitching and (b) Multi-Planar stitching. The multi-planar stitching has corrected the misalignment of the grasping tools and increased accuracy of the background alignment.

artifacts due to intensity mismatches or other noise introduced by cameras or lighting. However, for a typical end-use case, misalignment should be hidden as much as possible. The goal in these instances is to try to match the composite image as closely as possible to the result that would be achieved by a single angle camera.

For the purpose of this work, we chose a simple pixel weighting approach. For each pixel in the composite, the value of that pixel is given by

$$C(x,y) = \frac{1}{N}\sum_{k} I_k(x,y) \tag{5.8}$$

where $I_k(x,y)$ represents the value of that pixel in image k and N is the number of images that have $I_k(x,y) > 0$. This compositing method is not ideal for generating high-quality mosaics. It results in large brightness discrepancies between regions containing multiple segments and regions containing only the main view. However, this method makes misalignment clear as seen in Figure 5.1

An alternative approach to compositing that could be applied to better disguise misalignment when it does occur is to instead use a seam selection approach that places the seam-line between images based on object position. In this approach,

$$C(x) = I_{\hat{k}}(x, y)$$

where $\hat{k}$ is chosen by the formula:

$$\hat{k} = \max g(I_k(x, y)$$

and

$$g(I_k(x, y)) = \begin{cases} k & \text{if } I_K(x, y) = 0 \\ 0 & \text{if else} \end{cases}$$

By taking each pixel only from a single I_k, we guarantee that the only places where misalignment will be visible is across seamlines as any misalignment that occurs elsewhere will be erased by the compositing.

In the single object case, the seam selection approach is simple to implement and apply. If there is a foreground candidate for a given pixel, then we assign that value to the pixel. Otherwise, we assign the background candidate. In the multi-object case, there is ambiguity as to which object should be given priority when occlusion occurs between foreground objects.

Optimal compositing is strongly dependent on the use case of the mosaic. Blending the images together can make misalignments very clear while still creating high quality mosaics when misalignments do not occur. This allows for an excellent comparison of images with high amounts of overlap like the MICCAI 2017 dataset. However, highlighting misalignments often leads to suboptimal final mosaics. The optimal blending algorithm would mask any misalignments that occur away from the seam between main and side view as well as correcting for intensity variance across the seam to ensure that the final mosaic looks as much like the view from a single camera as possible. In video settings, the blending algorithm will also need to be computationally efficient enough to keep up with the required frame rate of the use case.

5.3 Application to Surgical Scenes

We apply Algorithm 2 to the MICCAI 2017 Robotic Instrument Segmentation Dataset [86]. This dataset consists of 8 videos of porcine surgical operations captured at a sampling rate of 2 Hz. These videos are labeled with ground truth segmentation values for the surgical tools used in the procedure and these ground truth values were used as the segmentation results. As the dataset we had was limited, and the focus of this work was not on the development of better segmentation methods, we opted to use the ground truth values provided by our dataset to showcase the effectiveness of our algorithm.

The porcine procedures captured in this dataset contain several surgical tools that act as foreground objects and a feature-rich background containing non-lambertian surfaces that make it a difficult dataset to stitch correctly.

The frames for these videos featured a high degree of overlap which offers more feature points shared between the two views and more locations where parallax misalignment may occur and be visible. However, the low sampling rate increases the amount of motion between frames, reducing the amount of information provided by prior frames, and thus the effectiveness of tracking algorithms.

5.3.1 Testing Validity of Multi-planar Model

In order to compare the fitting error of the multiplanar and single-planar models, we compare the fit of multiple planes chosen based on the ground truth segmentation results to a single plane fit to the scene's entire 3D geometry.

First, we need to compute point-wise correspondence between the two views. We use SURF [14] and a brute-force matcher using an L1 norm distance metric in order to compute point-wise correspondence. Using these feature matches, we construct a 3D point cloud using triangulation [87].

In order to fit a plane to the point cloud, we utilize an orthogonal distance regression-based plane fitting algorithm [88]. We consider the plane represented by the vector n and point c where n is the unit normal to the plane and c is a point

on the plane. Then we can see that the orthogonal distance between a point p_i and the plane is:

$$d = ((p_i - c)^T n) \tag{5.9}$$

Thus we can define our plane fitting problem as

$$\min_{c,\|n\|=1} \sum_{i=1}^{N} ((p_i - c)^T n)^2 \tag{5.10}$$

where $P = \{p_1, p_2, \ldots, p_N\}$ is the data to which the plane is being fit. Solving for c we can see that

$$c = \frac{1}{N} \sum_{i=1}^{N} p_i \tag{5.11}$$

and the problem can be re-formulated as

$$\hat{n} = \arg\min_{\|n\|=1} \|An\|_2^2 \tag{5.12}$$

by defining the matrix $A = [p_1 - c, p_2 - c, \ldots, p_N - c]$. Thus we can see that the solution $\hat{n} = U(:, 3)$ where $U(:, 3)$ is the third left singular vector of A.

Using this formulation, we can fit two models to the images of the MICCAI dataset. The dataset provides camera calibration information which can be used to triangulate a point cloud for each pair of stereo images [64]. For our baseline, we fit a single plane to the dataset. This measures the amount of parallax present in the single planar model. We compare this against a multi-planar model, where we fit N planes, one for each segment provided by our segmentation mask.

The 3D point cloud is constructed by triangulating 2D feature points to compute their 3D coordinates. Therefore, since each 2d point will have a segmentation label, that label can be propagated to the corresponding 3D point. Partitioning the point cloud based on these segmenting results allows us to fit a separate plane to each partition. Then, the distance d_i from each point to the plane can be computed using

Table 5.1: Comparison of Fitting error

	Single-Planar MSD	Multi-Planar MSD
Video 1	0.00325	0.00064
Video 2	0.0014	0.0000982
Video 3	0.0159	0.00172
Video 4	0.01156	0.0002466
Video 5	0.0209	0.00225
Video 6	0.00989	0.00718
Video 7	0.00094	0.00025
Video 8	0.00149	0.00018

Eq. 5.9. Finally, the Mean Squared Distance can be computed using the equation

$$d_m = \frac{1}{N} \sum_k = 1^N d_i \tag{5.13}$$

The resulting mean squared distance for each model can be seen in Table 5.1. As we can see, the multi-planar model is able to significantly reduce the planar fitting error, and thus the parallax in the scene.

5.3.2 Results

Table 5.2 shows a quantitative comparison of the standard single planar model with our new multi-planar model. The stitching error is measured as the mean squared distance in pixels between the transformed feature points after alignment and their corresponding location in the destination image.

A qualitative comparison of images stitched using As-Projective-As-Possible stitching [3], and multi-planar stitching can be seen in Figure 5.2

5.4 Discussion

These results are very promising. Both quantitatively and qualitatively, they show significant improvement over the existing methods. In reality, the results of this ap-

Table 5.2: Mean Squared Alignment Error (Pixels) on MICCAI 2017 Dataset

	Single-Planar MSE	Multi-Planar MSE
Video 1	14882.33	4519.41
Video 2	11259.84	3324.30
Video 3	6670.21	5025.19
Video 4	13512.44	2479.00
Video 5	7987.765	4315.777
Video 6	11828.349	4623.56
Video 7	17770.06	7830.45
Video 8	10793.67	3992.20

proach are heavily dependent on the accuracy of the chosen segmentation method. By using the ground truth segmentation in our surgical application, the effect of segmentation noise on the algorithm is removed. However, image and video segmentation algorithms may not be entirely noise-free and segmentation algorithms may introduce noise into the system that may reduce mosaic quality.

If segmentation results are inaccurate, then the assumption that the resulting regions are planar may begin to break down. As this algorithm relies on the assumption that each scene is broken into co-planar object regions, then poor segmentation accuracy may result in parallax occurring within the segmented object and reduce the ability for the algorithm to correct parallax. If the chosen segmentation algorithm partitions the images into regions that are too small, then the objects may no longer contain sufficient feature matches to robustly compute accurate homographies, which may result in inaccurate alignment and the introduction of new artifacts. In order for the four-point algorithm with RANSAC to accurately compute an aligning homography, four accurate pointwise correspondences are required and the number of accurate correspondences must outnumber inaccurate correspondences. The line alignment algorithm from Section 4.3.3 can be used to fix or augment feature-poor regions but does require the assumption of prior knowledge about object shape.

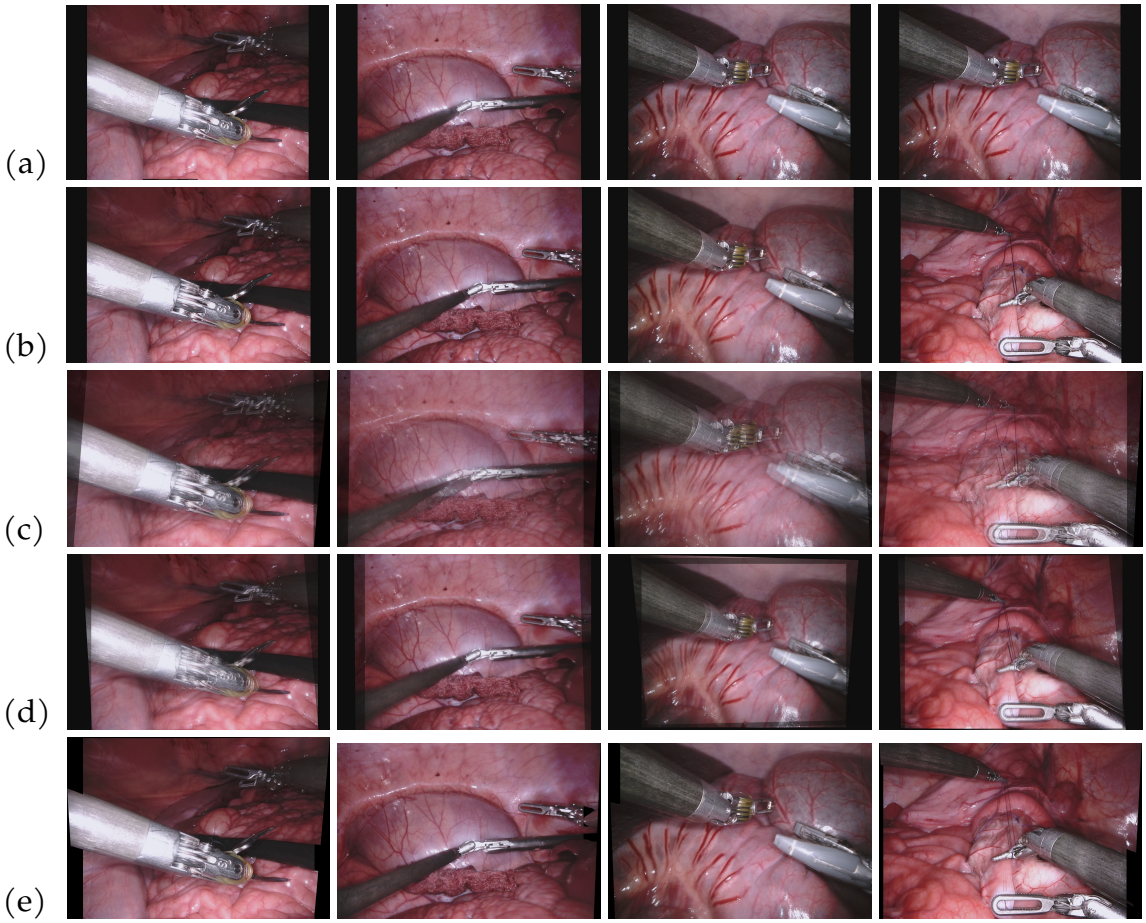


Figure 5.2: A comparison of the (a) left and (b) right frames from the MICCAI 2017 Dataset along with the stitching results from (c) Single planar stitching, (d) As-Projective-As-Possible Stitching, and (e) Multi-planar stitching

5.5 Conclusion

In this chapter, the multi-planar stitching algorithm has been extended from simple scenes to more complex scenes. By segmenting the images into planar objects, the problem of image stitching can be formulated as the union of several disjoint planar stitching problems where each object is assigned a projective transformation and aligned independently of the rest of the scene.

The algorithm was tested on the MICCAI 2017 Robotic Surgical Segmentation dataset. First, it was shown that fitting multiple planes to each object in the dataset could significantly reduce the total amount of parallax over the single planar model used by traditional stitching methods. Comparing the mean distance between aligned points for both single-planar and multi-planar stitching showed that the multi-planar model can be used to reduce parallax artifacts in the overlap region between images.

This work focused on the development of the stitching algorithm and as such, segmentation was treated as a given. As segmentation is still very much an actively studied area, future work in this area could examine the effect that certain segmentation algorithms may have on the quality of the resulting mosaic. Methods for using depth or motion to refine segmentation results and improve their quality, but these methods have not yet been tested.

The low sampling rate of the MICCAI dataset limited the ability to leverage any temporal correlation. Thus the multi-planar stitching algorithm outlined in this chapter functions as an image stitching algorithm but not a video stitching algorithm. Future work could adapt this algorithm to better fit the live video stitching case by reducing computation, tracking temporal information, and ensuring temporal continuity from frame to frame.

CHAPTER 6
Optimal Camera Placement for Maximal Field of View Stitching

6.1 Problem Formulation

An optimization problem is formulated to maximize the FoV of the camera array while ensuring that the chosen array will still create a useable stitched mosaic. To do this, image stitching constraints are combined with optimal camera placement models.

Similar to the traditional camera placement models [21, 22, 23], the camera positions are discretized based on the micro-camera array structure. The choice of camera position constraints will depend on where the camera network is meant to be deployed. For this chapter, the focus is on three different possible sets of constraints intended for use in the surgical camera array which can be described as naive, symmetric, and asymmetric camera placement. A coverage model similar to the one used by Fu et al. [24] is then evaluated. While the Fu model often uses additional constraints [22, 24] such as resolution and focus which limit the camera FoV even further, these bounds are unnecessary in close field applications such as the desired surgical application.

The scene is modeled as a plane in space as in Fu et al. [24]. This follows naturally from the stitching requirement that the recorded scene be approximately planar [1]. The planar scene constraint also precludes any need for occlusion handling as occlusion will not occur in a purely planar scene. Thus, occlusion handling methods [22] are not included in the model. In this setting, a plane which represents our approximation of the image stitching plane is selected. If the

coordinates of the image stitching plane are known exactly, then one can simply utilize the camera projection model to generate the image mosaic. The resulting mosaic is not robust to noise and will exhibit misalignment if the planar scene is not exactly planar. Thus, even with prior knowledge of the scene, it is still necessary to perform traditional image stitching in order to minimize the likelihood of misalignment.

Image stitching requires sufficient overlap between adjacent cameras to find the feature matches used to compute image correspondence. For multi-camera image stitching, each camera must be able to trace a path back to the main-view camera, with each step along the path transitioning between two cameras which share sufficient overlap. To ensure this continuity constraint is obeyed, a graph based method is used to determine if a global image correspondence can be achieved.

Rather than discretize the scene space as in many previous works, it's possible instead to evaluate the area of the continuous coverage region on the stitching plane in order to more accurately evaluate the total coverage. Since each camera's coverage region is the intersection of a rectangular pyramid with the plane of stitching, it is easy to see that each camera will contribute a single quadrilateral to the overall coverage region. The area of the resulting coverage region can be quantified as simply the area of a union of quadrilaterals which makes it simple to compute.

After determining how to calculate the area of the scene which is visible from a given camera configuration, the following optimization is proposed. Given a stitching plane, a number of known cameras N_c which can be placed in a discrete set of poses and locations, and a minimum threshold for overlap between the camera views to allow for image stitching, the cameras should be placed in such a way that:

$$\max_{R_i, \bar{t}_i} A(\cup_i Q_i) \tag{6.1}$$

$$Q_i = [B_{i1}, B_{i2}, B_{i3}, B_{i4}]^T \tag{6.2}$$

$$B_{ij} = \lambda_{ij} R_i V_j d_i + \bar{t}_i \tag{6.3}$$

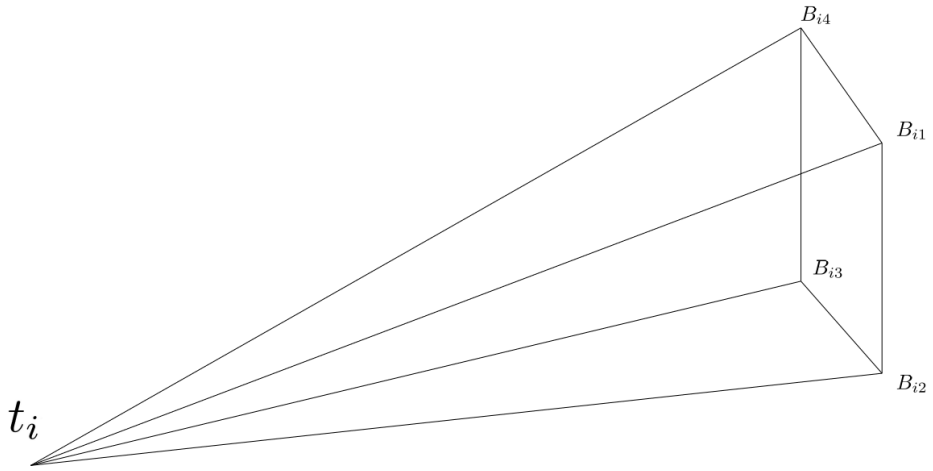


Figure 6.1: The camera viewing cone is defined by its four corner vectors (B_ij) and the camera center (t_i).

$$\lambda_{ij} = \frac{\bar{t}_i[\nu, \tau, \psi]^T + \eta}{R_i V_j d_i [\nu, \tau, \psi]^T} \tag{6.4}$$

$$g(\cup Q_i) = 1 \tag{6.5}$$

where $A(Q)$ is the function for the area of the polygon Q. Q_i denotes the quadrilateral defined by the intersection of camera i's viewing cone with the plane of stitching. Index $i = \{1, 2, \ldots, N_c\}$ refers to which camera we are using, and index $j = \{1, 2, 3, 4\}$ refers to the corners of the viewing cone for that camera.

The viewing cone (as shown in Figure 6.1) is defined by the camera center and four vectors denoting the four corners of the viewable region. These vectors are described in Equation (6.3), where R_i and $\bar{t}_i$ denote the camera i's rotation and translation matrices, d_i contains the information about the maximum viewing angles for camera i, and V_j is a matrix that selects the vector corresponding to the jth corner of the viewing cone for that camera.

The stitching plane coefficients $[\nu, \tau, \psi, \eta]$ define the stitching plane with $\nu x + \tau y + \psi z + \eta = 0$ being the plane of stitching and λ_{ij} determining the length of B_{ij} when projected onto the stitching plane using ray-plane intersection as in Equation (6.4).

Since we are dealing with a discrete set of possible camera poses and locations, $A(Q)$ can be computed prior to run-time and stored so that it can simply be accessed from a look up table when checked for each possible camera set up. $A(Q)$ will scale linearly with the number of allowable camera poses, but not scale up with the total number of array set ups. $g(Q)$ counts the number of simple polygons required to describe the polygon Q. This serves to simplify computation of the area of the final FoV and it ensures that the final FoV does not have any holes. This is relevant to most applications since a hole usually represents an area close to the region of interest for which we do not have data. In surveillance, a hole could be exploited to hide information from the cameras, and, in surgical settings, a hole could cause the surgeon to miss out on information about tissues or organs near their surgical tools. In some settings, where the region of interest is oddly shaped, holes may be allowable, however, in many, it is not. To enforce these two additional constraints, we follow the algorithm outlined in Figure 6.2.

R_i and $\bar{t}_i$ are the rotation and translation matrices for each camera. The optimization is over all possible sets of rotation and translation available to our cameras. The construction of the array itself limits the possible camera poses, and acts as additional constraints on our problem. Without constraining camera pose, the problem is ill-defined. However, the camera pose constraints are heavily dependent on the design of the array itself.

Unfortunately, this method requires checking each possible array set up to determine which solution is optimal if we consider a camera array where each camera can move and rotate freely. This creates six degrees of freedom for the camera (three degrees of rotation, and three degrees of translation). When we discretize the camera positions, if we allow n discrete values along each degree of freedom, this would cause there to be n^6 possible placements per camera. Thus, a k camera array would have approximately n^{6k} possible arrangements. In reality, we can condense the number of arrangements slightly. No two cameras can be placed in the same position though they are allowed to share the same rotational pose. This means that we can have at most $\binom{n^3}{k}$ possible positions rather than n^3k. Thus, the total magnitude of the solution space is actually $\binom{n^3}{k}n^{3k}$

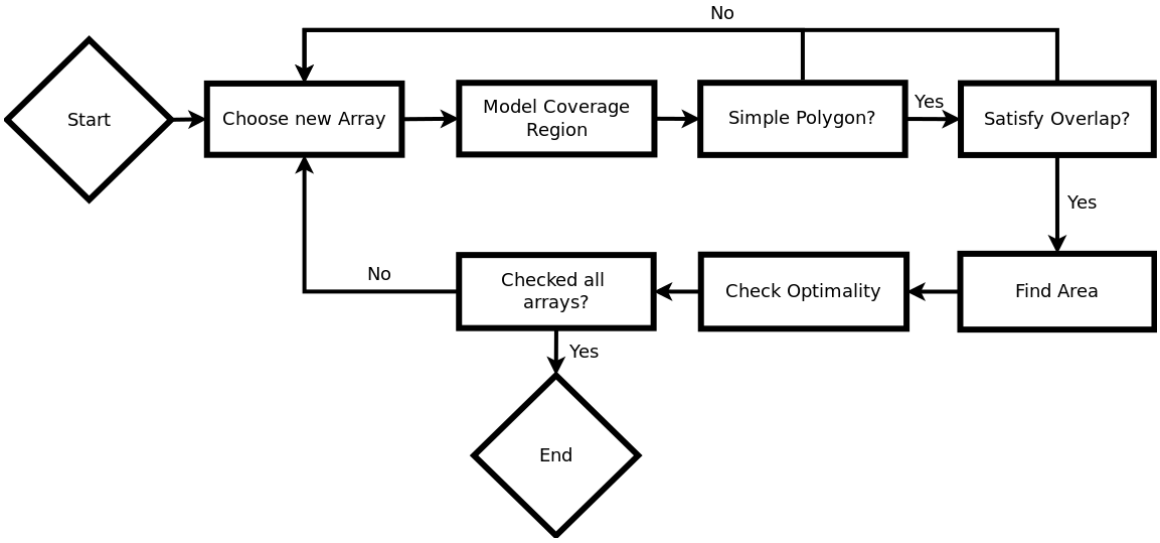


Figure 6.2: The general flow of our optimization algorithm. For each possible array, we model the coverage region, check to see that the result is a simple polygon which satisfies the overlap constraints, then find the area and compare that to the previously maximum area array.

As the number of allowable camera positions grows, solving our optimization quickly becomes computationally infeasible. To speed up computation time, we utilize a greedy heuristic to vastly simplify the computation time. While this heuristic is not guaranteed to find an optimal solution, It still provides good results, as seen in the paper by Zhao et al., and can simplify the exhaustive search algorithm down to polynomial time [23].

6.1.1 Scene Space Model

Most existing works in optimal camera placement utilize a discrete scene space model. In this model, rather than calculate the exact coverage region, a discreet grid of scene points is overlaid over the scene. The optimization function will then seek to maximize the number of these scene points which are covered by the camera array. This model helps to simplify computational constraints of the camera model and objective functions at the cost of some amount of accuracy in the size of the

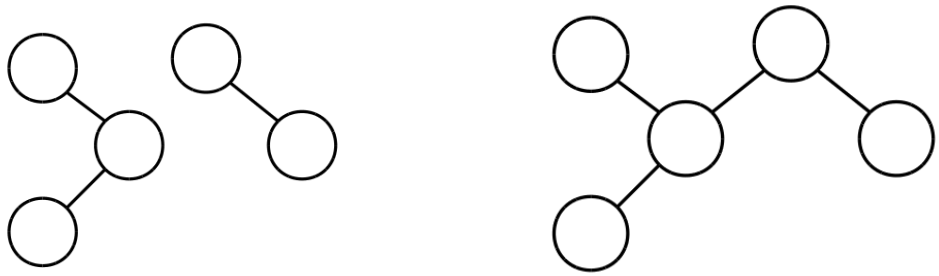
coverage region.

One of the largest benefits of using a discrete scene space model is that it allows the use of binary integer programming techniques that would not otherwise be possible. However, these techniques also require that the system constraints be expressed as a linear function. Since the computation of the individual camera coverage is a nonlinear function, the binary variable b_{ij} which denotes whether a discrete scene point j can be seen from a camera placed at position i must be computed and stored in full before the optimization can be performed. As such, as the camera coverage model becomes easier to evaluate, the the binary optimization model becomes less preferable.

Without occlusion handling required in our setting, computing our individual camera coverage model is simple, although it is nonlinear and our constraints mean that evaluating the coverage region is a simple matter of performing a union of quadrilaterals using polygon clipping techniques [89] and then calculating the area of the resulting simple polygon. Using a continuous scene space allows us greater accuracy for the simple exhaustive search methods we wish to use to evaluate our camera array's coverage quality.

6.1.2 Ensuring Continuity

To allow for stitching, we need to ensure that there is enough overlap between the cameras for feature matching to occur. The general approach to stitching together video from camera arrays is to find pairwise homographies which will transform the images such that features are matched between the resulting images. The amount of overlap required will vary based on the feature density of the scene viewed. However, assuming we know the amount of overlap required between our images for our expected scenes, we can set the following limitations on our set up to ensure that stitching may occur. We define the amount of overlap between two cameras to be the area of the intersection of their fields of view. To form a panorama from the cameras, we need to ensure that we can reasonably determine where each image needs to be in respect to all of the others. To do this, we propose a graph



(**a**) Disconnected Graph (**b**) Connected Graph

Figure 6.3: To determine whether stitching is possible, we need to check that there is sufficient overlap between the views. To do this, we create a graph where each node represents a camera and the existence of an edge specifies that there is enough overlap between the two cameras that stitching could occur. If the resulting graph is connected, then we will be able to create a mosaic from the camera array.

based method similar to the network connectivity method used by wireless sensor networks [23]. To generate our graph, we use the following steps:

1. Let each camera be a node in the graph.
2. Let the weight of an edge of the graph be the area of the intersection of the two connected nodes of the graph.
3. Apply a simple threshold to remove any edges of the graph which do not satisfy the requirements for stitching.

If the resulting graph is connected as in Figure 6.3b, this tells us that, from any given camera, we can create a path such that we reach every other camera and all of the paths transversed have an overlap greater than our desired threshold. This is equivalent to saying that we can connect any image from a camera in our array to any other camera image by chaining together feature matches. This allows pairwise stitching together all of the cameras. However, if the graph is disconnected as in Figure 6.3a, then our camera array covers two disconnected scenes and we have no way of understanding how those two scenes should interact.

The threshold chosen represents how much overlap is needed to find sufficient feature matches between cameras to perform stitching. In general, there is a minimal number of feature matches required to compute the proper transformations required for image stitching. Therefore, it would make sense that our threshold could be chosen based on the expected density of features in the scene so that we could try to ensure that minimum number of feature matches is fulfilled. However, the method outlined already gives us the relationship between the camera views from simply the camera pose and the stitching plane. Thus, if we know the stitching plane exactly, this threshold can be 0 and we can simply ensure that our resulting scene is continuous without needing any overlapping feature points. As our uncertainty about our estimated stitching plane grows, so to does the need for a high threshold to ensure that we can perform stitching through traditional means.

The adjacency matrix C of the camera FoV Q_i can be defined as the $N_c \times N_c$ matrix with

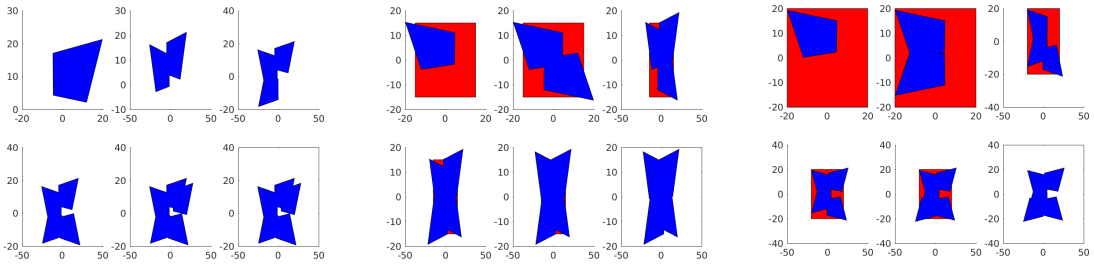
$$C_{ij} = \begin{cases} 1 & A(Q_i \cap Q_j) > \tau \\ 0 & A(Q_i \cap Q_j) \leqslant \tau \end{cases}$$

Using this adjacency, we can check whether C is connected by checking if the matric $C' = \sum_{k=0}^{n} C^k$ has any nonzero elements. Therefore, the block of constraints added to our optimization by the connectivity constraints are:

$$C_{ij} = \begin{cases} 1 & A(Q_i \cap Q_j) > \tau \\ 0 & A(Q_i \cap Q_j) \leqslant \tau \end{cases} \tag{6.6}$$

$$C'_{ij} = \sum_{k=0}^{n} C_{ij}^k \tag{6.7}$$

$$\sum_{i=1}^{N_c} \sum_{j=1}^{N_c} |C'_{ij}| \leqslant 0 \tag{6.8}$$



(**a**) No ROI (**b**) Good ROI (**c**) Too large ROI

Figure 6.4: The placement of cameras in the array following the greedy algorithm for various regions of interest (ROI). Blue shapes are the total coverage of cameras in the array, and red indicates the region of interest used. The plot in the lower right of each figure shows the resulting field of view without the region of interest.

6.1.3 Blind Spots

One of the potential dangers of optimizing camera placement for maximal FoV is that the viewable region that results may be irregularly shaped, as shown in Figure 6.4c. While these figures may offer the largest total FoV, they may not be practical for many applications due to the portions of the scene that are omitted from the region. If these regions contain important information about the scene, then the effective gain in FoV over other camera configurations may be significantly lessened.

While it can be difficult to quantify the effect that coverage irregularity has on effective FoV, there are some approaches that can be used to reduce the amount of coverage irregularity that occurs. While the evaluation and comparison of these approaches is left out as it is beyond the scope of this report, their potential is still worth noting.

First, a region of interest (RoI) can be used to try and focus the FoV on the portions of the scene that need to be covered. Rather than simply maximizing the total size of the coverage region, we instead define a space that we wish to cover and maximize our coverage of that region. This method is sensitive to the choice of RoI. As can be seen in Figure 6.4, different choices of a region of interest can result

in very different coverage regions and having a region that is too large or poorly shaped to be covered by the array given camera placement constraints can be just as bad as having no region of interest.

Another approach is to change the cost function for the optimization to better reflect our desire to create as large a region without blind spots as possible. A region without blind spots can be thought of as a region P such that, for any two points, $x, y \in P$. The direct path is $\overline{xy} \subseteq P$. We can see that a region without blind spots is equivalent to a convex region. Thus, rather than maximizing the area of the FoV, we can instead seek to maximize the largest convex region fully contained within the FoV. Finding the largest convex region inside of a non-convex polygon was originally proposed and dubbed the "potato peeling problem" by Goodman [90] and the "convex skull problem" by Woo [91] and was later solved in polynomial time by Chang and Yap [92]. Other works have found solutions or approximate solutions to sub-problems such as the largest inscribed rectangle [93, 94], longest line segment, or largest ellipse [95] contained within a non-convex polygon.

The largest rectangle problem is of particular interest to camera arrays as cropping the resulting mosaic into a rectangle would cause the camera array to behave more similarly to a traditional single camera set up. However, for the purposes of this chapter, we decided that cropping the mosaic down to the largest inscribed rectangle would disregard too much of the information received by the array.

The method proposed by Chang and Yap solves the potato peeling problem in $O(n^6)$ time. We instead chose to use a method that approximates the solution in $O(n^2)$ time since we need to solve the potato peeling problem for every proposed solution in our discretized solution space. We utilize the Butterfly Lemma provided and the resulting linear time solution when the non-convex polygon has only one reflex corner. By applying this solution to each reflex corner of our non-convex polygon, we generate a series of cuts, each of which is chosen such that it removes as little area from the polygon as possible. By applying each of the cuts to the polygon, we can then generate an approximate solution to the potato peeling problem. This approximation can fail to find the correct solutions when the optimal solution involves chains of butterflies with length more than 1, but generates good solutions

in many cases.

For our optimization, we simply create a function $p(Q)$ which peels the polygon Q into the largest convex polygon $Q' \subset Q$. Rather than maximizing $A(Q)$, we now seek to maximize $A(Q') = A(p(Q))$.

6.1.4 Greedy Heuristic

When we allow more freedom for camera placement, the complexity of the problem quickly makes exhaustive search infeasible. Thus, to improve performance of camera arrays with high amounts of freedom in pose and position, we propose a greedy suboptimal algorithm similar to the one proposed by Horster et al. [52].

In addition, since it is typically best to maximize the angle of the camera relative to the stitching plane, we may miss sections of the scene close to the center in favor of distant portions of the scene where the cameras can cover a lot of area. To ensure that we cover everything important about the scene, we introduce the concept of a region of interest.

We consider the case where we have a region of interest that we wish to cover with the camera array. Our goal now becomes to cover the region of interest while still maximizing the total area that we can see. We are still limited by the stitching constraints, namely that our scene is planar, and that we require a minimum amount of overlap between the cameras so that a sufficient number of feature matches can be gathered. This is similar to the polygon covering problem which is NP-hard [96, 97].

The methods described earlier in the chapter allow us to determine the total field of view of an array of cameras given their position. Therefore, we attempt to build a camera array which will primarily maximize the coverage of the region of interest and secondarily maximize the total field of view.

The greedy algorithm we propose is as follows:

1. Compute the footprint of all possible poses for the camera.

2. If cameras have been placed already, identify the region of overlap with all previously placed cameras and discard all poses which do not overlap with

the previously placed cameras.

3. Identify the region of overlap between each pose and the region of interest.
4. Choose the pose for which this region has maximum area.
5. If two or more poses are tied, choose the pose whose footprint has the maximum area.
6. Remove the chosen footprint from the region of interest.
7. Repeat steps 1–6 for each other camera in your array.

When applied to a sample set of restrictions for a grid based camera array, the resulting field of view for the array after the placement for each camera can be seen in Figure 6.5.

Let us consider a discretization model for a camera array of k cameras that allows each camera to be placed in n different locations, in one of N_p different poses. We can see that using this algorithm, we need to check all N_p poses for each camera, but for the ith camera we need to check only $n - i + 1$ locations to find our choice of solution. Thus, our resulting solution requires simply $\sum_{i=0}^{k-1} N_p(n-i) \leqslant N_p nk$ evaluations of the field of view of an array. Thus, our suboptimal solution is only $O(n)$, whereas the exhaustive optimal solution is $O(a^3 n)$. While this method is not guaranteed to find the optimal solution, it will allow for the computation of a suboptimal solution in a linear time rather than exponential.

6.1.5 Unified FIX Optimization

Including all of our additional connectivity and convexity constraints, we can re-write our optimization as

$$\max_{R_i, \bar{t}_i} A(p(\cup_i Q_i)) \tag{6.9}$$

$$g(\cup Q_i) = 1 \tag{6.10}$$

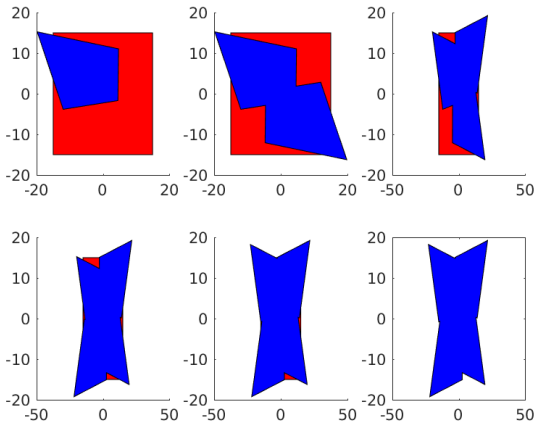


Figure 6.5: The placement of cameras for the greedy algorithm. The blue region denotes what area the camera can see, and the red region denotes the region of interest we wish to cover.

$$Q_i = [B_{i1}, B_{i2}, B_{i3}, B_{i4}]^T \tag{6.11}$$

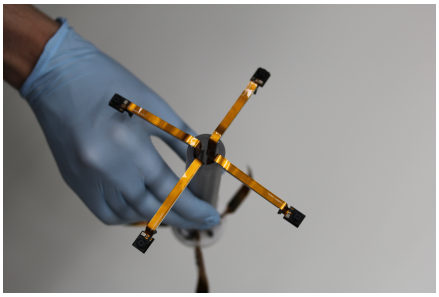
$$B_{ij} = \lambda_{ij} R_i V_j d_i + \bar{t}_i \tag{6.12}$$

$$\lambda_{ij} = \frac{\bar{t}_i [\nu, \tau, \psi]^T + \eta}{R_i V_j d_i [\nu, \tau, \psi]^T} \tag{6.13}$$

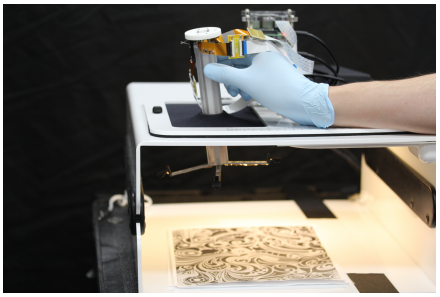
Using this framework, we can either use an extensive search method or our proposed greedy heuristic to evaluate the quality of allowable camera poses and choose an optimal camera array for a given camera space. The resulting camera array should be optimal (or near optimal in the case of the greedy heuristic) over all camera arrays found in the camera space.

6.2 Camera Spaces

As we can see in Figure 6.6, the trocar camera array on which we performed the optimization heavily limited in translation and rotation. We have cameras placed along four telescoping arms. The cameras can be angled towards or away from the center of the array.



(**a**) Trocar-based Camera Array



(**b**) Array and Laparoscopic Trainer Box

Figure 6.6: The trocar-based camera array offers improved FoV over traditional laparoscopes by utilizing multiple visual sensors. The expanded FoV minimizes the need for camera adjustments during surgery.

These constrictions mean that each camera has three degrees of freedom. Let $\gamma \in \{-\frac{\pi}{2}, 0, \frac{\pi}{2}, \pi\}$ denote the rotation due to the choice of arm, $t \in [t_{min}, t_{max}]$ denote the position along that arm, and $\theta \in [\theta_{min}, \theta_{max}]$ be the rotation towards the center of the array. Then, the triple $Cam_i = [\gamma_i, t_i, \theta_i]$ defines a camera position for camera i.

From these parameters, the camera translation and rotation matrices should be defined as follows.

Rotation matrices can be created from their individual components in each of the axes of the computational space.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then, the final rotation matrix can be computed by re-combining the individual matrices.

$$R = R_x(\alpha) * R_y(\beta) * R_z(\gamma) \tag{6.14}$$

In our case, $\alpha = \theta\cos(\gamma)$, $\beta = \theta\cos(\gamma)$.

The translation matrix that should be applied to each camera should be:

$$\bar{t}_i = \begin{bmatrix} t_i sin(\gamma_i) \\ t_i cos(\gamma_i) \\ 0 \end{bmatrix}$$

6.2.1 Asymmetric Camera Space

In our asymmetric camera space, we allow the cameras to be placed in any position and pose allowable by the construction of the camera array. For the trocar array in Figure 6.6, this means that the Cam_i has the full three degrees of freedom γ, t, θ. Cameras are only limited by the fact that no two cameras may share the same $[\gamma, t]$ pair, as no two cameras may be in the same space, and the individual limits on γ, t, and θ that are imposed by the camera space discretization. Using the asymmetric camera space allows us the most freedom for our camera placement, but with $O(3^n)$ degrees of freedom, it also has the largest solution space, which means it limits the allowable discretizations of the camera space.

6.2.2 Symmetric Camera Space

In this case, assume that the array is restricted to placing four of the cameras at the same arm length and angle, with a fifth camera placed at the center of the array to serve as the main reference viewpoint for stitching. In this case, we control the angle of all the cameras θ, and the arm length l. This adds the additional restriction

that

$$\theta = \theta_1 = \theta_2 = \theta_3 = \theta_4 \tag{6.15}$$

$$t = t_1 = t_2 = t_3 = t_4 \tag{6.16}$$

Each of these cameras is placed on its own arm, so

$$\gamma_1 = -\frac{\pi}{2}, \gamma_2 = 0, \gamma_3 = \frac{\pi}{2}, \gamma_4 = \pi$$

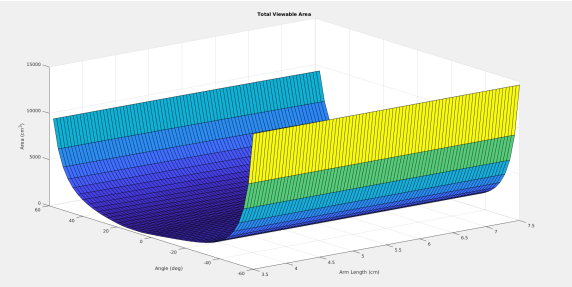
Our fifth camera, we place at the center of the array with $t_5 = 0$, $\theta_5 = 0$, and $\gamma_5 = 0$.

This allows us to simplify the computation to optimization over just two parameters $[t, \theta]$ and can be visualized easily since it has so few degrees of freedoms. The resulting cost function when applied to our surgical set up can be seen in Figure 6.7.

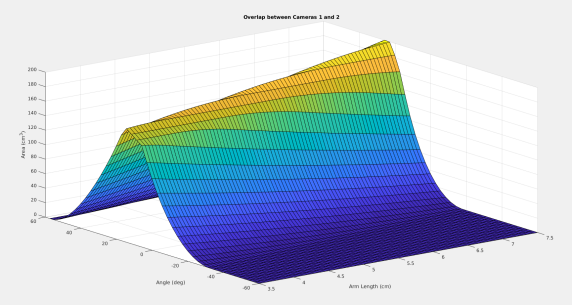
From these figures, we can see that camera angle appears to be the major driving factor for total area and that maximal coverage appears where angle is maximized and overlap is minimized. This seems reasonable as the FOV of an individual camera increases as the camera is rotated relative to the stitching plane, and the total field of view is simply the union of the individual cameras.

6.2.3 Naive Camera Space

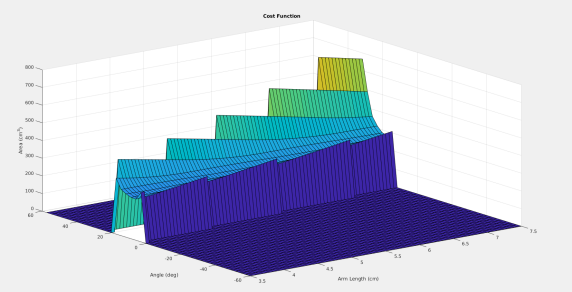
In the naive camera space, we simplify even further down to the single parameter t. The naive camera space represents the approach that one might initially think of for optimizing the trocar camera array. All cameras are placed facing in the same direction as the main reference camera and then moved to create the maximum allowable spread of the cameras. However, we can see that limiting ourselves to a single degree of freedom sacrifices a significant amount of possible coverage. Figure 6.8 shows the improvement in field of view for our laparoscopic test box in using the symmetric approach rather than the naive approach.



(**a**) Total covered area



(**b**) Area of overlap



(**c**) Resulting Cost Function

Figure 6.7: Applying the model to a set of five cameras placed symmetrically on the trocar camera array shows the following: (**a**) the area of the union of the fields of view; (**b**) the area of overlap between Cameras 1 and 2; and (**c**) the cost function that results from thresholding the overlap between each pair of cameras.

(**a**) Naive Approach (**b**) Symmetric Approach

Figure 6.8: Applying symmetric optimization to create camera arrays for use in a laparoscopic trainer box can give significant improvements in total visibility for tasks in the box: (**a**) a frame from the original naive camera system; and (**b**) a similar frame from our symmetrically optimized camera system. The symmetric optimization of camera arrays improves visibility and task performance in the trainer box. No blending techniques were applied so that the individual camera views can be easily picked out from the mosaic.

6.3 Results

The model optimization was run under the constraints required by our image stitching system. The plane of stitching was chosen to be the plane $z = 16.5$ to simulate a camera array placed at the ceiling of a laparoscopic trainer box with the camera array pointed directly at the surgical area. Due to the computational restrictions in computing the result in the exhaustive case, we must limit our space of possible camera poses to a relatively small number. We chose to create an array of five cameras using our trocar based array frame. Each camera was allowed to be placed on one of the four arms of the frame at one of two possible positions on the arm. The camera was then allowed to be rotated into one of three possible poses. Since we cannot have two cameras in the exact same position on the camera arms,

Table 6.1: Results of Maximal Area Optimization Methods on Surgical Array

Approach	FOV Area (cm^2)	Evaluation Time (s)
Naive	578	0.122
Symmetric	955	0.177
Greedy	11,220	0.519
Exhaustive	11,220	38,274
Upper Bound	18,214	0.073

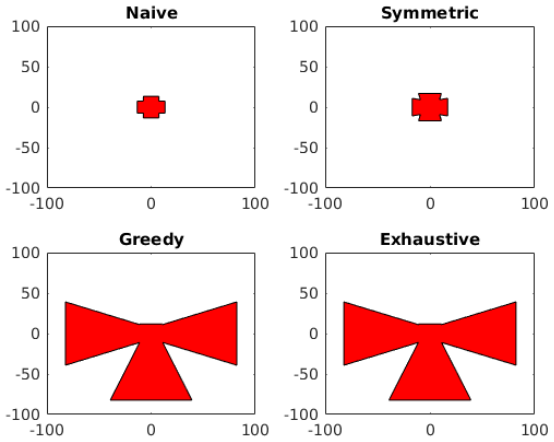


Figure 6.9: The resulting views of the four approaches applied to the restrictions of the trocar camera array.

this gives us $\binom{8}{5}3^5 = 13{,}608$ possible configurations to check in the exhaustive case. The resulting fields of view from this test set up can be seen in Figure 6.9 and the total areas corresponding to each method are shown in Table 6.1.

The upper bound provided is a loose upper bound determined based on the maximum field of view of a single camera. Let us define A_i as the field of view area of a camera at position/pose index $i = 1, 2, 3 \ldots, n$, k as the number of cameras to be placed in the array, and τ as the minimum amount of overlap such that we meet our overlap restrictions from Section 3D. Then, the upper bound U is defined as

$$U = \max_i kA_i - \tau$$

Table 6.2: Results of Maximal Area Optimization on Grid Array

Approach	**FOV Area (cm^2)**	**Evaluation Time (s)**
Naive	215	1.423
Symmetric	795	0.700
Greedy	1153	15.862
Upper Bound	1514	0.203

While this upper bound does not give us an achievable bound that our algorithm could approach, it serves to give us an idea of the result we would get if we could somehow find a way to set up the cameras such that each was individually maximized and they still had the minimum amount of overlap. It is important to note that even the exhaustive case does not manage to approach the upper bound due to the fact that the cameras must be placed in our array and the restrictions due to the discretization of the problem.

Using this upper bound, we can get a feel for how well our greedy algorithm can perform in cases where the total computational complexity for the exhaustive method would be much too high to compute. For example, we consider placing the cameras on a two-dimensional grid such as in the Stanford camera array [98], allowing for rotation around both the x and y axes. For this camera case, our naive approach allows for a camera to be placed at any spot in the array as long as no rotation is applied to it, and our symmetric approach places the cameras on the grid in the same pattern that they would have been on the trocar array with the cameras still arrayed in the cross pattern of the trocar arms and still required to rotate towards or away from the center.

The results of this optimization can be seen in Table 6.2, and the corresponding fields of view can be seen in Figure 6.10.

Next, the constrained system for the surgical camera array is evaluated for the maximal convex region rather than the maximal area region. The resulting regions can be seen in Figure 6.11 and the resultant area comparison can be seen in Table 6.3.

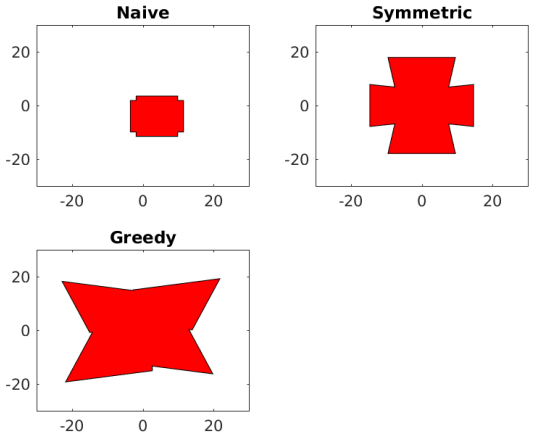


Figure 6.10: The resulting field of view of the three approaches used for the grid based camera array.

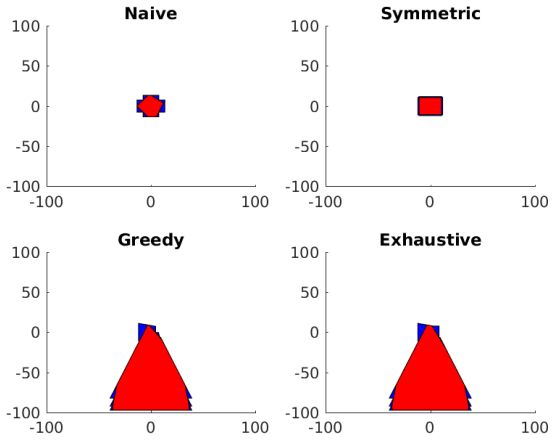


Figure 6.11: The resulting field of view when optimizing for maximal convex region rather than maximum total area. Blue represents the camera FoV and red represents the found convex region.

Table 6.3: Results of Maximal Convex Region Optimization on Surgical Array

Approach	**Max Area (cm^2)**	**Max Conv. Reg. (cm^2)**
Naive	577	577
Symmetric	795	795
Greedy	11,220	5069
Exhaustive	11,220	5104

6.4 Discussion

While the maximal area approach unsurprisingly achieved a larger area than the maximal convex region approach, and the resulting coverage region is nicely symmetrical, as we can see in Figure 6.4, removing the region of interest can cause the construction of the array to become very irregular and it would not be very useful for most mosaicing applications. However, even a poorly matched region of interest does a good job making an array which has a much larger field of view than the naive approach. However, the array coverage may be deceptively large as the result consists of three major regions that are only connected to one another by a very small region.

The maximal convex region approach is able to generate a region which is much more similar to what could be seen by a single large sensor as we desire for image stitching. The cost to the size of the array appears to be very large in this particular case, however, with most of the coverage region being dominated by a single camera and the other sensors not contributing nearly as much to the region. This likely follows from our restrictions on camera placement which cause a single camera with a high level of rotation to achieve a much higher total field of view than a large group of cameras placed so that they fulfill the convexity requirements since there are significantly fewer placements which can create large convex regions.

In addition, it is important to pay attention to restricting camera angle. Since we define our field of view as the intersection of the FoV cone with the stitching plane, if we allow for unrestricted rotation, we can easily allow for infinite FoV with a

single camera by placing our camera with a high enough angle relative to the field of view. However, these higher angled poses also lead to mosaics that do not feel like they accurately model the real world, as we can see in the slight tunnel vision effect that occurs in Figure 6.8, so restricting angle is necessary for good looking mosaics. We could discourage these high rotation cases by simply disallowing them from our array construction, by re-adding the resolution constraints from Mavrinac et al. [22], and/or by our choice of region of interest. Even with the resolution constraints, we require our cameras to be placed further away from the actual scene when using high rotation cameras without a region of interest, so a mixture of the proposed methods in this chapter may be preferable.

6.5 Future Work

The optimization provided here shows that there is significant room for optimization in the choice of camera placement but that poor choice of the optimization parameters can lead to an unusable camera array.

6.5.1 Analyzing Overlap

The overlap constraint used in this work is a naive approach to bounding overlap. We chose to require that the total area of the overlap region must be equal to or greater than the overlap area in our first functional prototype. The general idea behind this choice was that so long as the total area of overlap remained the same, the total number of detected features should remain the same, and this should result in the same quality of image stitching.

This assumption may not always be valid. Feature detection and matching algorithms are noisy and subject to errors. This is the reason that RANSAC is so often used in order to fit homographies even with a significant number of outlying feature points. While a higher number of accurate feature matches will increase the likelihood that RANSAC will fit a good homography, four perfectly chosen feature matches would be sufficient. However, pixel correspondences are limited to integer

values and are thus subject to quantization error, which may be amplified when the transformation is applied to regions outside of the overlap region.

It is clear that the overlap assumption is dependent on the feature detection and matching algorithm, but it is not clear exactly how the algorithm will affect the properties of desired overlap. In addition, the scene itself may affect the quality and quantity of features. Future work should replace the existing overlap constraint with a constraint that more accurately reflects the feature matching dependencies of image stitching. If features could be added to the scene in order to loosen the feature matching constraint or to lessen the dependency on the contents of the scene for feature density.

6.5.2 Depth Robust Field of View Maximization

While image stitching relies on assumptions about the planar nature of the scene, these assumptions can be loosened by parallax mitigation techniques such as the one detailed in Part 1 of this report. Remember that the final FoV is found by computing the intersection of the view cones with the planar scene. Thus, when the planar assumption no longer holds, the optimality of the camera placement begins to suffer. The field of view computed for one depth d will not be the same as the field of view for the perturbed depth $d + \epsilon$. In some cases, this can lead to objects which should be visible in the scene falling into blind spots that lie off the plane of stitching. Future work will need to investigate how best to account for noisy depth information or scenes that may not be exactly planar.

I believe that the best approach to solve this problem would be to identify a new objective function that accounts for the allowable scene depth. Either by evaluating the FoV as a 3D region in space, or by shrinking the planar FoV into the largest region which will still lie within the FoV for the whole set of allowable distances $d + \epsilon$.

6.5.3 Ensuring Coverage of space for 3D rendering

Depth information is very important in surgical procedures. One of the most commonly cited complaints about the TCA in an early round of testing was that the top down view provided by the TCA made it much harder to determine depth than in a traditional laparoscope. As such, gathering a full 3D model of the scene may be preferable to the composite image. 3D modeling extends the problem of optimal camera placement into three dimensions and introduces new considerations which must be taken into account. In order for accurate 3D models to be generate from a scene, feature correspondence must be computed as in image stitching. However, unlike in image stitching, feature correspondences can not be extrapolated out to regions that are only captured by a single camera. Furthermore, as scenes are no longer planar, the FoV model needs to be updated to handle occlusion. Objects may not lie in the field of view if another object blocks vision from the camera or if the surface of the object is facing at too sharp of an angle to be seen clearly.

Expanding our work in optimal camera placement from the 2D to 3D visualization case offers several potential challenges:

- Computing total field of view as the volume of the region in space 3D space.
- Reduce the total field of view to only that region which can be seen in at least 2 cameras.
- Detect opportunities for occlusion and account for them in objective function

6.6 Conclusions

In this chapter, a new optimization scheme for the purposes of image and video stitching is proposed. The constraints of stitching are leveraged to propose a set of constraints on camera placement and view. Using those constraints, a FIX optimization problem for placing a set number of cameras to create maximal FoV is proposed.

Since the FoV of an individual camera on the scene exhibits nonlinear properties, we cannot use linear programming methods to perform the optimization. Instead, the solution space is discretized into a finite discrete set of positions and poses. To ensure accuracy of the stitched image if the estimate for the stitching plane is incorrect, we additionally constrain our solution space such that there must be sufficiently large overlap area between images captured by a pair of cameras. The amount of overlap required will depend on the feature density of the scene, thus this is left as a tuneable parameter depending on the scene.

This problem is formulated as a constrained discrete optimization problem and show the solution space grows exponentially as the number of camera grows. A sub-optimal, greedy heuristic for solving this problem in polynomial time is presented to allow the extension of this problem to less heavily constrained arrays. The greedy algorithm is not intended to be an optimal solution to the optimization problem proposed in the chapter. Instead, it is intended to be a method for evaluating the optimization model for complex camera arrays and improve camera array placement over the naive approach that can be computed quickly even when run on a large space of possible camera placements. This would be a first attempt to maximize the field of view of the stitched image subject to the overlapping constraint required for proper stitching.

To do so, a continuity constraint was introduced to ensure that cameras had sufficient overlap and a continuous path of camera overlap so that stitching could be performed. In addition, there is a requirement that the coverage region be free of holes, since many optimal solutions that do have holes in them do not make for good mosaics.

To further tailor the optimization to the needs of the stitching setting, a new measure is proposed for maximal FoV that better reflects the stitching desire to create a large region that mimics the behavior of a single visual sensor. Rather than focusing on the total area of the coverage region, the focus is on a subset of the coverage region. First, a region of interest based approach was considered, but it was found that the behavior of this approach is very heavily dependent on the chosen region of interest, with size variations in the region leading to some

potentially unusable camera arrays. Next, a variant of the potato peeling algorithm, seeking to find the largest convex region that fits inside the FoV. This seems to behave more as desired, but does not take into account that some regions may be more desirable to monitor than others. However, these two approaches give the user more ways to control how they wish to place cameras to monitor a scene.

CHAPTER 7
Conclusion

The above research describes progress on generating composite images for remote scene interaction, and it proposes a path forward to guide future research. It makes several contributions to computer vision and 3D geometry:

- **Computer Vision**
  - **A video stitching pipeline**, which corrects for parallax by segmenting foreground objects and using a multi-planar model to fit aligning transformations to the scene even when the camera array is moving or the scene background is slowly changing.
  - **A method** for leveraging shape prior information to compute aligning homographies for feature poor regions.
- **3D Geometry**
  - **A model** for computing the field of view of a camera array being used for image stitching. This model serves as the basis for evaluating the quality of the poses and positions given to the cameras in the array.
  - **A model** for how features can be projected onto a scene to control the required overlap between cameras for optimal image stitching.
  - **An optimization framework**, for computing the optimal camera placement for image stitching. This model should optimize the total area of the FoV while still being robust to parallax in the scene to encourage parallax correction in non-planar scenes.

– **An optimization framework**, for computing the optimal placement for 3D modeling. This model will attempt to optimize the volume of the FoV while accounting for the need for multi-camera coverage at every point in the FoV. The model should take into account the possibility of occlusion and the need for multiple viewing angles to avoid holes in the 3D model.

Bibliography

[1] R. Szeliski, "Image Alignment and Stitching: A Tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[2] Y. S. Chen and Y. Y. Chuang, "Natural image stitching with the global similarity prior," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9909 LNCS, pp. 186–201, 2016.

[3] J. Zaragoza, T. J. Chin, Q. H. Tran, M. S. Brown, and D. Suter, "As-projective-as-possible image stitching with moving DLT," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1285–1298, 2014.

[4] J. Gao, S. J. Kim, and M. S. Brown, "Constructing Image Panoramas using Dual-Homography Warping," *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 49—-56, 2011.

[5] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Transactions on Graphics*, vol. 28, no. 3, p. 1, 2009.

[6] C. H. Chang, C. J. Chen, and Y. Y. Chuang, "Spatially-varying image warps for scene alignment," in *Proceedings - International Conference on Pattern Recognition*, pp. 64–69, 2014.

[7] Y. Lu, Z. Hua, K. Gao, and T. Xu, "Multiperspective image stitching and regularization via hybrid structure warping," *Computing in Science and Engineering*, vol. 20, no. 2, pp. 10–23, 2018.

[8] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang, S. Watson, and M. Gross, "Panoramic Video from Unstructured Camera Arrays," *Computer Graphics Forum*, vol. 34, no. 2, pp. 57–68, 2015.

[9] A. Mills and G. Dudek, "Image stitching with dynamic elements," *Image and Vision Computing*, vol. 27, no. 10, pp. 1593–1602, 2009.

[10] B. He and S. Yu, "Parallax-robust surveillance video stitching," in *Sensors (Switzerland)*, vol. 16, pp. 1–12, 2015.

[11] J. Chen, N. Li, and T. Liao, "Graph-based Hypothesis Generation for Parallax-tolerant Image Stitching," *arXiv preprint arXiv:1804.07492*, pp. 4–6, 2018.

[12] J.-J. Kim, A. Watras, H. Liu, Z. Zeng, J. A. Greenberg, C. P. Heise, Y. H. Hu, and H. Jiang, "Large-Field-of-View Visualization Utilizing Multiple Miniaturized Cameras for Laparoscopic Surgery," *Micromachines*, vol. 9, no. 9, p. 431, 2018.

[13] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *International Conference on Computer Vision*, 1999.

[14] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," in *European conference on computer vision*, pp. 404–417, 2006.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[16] S. Hengstler, D. Prashanth, S. Fong, H. Aghajan, and U. States, "MeshEye : A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance," in *Proceedings of the 6th international conference on Information processing in sensor networks.*, 2007.

[17] G. Carrera, A. Angeli, and A. J. Davison, "SLAM-Based Automatic Extrinsic Calibration of a Multi-Camera Rig," in *IEEE International Conference on Robotics and Automation*, 2011.

[18] B. K. Cabral, F. S. Briggs, J. Hsu, A. P. Pozo, and A. H. Coward, "Three Dimensional, 360-Degree Virtual Reality Camera System," 2019.

[19] R. Anderson, D. Gallup, J. T. Barron, N. Snavely, C. Hern, and S. M. Seitz, "Jump : Virtual Reality Video," *ACM Transactions on Graphics*, 2016.

[20] A. Kanhere, K. L. V. Grinsven, C.-c. Huang, Y.-s. Lu, J. A. Greenberg, C. P. Heise, Y. H. Hu, H. Jiang, and S. Member, "Multicamera Laparoscopic Imaging With Tunable Focusing Capability," *Journal of microelectromechanical systems*, vol. 23, no. 6, pp. 1290–1299, 2014.

[21] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai, "A Survey of Sensor Planning in Computer Vision," in *IEEE Transactions on Robotics and Automation*, vol. 11, 1995.

[22] A. Mavrinac and X. Chen, "Modeling coverage in camera networks: A survey," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 205–226, 2013.

[23] J. Zhao, D. Haws, R. Yoshida, and S. C. S. Cheung, "Approximate techniques in solving optimal camera placement problems," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2013, pp. 1705–1712, 2011.

[24] Y.-g. Fu, J. Zhou, and L. Deng, "Surveillance of a 2D Plane Area with 3D Deployed Cameras," *Sensors*, vol. 14, pp. 1988–2011, 2014.

[25] C. Piciarelli, C. Micheloni, and G. Foresti, "PTZ Camera Network Reconfiguration," in *Proceedings of the Third ACM/IEEE International Conference on Distributed Smart Cameras*, 2009.

[26] Y. Sun, A. Anderson, C. Castro, B. Lin, R. Gitlin, S. Ross, and A. Rosemurgy, "Virtually transparent epidermal imagery for laparo-endoscopic single-site surgery," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, no. August, pp. 2107–2110, 2011.

[27] A. L. Anderson, B. Lin, and Y. Sun, "Virtually transparent epidermal imagery (VTEI): On new approaches to in vivo wireless high-definition video and image processing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 6, pp. 851–860, 2013.

[28] R. Szeliski, "Image mosaicing for tele-reality applications," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pp. 44–53, 1994.

[29] R. Szeliski, "Video mosaics for virtual environments," *IEEE Computer Graphics and Applications*, vol. 16, no. 2, pp. 22–30, 1996.

[30] R. Szeliski, "Image Alignment and Stitching: A Tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[31] B. Luo, F. Xu, C. Richardt, and J. H. Yong, "Parallax360: Stereoscopic 360 scene representation for head-motion parallax," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 4, pp. 1545–1553, 2018.

[32] C. Schroers, J.-C. Bazin, and A. Sorkine-Hornung, "An Omnistereoscopic Video Pipeline for Capture and Display of Real-World VR," *ACM Transactions on Graphics*, vol. 37, no. 3, pp. 1–13, 2018.

[33] Z. Tan, S. Zhang, and R. Wang, "Stable stitching method for stereoscopic panoramic video," *CAAI Transactions on Intelligence Technology*, vol. 3, no. 1, pp. 1–7, 2018.

[34] C. Xie, X. Zhang, H. Yang, L. Chen, and Z. Gao, "Video Stitching Based on Optical Flow," *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB*, vol. 2018-June, pp. 1–5, 2018.

[35] Q. Liu, X. Su, L. Zhang, and H. Huang, "Panoramic video stitching of dual cameras based on spatio-temporal seam optimization," *Multimedia Tools and Applications*, 2018.

[36] T. Bertel, N. D. F. Campbell, and C. Richardt, "MegaParallax: Casual 360 Panoramas with Motion Parallax," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 5, pp. 1–1, 2019.

[37] J. Civera, A. J. Davison, J. A. Magallón, and J. M. Montiel, "Drift-free real-time sequential mosaicing," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 128–137, 2009.

[38] A. Limonov, X. Yu, L. Juan, C. Lei, and Y. Jian, "Stereoscopic realtime 360-degree video stitching," *2018 IEEE International Conference on Consumer Electronics, ICCE 2018*, vol. 2018-Janua, pp. 1–6, 2018.

[39] E. Molina, Z. Zhu, and C. N. Taylor, "A layered approach for fast multi-view stereo panorama generation," *Proceedings - 2011 IEEE InternationalSymposium on Multimedia, ISM 2011*, pp. 589–594, 2011.

[40] J. Eledath, L. Mcdowell, M. Hansen, L. Wixson, A. Pope, G. Gendel, and W. Rd, "Mosaic Construction and Moving Object from a Moving Camera," *International Journal*, pp. 284–285, 1998.

[41] J. Li, W. Xu, J. Zhang, M. Zhang, Z. Wang, and X. Li, "Efficient Video Stitching Based on Fast Structure Deformation," *IEEE Transactions on Cybernetics*, vol. 45, no. 12, pp. 2707–2719, 2015.

[42] M. Zheng, X. Chen, and L. Guo, "Stitching video from webcams," *Advances in Visual Computing*, pp. 420–429, 2008.

[43] K. Krishnakumar and S. I. Gandhi, "Video stitching using interacting multiple model based feature tracking," *Multimedia Tools and Applications*, vol. 78, no. 2, pp. 1375–1397, 2019.

[44] S. Galliani, K. Lasinger, and K. Schindler, "Massively parallel multiview stereopsis by surface normal diffusion," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 873–881, 2015.

[45] G. P. Moustris, S. C. Hiridis, K. Deliparaschos, and K. Konstantinidis, "An actuated force feedback-enabled laparoscopic instrument for robotic-assisted surgery," *The international journal of medical robotics + computer assisted surgery : MRCAS*, vol. 7, no. April, pp. 375–392, 2011.

[46] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 1–13, 2013.

[47] P. Moulon, P. Monasse, R. Marlet, P. Moulon, P. Monasse, R. Marlet, and A. Structure, "Adaptive Structure from Motion with a contrario model estimation," in *Asian Conference on Computer Vision*, pp. 257–270, 2012.

[48] E. Cerezo, F. Pérez, X. Pueyo, F. J. Seron, and F. X. Sillion, "A survey on participating media rendering techniques," *Visual Computer*, vol. 21, no. 5, pp. 303–328, 2005.

[49] J. Geng, "Structured-light 3D surface imaging: a tutorial," *Advances in Optics and Photonics*, vol. 3, no. 2, p. 128, 2011.

[50] R. Lange and P. Seitz, "Time-of-Flight Range Camera," *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390–397, 2001.

[51] J. Urrutia, *Art Gallery and Illumination Problems*. Woodhead Publishing Limited, 2000.

[52] E. Hörster and R. Lienhart, *Optimal Placement of Multiple Visual Sensors*. Elsevier Inc, 2009.

[53] C. Sterle, A. Sforza, A. E. Amideo, and C. Piccolo, "A unified solving approach for two and three dimensional coverage problems in sensor networks," *Optimization Letters*, vol. 10, no. 5, pp. 1101–1123, 2016.

[54] U. M. Erdem and S. Sclaroff, "Optimal Placement of Cameras in Floorplans to Satisfy Task Requirements and Cost Constraints," in *OMNIVIS 2004*, 2004.

[55] K. Chakrabarty, S. Member, S. S. Iyengar, and H. Qi, "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448–1453, 2002.

[56] C. Wang, F. Qi, G. Shi, and X. Wang, "A sparse representation-based deployment method for optimizing the observation quality of camera networks," *Sensors (Switzerland)*, vol. 13, no. 9, pp. 11453–11475, 2013.

[57] J. Zhao, S. c. S. Cheung, and T. Nguyen, *Optimal Visual Sensor Network Configuration*. Elsevier Inc, 2009.

[58] F. Angella, L. Reithler, and F. Gallesio, "Optimal deployment of cameras for video surveillance systems," *2007 IEEE Conference on Advanced Video and Signal Based Surveillance, AVSS 2007 Proceedings*, pp. 388–392, 2007.

[59] R. Chabra, A. Ilie, N. Rewkowski, Y. W. Cha, and H. Fuchs, "Optimizing placement of commodity depth cameras for known 3D dynamic scene capture," *Proceedings - IEEE Virtual Reality*, pp. 157–166, 2017.

[60] F. Zhang and F. Liu, "Parallax-tolerant image stitching," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3262–3269, 2014.

[61] X. Zhang, X. Chen, and J. L. Alarcon-herrera, "3-D Model-Based Multi-Camera Deployment : A Recursive Convex Optimization Approach," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 3157–3169, 2015.

[62] R. Malik, P. Bajcsy, R. Malik, P. Bajcsy, A. Placement, S. Cameras, and T. Workshop, "Automated Placement of Multiple Stereo Cameras To cite this version : HAL Id : inria-00325382 Cameras," in *The 8th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras - OMNIVIS*, 2008.

[63] A. A. Altahir, V. S. Asirvadam, N. H. Hamid, P. Sebastian, N. Saad, R. Ibrahim, and S. C. Dass, "Sensor systems Modeling Multicamera Coverage for Placement Optimization," *IEEE Sensors*, vol. 1, no. 6, pp. 2–5, 2017.

[64] R. Szeliski, "Computer Vision : Algorithms and Applications," *Computer*, vol. 5, p. 832, 2010.

[65] C. Herrman, C. Wang, R. S. Bowen, E. Keyder, and R. Zabih, "Object Centered Image Stitching," in *Proceedings of the European Conference on Computer Vision*, Lecture Notes in Computer Science, Springer International Publishing, nov 2018.

[66] T.-z. Xiang, G.-s. Xia, X. Bai, and L. Zhang, "Image stitching by line-guided local warping with global similarity constraint," *Pattern Recognition*, vol. 83, pp. 481–497, 2018.

[67] C. Baillard and A. Zisserman, "Automatic reconstruction of piecewise planar models from multiple views," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 559–565, 1999.

[68] P. Sturm and S. Maybank, "A Method for Interactive 3D Reconstruction of Piecewise Planar Objects from Single Images," tech. rep., 1999.

[69] C. Mura, O. Mattausch, and R. Pajarola, "Piecewise-planar Reconstruction of Multi-room Interiors with Arbitrary Wall Arrangements," *Computer Graphics Forum*, vol. 35, no. 7, pp. 179–188, 2016.

[70] W. Xi and X. Chen, "Reconstructing piecewise planar scenes with multi-view regularization," *Computational Visual Media*, vol. 5, pp. 337–345, dec 2019.

[71] F. Fraundorfer, K. Schindler, and H. Bischof, "Piecewise planar scene reconstruction from sparse correspondences," *Image and Vision Computing*, vol. 24, pp. 395–406, apr 2006.

[72] S. N. Sinha, D. Steedly, and R. Szeliski, "Piecewise planar stereo for image-based rendering," in *2009 International Conference on Computer Vision*, pp. 1881–1888, 2009.

[73] E. Bayraktar and P. Boyraz, "Analysis of Feature Detector and Descriptor Combinations with a Localization Experiment for Various Performance Metrics," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 25, no. 3, pp. 2444–2454, 2017.

[74] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2281, 2012.

[75] K.-Y. Lee and J.-Y. Sim, "Warping Residual Based Image Stitching for Large Parallax," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognitio*, 2020.

[76] Y.-H. Tsai, M.-H. Yang, and M. J. Black, "Video Segmentation via Object Flow," in *CVPR*, 2016.

[77] H. Rashed, S. Yogamani, A. El-Sallab, A. Das, and M. El-Helw, "Depth Augmented Semantic Segmentation Networks for Automated Driving," in *Communications in Computer and Information Science*, vol. 1019 CCIS, pp. 1–13, jan 2019.

[78] L. C. Garc, W. Li, C. Gruijthuijsen, D. Stoyanov, T. Vercauteren, L. C. García-Peraza-Herrera, W. Li, C. Gruijthuijsen, A. Devreker, G. Attilakos, J. Deprest, E. V. Poorten, D. Stoyanov, T. Vercauteren, and S. Ourselin, "Real-Time Segmentation of Non-Rigid Surgical Tools based on Deep Learning and Tracking," *International Workshop on Computer-Assisted and Robotic Endoscopy*, vol. 10170 LNCS, pp. 1–12, 2016.

[79] S. Bodenstedt, S. Speidel, M. Allan, A. Agustinos, X. Du, D. Pakhomov, R. Sznitman, M. Teichmann, M. Thoma, and T. Vercauteren, "Comparative evaluation of instrument segmentation and tracking methods in minimally invasive surgery," *arXiv preprint arXiv:1805.02475*, 2018.

[80] J. L. Schonberger and J. M. Frahm, "Structure-from-Motion Revisited," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 4104–4113, 2016.

[81] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism," *ACM Transactions on Graphics*, vol. 25, no. 3, p. 835, 2006.

[82] A. Khatamian and H. R. Arabnia, "Survey on 3D surface reconstruction," *Journal of Information Processing Systems*, vol. 12, no. 3, pp. 338–357, 2016.

[83] P. Labatut, J. P. Pons, and R. Keriven, "Robust and efficient surface reconstruction from range data," *Computer Graphics Forum*, vol. 28, no. 8, pp. 2275–2290, 2009.

[84] M. Jancosek and T. Pajdla, "Multi-view reconstruction preserving weakly-supported surfaces," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3121–3128, 2011.

[85] M. Jancosek and T. Pajdla, "Exploiting Visibility Information in Surface Reconstruction to Preserve Weakly Supported Surfaces," *International Scholarly Research Notices*, vol. 2014, pp. 1–20, 2014.

[86] M. Allan, A. Shvets, T. Kurmann, Z. Zhang, R. Duggal, Y.-H. Su, N. Rieke, I. Laina, N. Kalavakonda, S. Bodenstedt, L. Herrera, W. Li, V. Iglovikov, H. Luo, J. Yang, D. Stoyanov, L. Maier-Hein, S. Speidel, and M. Azizian, "2017 Robotic Instrument Segmentation Challenge," feb 2019.

[87] Richard Hartley, *Multiple View Geometry In Computer Vision*. 2000.

[88] C. M. Shakarji, "Least-Squares Fitting Algorithms of the NIST Algorithm Testing System," *J. Res. Natl. Inst. Stand. Technol*, vol. 103, no. 6, 1998.

[89] B. R. Vatti, "A generic solution to polygon clipping," *Communications of the ACM*, vol. 35, no. 7, pp. 56–63, 1992.

[90] J. E. Goodman, "On the largest convex polygon contained in a non-convex n-gon, or how to peel a potato," *Geometriae Dedicata*, vol. 11, no. 1, pp. 99–106, 1981.

[91] T. C. Woo, *The convex skull problem*. 1986.

[92] J. S. Chang and C. K. Yap, "A polynomial solution for the potato-peeling problem," *Discrete & Computational Geometry*, vol. 1, no. 1, pp. 155–182, 1986.

[93] R. Molano, P. G. Rodríguez, A. Caro, and M. L. Durán, "Finding the largest area rectangle of arbitrary orientation in a closed contour," *Applied Mathematics and Computation*, vol. 218, no. 19, pp. 9866–9874, 2012.

[94] C. Knauer, L. Schlipf, J. M. Schmidt, and H. R. Tiwary, "Largest inscribed rectangles in convex polygons," *Journal of Discrete Algorithms*, vol. 13, pp. 78–85, 2012.

[95] O. Hall-Holt, M. J. Katz, P. Kumar, J. S. Mitchell, and A. Sityon, "Finding large sticks and potatoes in polygons," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 474–483, 2006.

[96] J. C. Culberson and R. A. Reckhow, "Covering Polygons is Hard," *J. Algorithms*, vol. 17, no. 1, pp. 2–44, 1991.

[97] J. O' rourke and K. J. Supowit, "Some NP-Hard Polygon Decomposition Problems," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 181–190, 1983.

[98] B. Wilburn, N. Joshi, V. Vaish, E. V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High performance imaging using large camera arrays," in *ACM Transactions on Graphics*, vol. 24, pp. 765–776, 2005.