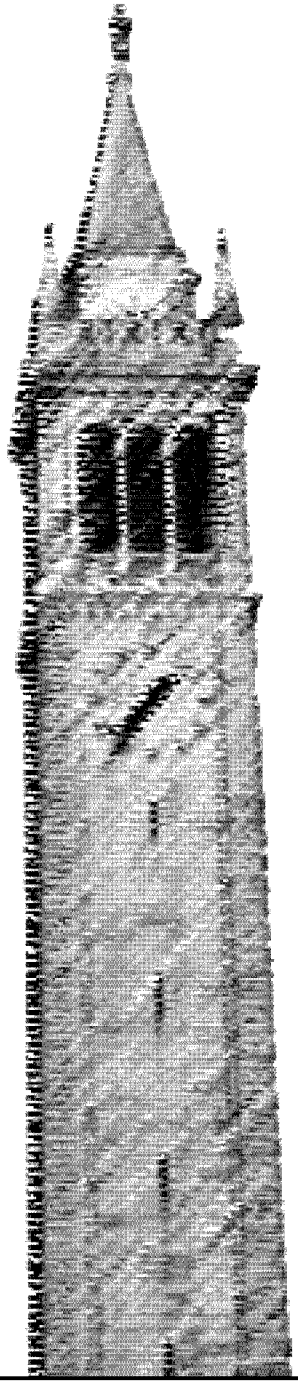
Towards Ubiquitous Augmented Reality in Structured Environments



Joseph Menke

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-237
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-237.html

December 1, 2022

Towards Ubiquitous Augmented Reality in Structured Environments

by

Joseph Andreas Menke

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor S. Shankar Sastry, Chair
Professor Yi Ma
Professor Luisa Caldas

Summer 2020

Towards Ubiquitous Augmented Reality in Structured Environments

Abstract

Towards Ubiquitous Augmented Reality in Structured Environments

by

Joseph Andreas Menke

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor S. Shankar Sastry, Chair

Augmented reality has incredible potential to change the way we interact with information. Smart mobile devices have enabled us to be constantly connected to an ever expanding source of information known as the internet. Augmented Reality devices can present a method for organizing this information and ensure that relevant information is present in the location where it is most useful. Similarly, the spatial organization of information alongside physical objects can present new forms of creativity and artistry. These devices use computer vision algorithms to understand the structure of objects in the world as well as the devices' location relative to these objects. This understanding is used to display virtual objects in such a way as to appear present in the physical world. While these algorithms present a viable solution in many situations, they still have many failure cases that can prevent the ubiquitous adoption of these devices.

This thesis discusses methods for improving the localization and mapping capabilities of these Augmented Reality devices by exploiting the structure that is present in many man-made environments. A discussion of some open problems in augmented reality is presented in the context of the open source package OpenARK. The remainder of thesis discusses methods for improving localization and mapping and 3D reconstruction. A method is presented for incorporating planar structures from a Time of Flight depth sensor into a state of the art Visual Inertial Odometry algorithm. This algorithm is demonstrated to improve localization accuracy in low light and low texture environments and maintain realtime performance on a mobile device with limited Time of Flight depth sensing range and limited compute resources. A method is presented for enabling the realtime matching of image wireframes. Image wireframes represent the junctions, lines and intersection relationships that form the structure of the scene. This algorithm exploits these relationships to improve the matching of image wireframes beyond standard feature matching. This method is further demonstrated to be capable of exploiting the additional constraints introduced by multiple cameras.

Contents

List of Figures

List of Tables

Acknowledgments

This thesis would never have happened if it were not for the interactions with so many wonderful people. First is my advisor, Professor Shankar Sastry, thank you for being an incredible source of guidance and insight into the future of research. Your encouragement and enthusiasm has always made me believe this future was in my grasp. Second is my closest collaborator and mentor for the last 3 years, Dr. Allen Yang. Thank you for all the support, resources, advice, and leadership. Thank you for being willing to take a chance on me and always believing in me. Most importantly, thank you for the incredible adventure that has been OpenARK. You have opened so many doors to so many experiences I never thought I would have.

Many thanks to Professor Yi Ma. Our discussions on computer vision have done much to shape my research interests, and the prior work of you and your students is the backbone on which this thesis is built. Many thanks to Professor Luisa Caldas, whose work on combining architecture and virtual reality provides important context to my research. Many many thanks to Professor Ruzena Bajcsy whose caring insight has shaped so many of my views on research and science.

Sastry group is filled with wonderful and inspiring graduate students. I am appreciative of absolutely everyone in this group for making our lab a place I looked forward to coming to every day. Joshua Achiam, who will always be one of my best friends, our early discussions on AI were some of the best times of my years at Berkeley. Dr. Jaime Fisac, who taught me about ethics in technology and the responsibilities of researchers. David McPherson, who stepped up to take on many of my responsibilities just when they were becoming overwhelming. Dr. Dexter Scobee who I have turned to for advice on many occasions on everything from control systems to life after graduate school. Oladapo Afolabi, who has shared my research interests and who took on many of the responsibilities for OpenARK, your support and collaboration has been invaluable. Tyler Westenbroek, who has supported many a discussion about the future of research and technology, and in whose capable hands I leave this lab tradition. Chih-Yuan (Frank) Chiu, I have learned more about SLAM from our discussions than from any other source and your mathematical brilliance continues to amaze me. Thank you to all the others who continue to inspire me with their research and collaboration: Dr. Roy Dong, Kamil Nar, Eric Mazumdar, Kshama Dwarakanath, Michael Estrada, Victoria Tuck, Valmik Prabhu, and Yuri Murakami.

I would also like to thank those collaborators from my prior research group from whom I have learned so much. Dr. Eric Tuner and Dr. Nichoas Corso who taught me so much about computer vision and whose guidance led me through my first years at Berkeley. Many thanks to Dr. David Fridovich-Keil for many years of friendship and for being the excellent researcher and wonderful human that I aspire to be.

Jessica Gamble and Huo Chao Kuan, thank you so much for all your kindness over the years and especially for all the hard work in supporting our research efforts. Our lab would not run without you. I cannot thank enough Shirley Solanio. You are the person myself

and so many others turn to when we don't know what to do and your help and support has meant so much to me.

All of the students who worked with me on OpenARK, I am constantly amazed by the things you are able to accomplish. Especially Bill Zhou, Sixian (Alex) Yu, and Adam Chang, without whom, OpenARK would not exist.

My collaborators at Google who supported the work on planar structures and who put in extra work to ensure I could include this work in my thesis: Chucai Yi, Chao Guo, Ryan DuToit, and Hongsheng Yu. Thank you for a wonderful internship experience.

Thank you to my many mentors at UC Riverside who prepared me for this journey and who gave me so many opportunities: Professor Matthew Barth, Dr. Mike Todd, Professor Kent Johnson, Professor Roman Chomko, Professor Phillip Brisk, and Jun Wang.

Thank you to my family. My dad, John Menke, will always be the coolest person I know, and his ability to fix anything is what inspired me to become an engineer. My mom Karina Menke, is the source of my work ethic and sense of responsibility. I could not have accomplished what I have without you. Thank you both for always encouraging me to follow my dreams, whatever they may be. My brother, Dr. Michael Menke, who taught me the art of debate and how to discuss ideas. You have always been a guide showing me what is possible in life. My sister-in-law Dr. Francesca Parente, for showing me that it is indeed possible to best my brother in debate, and for giving me new goals to strive for. Kermit and Ada Payne, for welcoming me into your lives, and for raising a most amazing daughter.

The biggest thanks of all to my brilliant fiancee, Dr. Margaret Payne. Maggie is my inspiration, my guiding light, and the most caring human I have ever met. She has been my accomplice in every challenge and kept me going through all the struggles. There is no way I could have done this without her.

This research was supported in part by the Office of Naval Research under grant N00014-19-1-2055

Chapter 1

Open Augmented Reality Kit

1.1 Introduction

Open Augmented Reality Kit (OpenARK) is a software development kit that explores those algorithms necessary for state of the art Augmented Reality (AR). OpenARK is capable of performing Simultaneous Localization And Mapping (SLAM) that can accurately track devices across entire buildings. OpenARK is also capable of generating accurate 3D models of the environments, enabling realistic occlusions as well as interactions between the environment and virtual objects. Lastly, OpenARK performs hand tracking that enables the user to interact with these virtual devices through gestures. We developed OpenARK to provide an open source way for students to learn about the computer vision algorithms necessary for Augmented Reality and a fast way for companies and researchers to get started developing for Augmented Reality. This work builds upon several other open source packages, combining these works and adding our own algorithms to provide a complete Augmented Reality solution.

In this chapter we present a breif high-level introduction to those algorithms used in OpenARK and discuss how they are combined to enable an effective AR experience. We also discuss how our work on this kit has motivated the open problems in SLAM and 3D reconstruction that we begin to address in the remainder of the thesis.

1.2 OpenARK Hardware

In addition to a computer, OpenARK utilizes the commercially available Intel Realsense d435i camera system. This camera system consists of: one global shutter RGB camera, two time-synchronized global shutter IR cameras, an IR dot matrix projector, an Inertial Measurement Unit (IMU), and an internal hardware clock. Each of these components contributes to OpenARK's ability to provide a compelling augmented reality experience.

The two time-synchronized global shutter IR cameras are utilized for localization and mapping. Global shutter cameras capture the entire image frame at the same time instant.

Rolling shutter cameras, which are commonly used in modern smartphone devices, capture images a single row at a time. While this keeps costs low, motion of the camera introduces distortion into the captured image of rolling shutter cameras. While the effects of rolling shutter cameras can be estimated [54, 25], this introduces additional noise and complexity to the algorithm.

The time synchronization between the between these cameras enables the formation of a stereo pair. In this way, features in these cameras can easily be matched and their 3D positions triangulated. The IR dot matrix projector, projects a precise dot pattern onto the environment that enables image patches to be matched across these two image frames even when no visual texture would otherwise be present (such as the camera viewing a white wall). This enables a dense depth image to be produced for many scenes. This type of camera arrangement is often called a structured light depth camera [4]. Note that these dot matrix features do not, however, represent features that can be matched over time as their positions are not fixed in the environment but rather move as the camera moves.

The Inertial Measurement Unit (IMU) provides information about the relative transformation between camera frames. Not only does this added information improve the accuracy of the overall location estimate, it adds robustness by providing information that may otherwise be unavailable. For example, fast rotational motion induces heavy motion blur into a camera capture. This can prevent features being identified and matched across images. A system that only relied on visual information would have no way to estimate the pose of the camera in these scenarios. The IMU can provide an estimate of the rotation and translation in these scenarios. An important aspect of this is the internal hardware clock. This provides accurate timestamps to all sensors in the system. Specifically this enables the system to know the time difference between when a camera image and an IMU measurement arrives. While methods have been proposed for estimating these timestamps [25], a single hardware clock contributes to a simpler and more accurate system.

We have optimized OpenARK to make use of these various components in order to provide an experience that is computationally efficient while enabling state of the art capabilities.

1.3 OpenARK SLAM

When a virtual object is "placed" at a specific position in the world, the AR system uses an estimated pose to compute how this virtual object would have looked from this perspective if it were physically placed in the environment. This artificial view is then rendered over the view of the environment and displayed to the user, enabling the illusion of presence. The process of precisely estimating the pose of the AR device is called Simultaneous Localization And Mapping (SLAM).

The OpenARK SLAM system divides the SLAM problem into two parts: a local tracking system, and a global mapping system. The local mapping system is built on the open-source Visual Inertial Odometry (VIO) system described by Leutenegger et al. [52]. This algorithm estimates the pose of the current camera, the poses of a sliding window of "keyframes", and

the 3d positions of all features observed by these keyframes. A new camera frame is added to the optimization problem as a keyframe any time the current frame has little overlap with the most recent keyframe. This reduces the optimization of redundant information as compared to adding every camera frame as a keyframe. Each time a keyframe is added to the optimization problem the oldest keyframe and the features associated with it are marginalized out and passed to the global mapping system.

The global mapping system keeps all the past keyframes in a single optimization problem. An optimization problem optimizing over the keyframe poses and the positions of all the features observed by these keyframes would quickly become intractable. As a result, in order to keep the optimization problem manageable, only the poses of these keyframes optimized. The feature positions are not optimized by the global mapping system. Instead, as each keyframe is added to the global mapping system the estimated relative transform between the current and previous keyframe is added as a residual to the optimization problem. The features for the current keyframe are converted to a bag-of-words descriptor using the open-source package DBoW2 [19]. Here we have modified DBoW2 to enable the use of BRISK [51] descriptors in order to reuse the descriptors computed by the local tracker. This bag of words descriptor provides an efficient method for comparing images, enabling us to determine if the current keyframe overlaps with a previous keyframe in our map. If an overlapping keyframe is found, we attempt to compute a transform between these two keyframes by matching features across these keyframes. To reduce the dependence on the quality of the estimates from our local tracker, we utilize the depth image provided by the d435i to compute the 3d position of each feature relative to the keyframe. If a transform is successfully estimated we add this estimate as a residual to our optimization problem. This process is called "loop closure". Each time a loop closure is found, we solve the optimization problem using iterative linear approximation (via Ceres Solver [2]). This optimization problem is often called pose graph optimization [70].

This global mapping system utilizes loop closures to correct the errors that are slowly introduced by approximations made by the local tracker. If the time between loop closures is large, the estimated poses of certain keyframes could change drastically when a loop closure is found. This means that any virtual objects placed in the scene using those estimates would have an incorrect pose. To account for this, every virtual object is tied to a keyframe, and the objects pose is specified relative to that keyframe's pose. In this way, every time a keyframe is found and the pose graph is optimized, the virtual object's pose is also updated. An example of how the localizaton and mapping system enables virtual objects to appear physically present from different viewpoints is shown in Figure 1.1.

Unfortunately, any visual mapping system is likely to have failure cases such as scenes with little visible texture or many repeated features. This can lead to a very disorienting experience as virtual objects appear to drift from their assigned location due to incorrect pose estimates. To deal with this we introduce a feature to detect these cases and reset the estimator. This feature is added directly into the local tracker to enable immediate detection of these failure modes. Our system enables detection of failures due to lack of visual texture, repeated features, and other failure cases. When the local tracker is reset, we must also begin

Figure 1.1: A visualization of the a virtual cube projected onto a camera image using the estimated camera position from OpenARK SLAM

a new global map as we can no longer trust our estimated location relative to the previous map. In doing this, we also save the previous global map. Each time a new keyframe is introduced, we then check for overlapping keyframes, not just in the current global map, but in the previous global maps as well. If a transform can be found relative to a keyframe in a previous global map, this keyframe is added to that map, and the system is able to proceed as if no failure had occurred. While this procedure makes failure cases more bearable to the user, the reduction of failure cases completely remains an open problem that this thesis will begin to address.

1.4 OpenARK Reconstruction

In addition to estimating the pose of the augmented reality device, determining the 3D shape of the environment plays a big role in making virtual objects appear present in the physical scene. For example, mapping surfaces in the environment in 3D can enable us to compute realistic occlusions and enable virtual objects to interact with the mapped environment.

To this end, OpenARK follows the method introduced as KinectFusion [36] in order to map the environment using a Truncated Signed Distance Field (TSDF) representation. Here we utilize the output of the OpenARK SLAM system in order to obtain a more accurate pose estimate than would be possible with the KinectFusion approach. The original paper utilized a voxel-grid memory representation for the TSDF. This approach only allows very small scenes to be represented at high resolution, even on modern devices. Several approaches have since been introduced to reduce the memory usage of TSDF maps such at oct-tree [32] and hash-map [68] representations. To this end, OpenARK utilizes the open-source package Open3d [111] which utilizes a hash map representation of the TSDF and also provides a marching cubes [61] implementation for computing a mesh from the TSDF.

While the hash map representation drastically reduces the size of the map relative the

the voxel grid, there does still come a point where keeping the entire TSDF in memory becomes intractable. Further, we run into a similar problem as described in the previous section. If we progress for a long period of time before a loop closure, errors will accumulate and make the estimated TSDF incorrect. Utilizing the loop closure to optimize the TSDF remains a computational challenge. While some have introduced methods for optimizing the mesh directly, such as the work of Whelan et al. [95], we utilize a simpler method that will address both the loop closure and growing memory issues together. Instead of computing a single TSDF representation of the entire environment, we compute a representation of only the environment in a block around the device's current location. Whenever the device leaves this block a new block is generated and the old block is saved as a mesh. Similar to how virtual objects were treated in the previous section, each mesh is associated with a specific keyframe and has its pose specified relative to that keyframe. In this way every time the keyframe pose is updated, the mesh position is updated as well. This leads to a constant memory representation of the TSDF at runtime, enabling us to map very large scenes. As we no longer optimize the TSDF inside a block we have left behind, there will be some error in the corresponding mesh that cannot be optimized out. As long as the local tracking error is small relative to the size of the block, the errors in the corresponding mesh will be small.

A downside of this mesh partitioning is that components of the mesh may be duplicated many times on disk if a location of the scene is visited repeatedly. To resolve this, and enable the most accurate mesh estimate, we save the color and depth images corresponding to each keyframe, as well as the most optimized pose for each keyframe. After the session is complete, the mesh can then be recomputed offline at a higher resolution using the optimized keyframe poses. We believe this represents a practical compromise enabling both fixed memory online reconstruction and accurate high resolution offline reconstruction. An example mesh computed by this method is shown in Figure 1.2.

If we look towards a future where AR is ubiquitous, we may run into issues when trying to simply represent the environment as a mesh created from a TSDF. Specifically, accurately representing the world using this type of mesh requires a large amount of data that may become intractable to share across users. Instead we may look towards more compact representations, specifically those that provide strong priors on the structure of the environment, to reduce the amount of data required to represent the world. One such structure is the image wireframe [35, 113, 114]. In this thesis we will address some challenges in moving towards using wireframes for Augmented Reality.

1.5 OpenARK Hand Tracking

While 3D reconstruction enables virtual objects to interact with the physical environment, hand tracking represents a common method of enabling the user to interact with the virtual objects. While many hand tracking methods exist [15, 55], OpenARK proceeds with the relatively simple method described by Zhou et al. [108]. This method utilizes the depth image provided by the d435i to simplify the hand tracking problem. This method proceeds

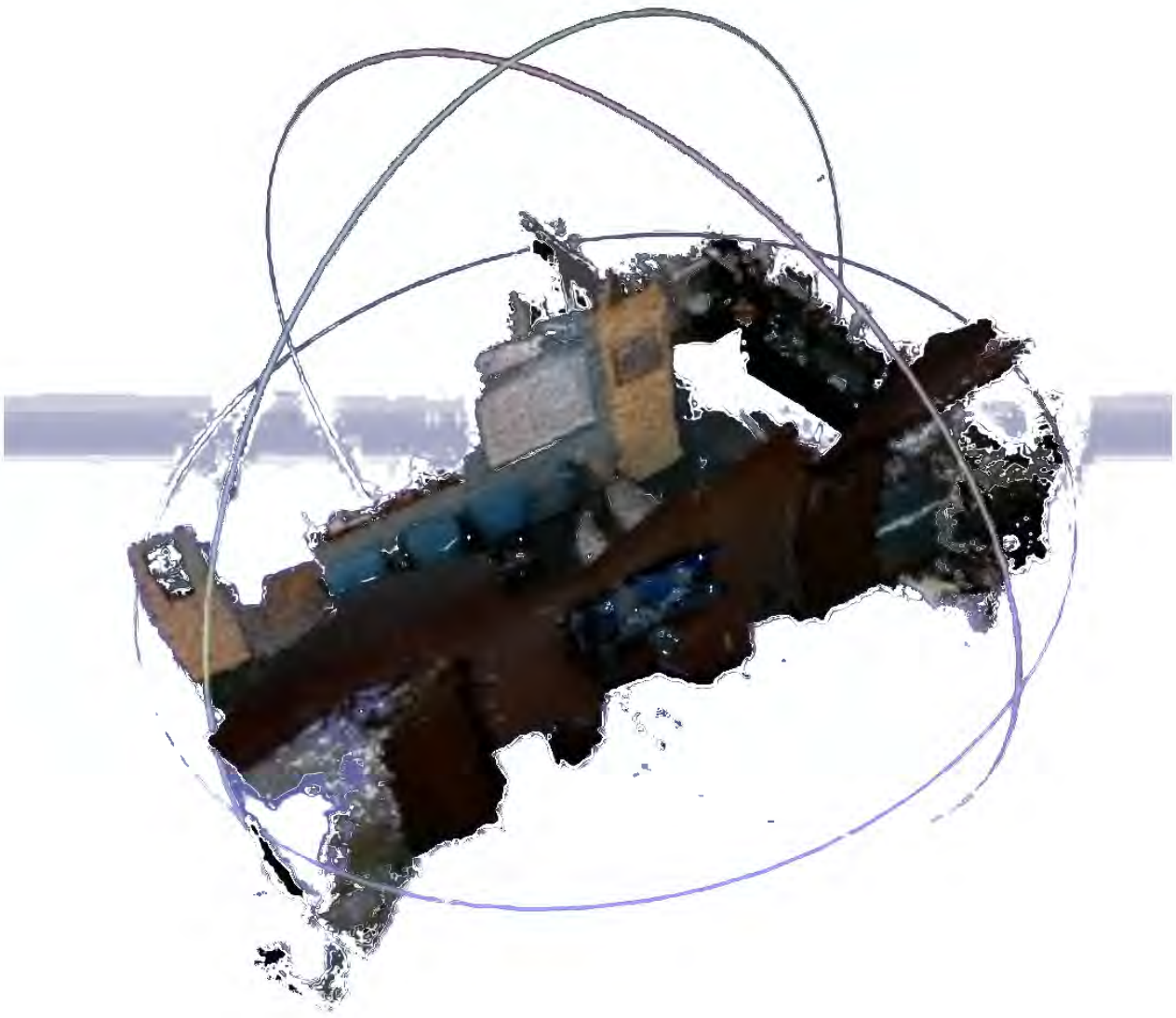


Figure 1.2: A example colorized 3D Mesh generated by OpenARK Reconstruction

in two steps: hand detection and fingertip classification.

The hand detection method proceeds by first detecting and removing planar objects from the scene. This has two purposes. First it removes many of the pixels from the scene from consideration as part of the hand. Second it enables us, later in the pipeline, to detect when the hand is interacting with a planar surface, and allow us to use this surface like a touch screen. Next the remaining pixels in the depth image are clustered using a flood-fill algorithm. A classifier is run on each pixel cluster to determine if the cluster corresponds to

the user's hand. This classifier is based on a set of heuristics that exploit a known device orientation relative to the user, followed by a Support Vector Machine classification.

Once the hand is detected, an algorithm is run on the hand to detect visible fingertips and their locations. For this process, we compute the 2D contour of the hand and find concavities in this contour. A series of simple heuristics determines if each concavity is adjacent to a finger. If a concavity is determined to be finger adjacent, the fingertip position is determined as the maximum value of this concavity. This 2D fingertip position is then projected into 3D using the observed depth. Example detection results are shown in Figure 1.3.

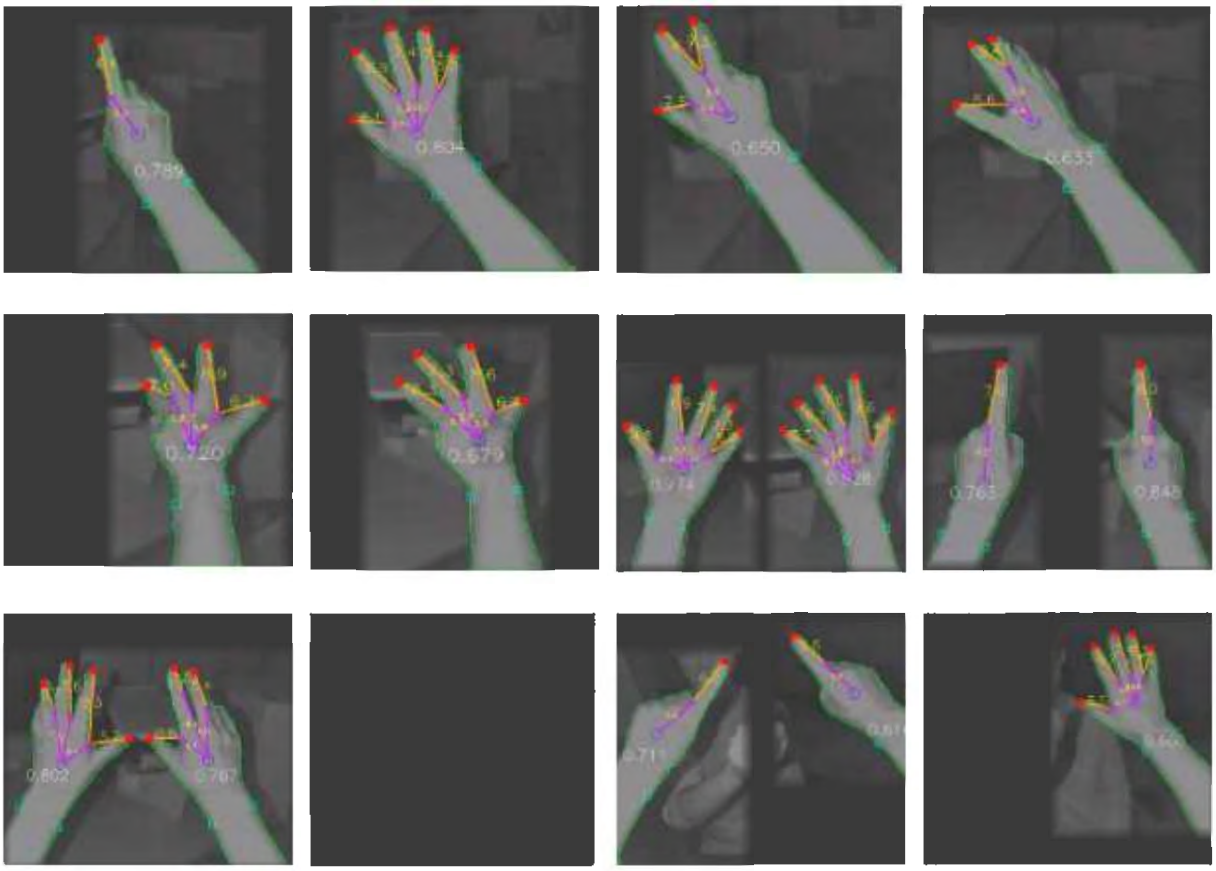


Figure 1.3: Example fingertip detection results generated by OpenARK Hand Tracking [108]

The relative 3D fingertip positions can then be combined with the OpenARK SLAM system to compute the positions of the fingertips relative to any virtual object in the scene. This enables the use of virtual buttons, touch gestures on surfaces, and much more. An example application built using OpenARK is shown in Figure 1.4.

The result of this algorithm is a fast and simple approach to hand and fingertip detection.

Figure 1.4: Example application built using OpenARK Hand Tracking. The user is able to scale the virtual object by touching the white dots and moving their fingers closer together or further apart

In addition, the choice to detect fingertips enables very familiar gestures from the user, such as the swipe gestures used on typical smartphones or track pads. There do remain challenges however. Particularly, because the hands are often used relatively close to the device, they can take up much of the camera image. This can lead the SLAM system to mistakenly use features detected on the hands for estimation, leading to poor results. A similar form of this problem is present in a variety of scenarios such as navigating a crowded room. To reach ubiquitous adoption, our SLAM system must be able to determine which features represent

useful static objects in the scene. By utilizing a wireframe detection network for feature matching, we hope to be able to address these issues.

1.6 Conclusion

While these systems work well in many cases, there still remain several challenges to achieve ubiquitous Augmented Reality. Most commercial augmented reality systems have some sort of Simultaneous Localization And Mapping "reset" feature similar to (though potentially more complex than) the system described in this chapter. These reset systems are designed around detecting situations where SLAM is unable to perform accurately, or situations where the approximations made by the SLAM system result in a poor estimate. These situations include, among others, low light and low texture environments, repeated image features, and varied lighting conditions. The remainder of this thesis will be directed at addressing some of these challenges.

Chapter 2

Planar Structures

2.1 Introduction

Real-time motion estimation on mobile devices has remained a popular topic thanks to the rise of augmented reality (AR). Google's Tango and ARCore, and Apple's ARKit have validated the use of Visual Inertial Odometry (VIO) for 3D localization of a mobile device in real-time, providing 6 degree-of-freedom (6-DoF) position and orientation (pose) relative to a gravity aligned frame. We refer to 6-DoF tracking as the process by which motion states (including 6-DoF poses) are generated and served for a real-time constrained use-case, such as rendering an AR object.

A typical VIO system obtains visual measurements from one or more cameras perceiving only pixel irradiance and measurements of linear acceleration and rotational velocity from an inertial measurement unit (IMU). In addition to the poses of a mobile device over time, a 6-DoF tracking process could reconstruct point clouds from the extracted visual features from surrounding environments, which allows camera-based images to be augmented to display virtual objects in physical world. As a well-known example, the multi-state constraint kalman filter (MSCKF) has demonstrated real-time 6-DoF tracking capability with high precision [65], even under the adversary conditions such as limited computing power, rolling shutter cameras, coarse calibrations, and low-cost IMUs [26] that are typically found on consumer-grade mobile devices.

However, a 6-DoF tracking process relying on point-based visual features is unable to provide accurate estimates of the motion states when the surrounding environments contain little visual texture, see Fig. 2.1. This is because these systems depend on multiple observations of the detectable and reliable visual features. Without these visual features as measurements, the motion states could not be corrected by observation update. There has been significant research into structural features that can be tracked across the multiple scans of a sensor to complement point-based visual features, such as line observations [46, 104], 3D curves [69, 103], small image patches [86, 107], and planes [18]. Engel *et. al.* [14] used direct image alignment to generate consistent map for feature-less SLAM. Of the

available structural features, planes are commonly used due to their well-defined models and prevalence in man-made environments. Therefore, a VIO system can adopt both visual and plane measurements to correct the drift of the 6-DoF poses over time.

Many research groups previously adopted active depth sensing techniques to enable device localization and 3D reconstruction in low-texture environments [36, 96, 105, 31]. The active depth sensing techniques currently available for a practical application include structured light [4], coded light [93], ToF depth sensor [48], and LIDAR sensor [53].

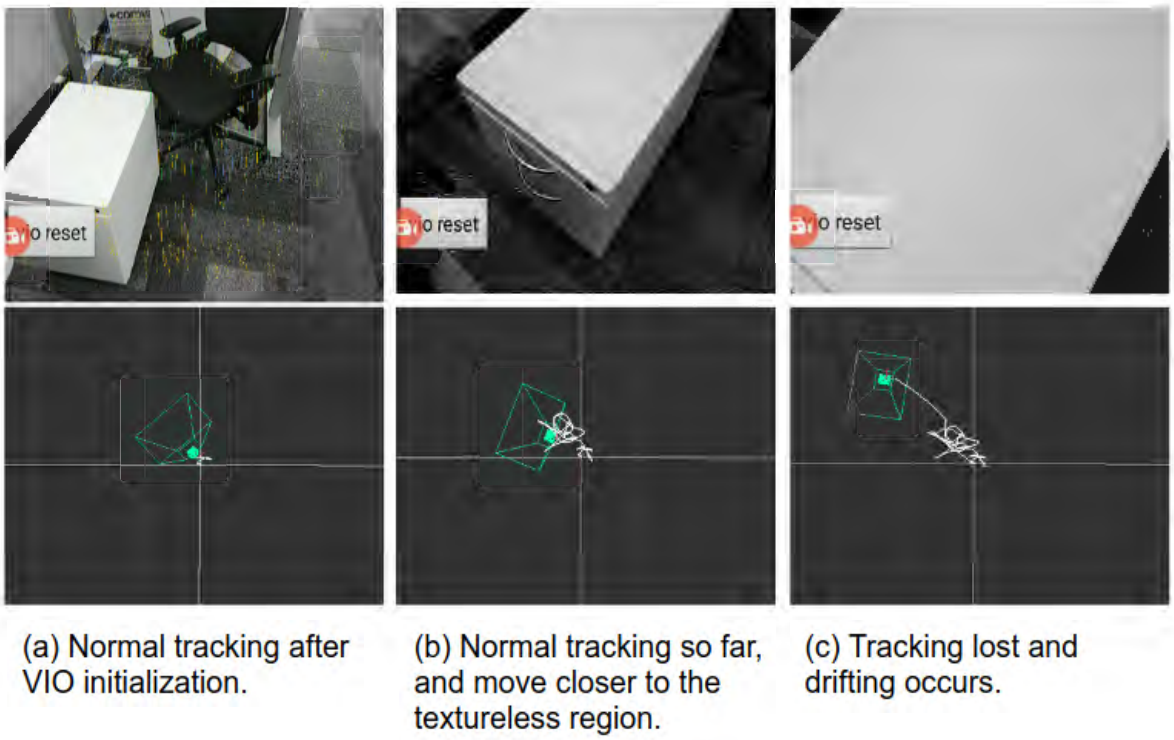


Figure 2.1: An example of 6-DoF tracking in a low-texture environment. The top windows show the captured image and the bottom windows show the corresponding 6-DoF tracking trajectory. (a) VIO is initialized and tracking in a normally textured environment; (b) healthy 6-DoF tracking for a while and the device moves closer to the textureless region; (c) tracking lost and drifts because there are not enough visual features when the device is still facing the low-texture region and no valid 6-DoF poses are output.

Recently many smartphones have been equipped with ToF depth sensors. Compared with dedicated depth sensors in robotics applications [88, 17, 29, 10], the sensing range of these sensors are typically limited. In addition, the depth image from the ToF depth sensor and gray-scaled image from the camera are typically obtained at different time instants, and an estimation algorithm of 6-DoF tracking process has to consider this effect while maintaining real-time efficiency.

The main contributions of this chapter include: (1) An analytical error model is proposed to encode the ToF planes into structural measurements for motion state estimation. (2) It proposes a generic and effective solution of seamlessly integrating plane measurements into an MSCKF-based VIO system. (3) The proposed solution maintains real-time performance of the 6-DoF tracking on consumer-grade mobile devices with limited ToF depth sensing range and computing resource. Given these characteristics our proposed solution is able to effectively improve 6-DoF tracking even in low-texture environments. Fig. 2.2 illustrates the flowchart of data processing in our system.

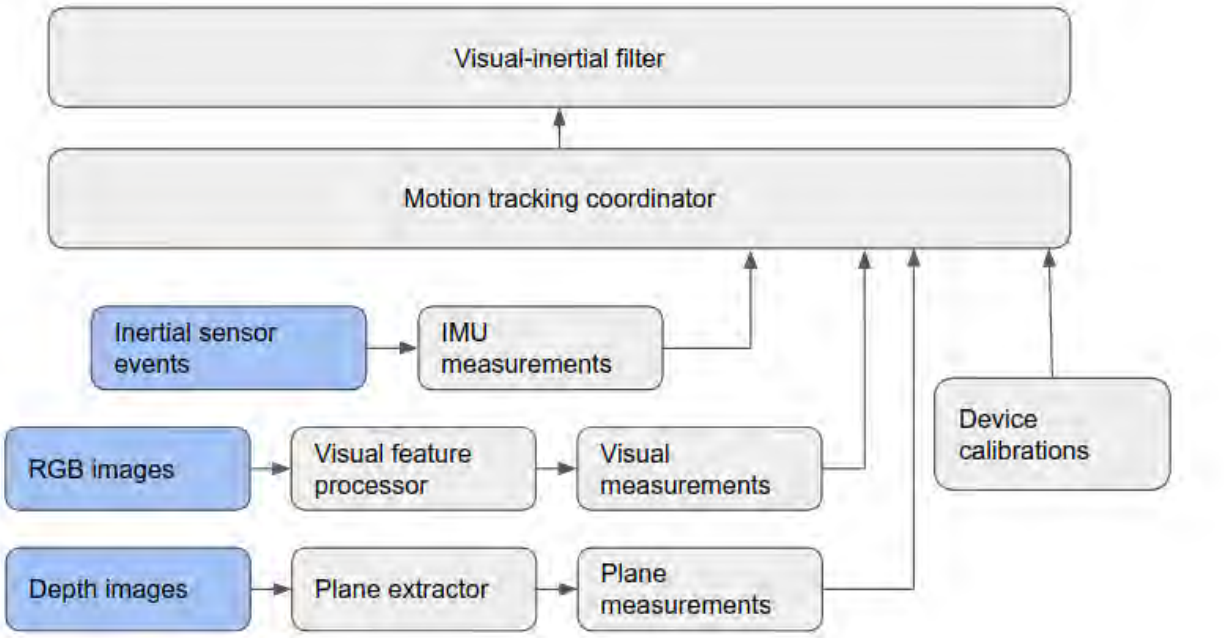


Figure 2.2: A flowchart of our VIO system. The motion tracking coordinator schedules the receiving and processing of the sensor data (IMUs, RGB images and depth images), and the visual-inertial filter updates the motion states by the visual and structural (plane) measurements.

2.2 Related Work

Planar structure based localization has been widely studied in previous research work. Salas-Moreno *et. al.* [16] showed that planes could provide a smaller map representation than occupancy grid methods but improve the overall performance. It was noted that both systems had degraded performance when the depth image did not provide full 3D constraints. Sun *et. al.* [84] used planes to compute constraints to estimate transformations between successive frames. The system fell back on non-planar features when the involved planes were not able

to provide full constraints, thus it still did not fully constrain the system. Similarly, Pathak *et al.* [72] described a closed-form solution to transform motion state estimates between frames using planes and reverted to wheel odometry estimates when the system was not fully constrained. Due to the short sensing range of the ToF depth sensors in our work, the VIO system will not be fully constrained in many scenarios and motion estimation by using depth sensor alone is an impractical solution.

Weingarten and Seigwart [94] proposed an EKF-SLAM method using wheel odometry and laser scanners. Their paper noted that a plane should have only 3 degrees of freedom, although it was often modelled by more parameters to avoid singularities. Their formulation proposed a model with symmetries and perturbations, which allowed a 7 parameter plane state parameterization along with a 3x7 binding matrix for computing the minimal parameterization. In contrast, our solution utilizes the parameterization from Förstner and Khoshelham [18], and applies it to an EKF formulation. This formulation only keeps a 4 parameter plane state along with a 3x2 projection matrix for computing the minimal parameterization.

Kaess [40] introduced a unit quaternion parameterization of plane measurements and a quaternion error for the minimal representation. It allowed the gradient based approximation in a graph optimization framework. This work was later applied to the visual and inertial constraints in [33, 34]. Geneva *et al.* [21] introduced a new parameterization denoted as closest point (CP) parameterization, which proved better performance than the quaternion parameterization. This work also utilized a graph-based optimization framework, in which planes were expressed relative to the pose where they were first observed. In their solution, the CP parameterization did not have singularities, but that is not the case in our proposed solution because we employ a framework on the basis of EKF, in which only a few planes are observed at a time and these planes are represented in a global coordinate frame to enable more computational efficiency. The CP parameterization in EKF-based framework has singularities when the plane intersects with the global origin, even though it rarely occurs as presented in Yang *et al.* [101] that did use the CP parameterization [21] in an EKF-based framework without explicit issues. However, our solution still designs a parameterization to employ EKF-based framework while avoiding the singularities altogether. The main difference between our work and [101] lies in two aspects. First our proposed solution is based on the sensors from a smartphone instead of a dedicated depth sensor. Second our solution enables real-time 6-DoF tracking with visual and plane measurements from smartphones.

2.3 Plane Measurement Model

This section will present the plane extraction from a ToF depth image and the error model of the plane measurements.

Plane Detection And Matching

A depth image received from the ToF depth sensor consists of 3D points in its coordinate frame. A RANSAC-based method is used for plane detection from a ToF depth image. For each iteration of the RANSAC, 3 points are selected to fit a plane with the normal vector $\mathbf{n}$ in 3 by 1 in the depth sensor coordinate frame. Next the distance from the plane to the coordinate origin d is computed as:

$$d = -\mathbf{n}^\top \bar{\mathbf{p}} \tag{2.1}$$

where $\bar{\mathbf{p}}$ is a 3 by 1 vector representing the mean of the points used for the plane normal computation. For each remaining point, the distance d_p from the point to the plane is computed as:

$$d_p = \mathbf{n}^\top \mathbf{p} - d \tag{2.2}$$

If d_p is below a threshold, the point is considered as an inlier. This is repeated until a sufficient number of inliers are found or a maximum iteration number is reached. With a sufficient number of inliers, the plane normal is refined by a least square fit of all the inliers. Next we mask out the inliers of this plane and repeat the process for the remaining points until the termination conditions are met.

Each plane measurement from depth image at t_m searches for its observation at the previous depth image at t_{m-1}, through normal, distance and gyro compensation. Only the plane measurements with matched counterparts are used to estimate the motion states.

Error Model of Plane Measurement

For each plane measurement extracted from a depth image and expressed in the local depth sensor coordinate frame, the error model of the plane normal is computed under the two following conditions. Firstly, the plane normal only has 2 degrees of freedom, e.g. a horizontal plane can be rotated by "pitch" and "roll". Secondly, the per-point noise is highly correlated and thus a single point should not be considered as an independent measurement. The per-point noise correlation originates mainly from the ToF depth sensor with multi-path interference errors [37], which could cause similar corruptions across neighboring points.

Then the error of a plane normal $\mathbf{n}$ is parameterized as a combination of the rotations about two orthogonal axes.

$$\mathbf{n} = \mathbf{R}_{\mathbf{e}_1}(\theta_1)\mathbf{R}_{\mathbf{e}_2}(\theta_2)\hat{\mathbf{n}} \tag{2.3}$$

where $\hat{n}$ is the estimated value of the plane normal, $\mathbf{R}_{\mathbf{e}_1}(\theta_1) \in SO(3)$ and $\mathbf{R}_{\mathbf{e}_2}(\theta_2) \in SO(3)$ are rotation matrices representing rotations around the vectors $\mathbf{e}_1 \in \mathbb{R}^3$ and $\mathbf{e}_2 \in \mathbb{R}^3$ by θ_1 and θ_2 respectively. Obviously $\begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & n\end{bmatrix}$ form an orthonormal basis of space $\mathbb{R}^3$ containing the plane.

By applying small angle approximation of given the detected planar basis $\mathbf{e}_1$ and $\mathbf{e}_2$, the above equation can be written as:

$$\mathbf{n} \approx (\mathbf{I}_3 - \theta_1 \lfloor \mathbf{e}_{1\times} \rfloor)(\mathbf{I}_3 - \theta_2 \lfloor \mathbf{e}_{2\times} \rfloor)\hat{\mathbf{n}} \tag{2.4}$$

With small θ_1 and θ_2 values, the higher order terms are ignored without introducing significant error. Also note that $\mathbf{e}_1 \times \mathbf{n} = -\mathbf{e}_2$ and $\mathbf{e}_2 \times \mathbf{n} = \mathbf{e}_1$. Thus the above equation turns to:

$$\mathbf{n} \approx \hat{\mathbf{n}} + \hat{\mathbf{B}}_m \begin{bmatrix} \theta_2 \\ \theta_1 \end{bmatrix} \tag{2.5}$$

where $\hat{\mathbf{B}}_m = \begin{bmatrix} -\mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix}$ is named as the *planar basis*, and will be involved in the modelling of a plane measurement. Then the error of plane normal measurement is approximated as $\hat{\mathbf{B}}_m^\top (\mathbf{n} - \hat{\mathbf{n}})$.

On the condition that $\theta_1 \sim \mathcal{N}(0, \sigma_{\theta_1}^2)$ and $\theta_2 \sim \mathcal{N}(0, \sigma_{\theta_2}^2)$, the values to determine for plane measurements are σ_{θ_1}, σ_{θ_2}, $\mathbf{e}_1$, and $\mathbf{e}_2$. Among multiple options to model $\mathbf{e}_1$ and $\mathbf{e}_2$, we choose the major and minor axis of the points corresponding to the plane, because these two directions enable the best estimate of σ_{θ_1} and σ_{θ_2}. Since the inlier points of a plane are heavily correlated, we can compute the error of the plane by using only two points in addition to the mean point. To reduce the plane error as much as possible, we select the two points with the largest projections along $\mathbf{e}_1$ and $\mathbf{e}_2$ respectively.

Given a point $\mathbf{p}_1 \in \mathbb{R}^3$ which has the largest projection along $\mathbf{e}_1$ of the points in the plane. $\mathbf{p}_1' = \mathbf{p}_1 - \Delta\mathbf{p}_1$ where $\Delta\mathbf{p}_1 \sim \mathcal{N}(0, \Sigma_{p_1})$ and Σ_{p_1} is a diagonal matrix with σ_x^2, σ_y^2, and σ_z^2 at its main diagonal.

The σ_z of the ToF depth sensor is linearly scaled with the depth at a value determined by the camera specification. The σ_x and σ_y are not only scaled linearly with depth, but also depending on the resolution and focal length of the camera.

With the noise of $\mathbf{p}_1$, we can compute its effect on the plane rotation around $\mathbf{p}_2$ by the small angle approximation, as shown in Fig. 2.3. It obtains:

$$\theta_2 = tan(\theta_2) = \frac{\text{distance from plane}}{\text{projection along } e_1} = \frac{\mathbf{n} \cdot \Delta\mathbf{p}_1}{\mathbf{p}_1 \cdot \mathbf{e}_1} \tag{2.6}$$

$$\sigma_{\theta_2}^2 = \frac{\mathbf{n}^\top \Sigma_{p_1} \mathbf{n}}{(\mathbf{p}_1 \cdot \mathbf{e}_1)^2} \tag{2.7}$$

The same method can be used to compute σ_{θ_1} from the point with the largest projection along $\mathbf{e}_2$. Next, we set both σ_{θ_1} and σ_{θ_2} to be the average of the two values. After this operation, plane error depends more on the average radius of its inlier point cloud, less on the choices of the planar basis $\mathbf{e}_1$ and $\mathbf{e}_2$.

To compute the error of the plane distance to the coordinate origin as Eq. 2.1, we still treat the points as correlated with each other and obtain the noise from the mean point. The plane distance depends on the plane normal.

$$\Sigma_{dd} = \begin{bmatrix} \bar{\mathbf{p}}^\top \hat{\mathbf{B}}_m & \mathbf{n}^\top \end{bmatrix} \begin{bmatrix} \Sigma_{nn} & 0_{2\times 3} \\ 0_{3\times 2} & \Sigma_p \end{bmatrix} \begin{bmatrix} \bar{\mathbf{p}}^\top \hat{\mathbf{B}}_m & \mathbf{n}^\top \end{bmatrix}^\top \tag{2.8}$$

where

$$\Sigma_{nn} = \begin{bmatrix} \sigma_{\theta_2}^2 & 0 \\ 0 & \sigma_{\theta_1}^2 \end{bmatrix} \tag{2.9}$$

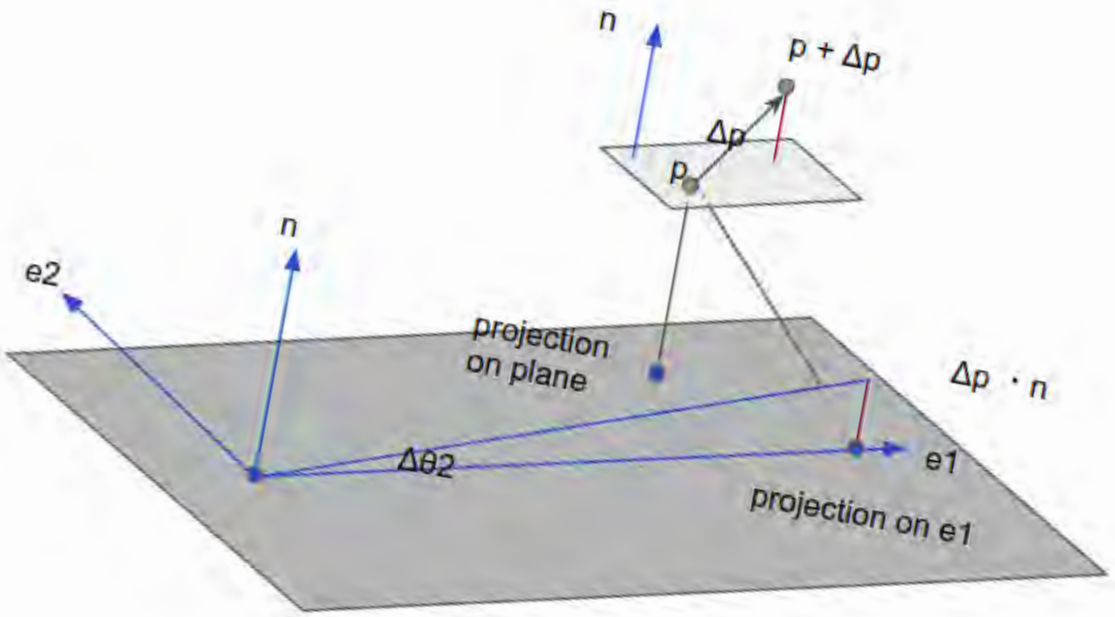


Figure 2.3: The error of plane rotation θ_2 around the axis $\mathbf{e}_2$ is computed from the noise of point $\mathbf{p}$ with a large projection into the axis $\mathbf{e}_1$. Note that the small plane and the large plane are parallel, and the two red lines denote the projection of $\mathbf{\Delta p}$ into the plane normal direction $\mathbf{n}$.

Furthermore the cross-covariance of the plane normal and distance should be computed as:

$$\Sigma_{dn} = \sum (d - \bar{d})(\mathbf{n} - \bar{\mathbf{n}})^\top = -\bar{\mathbf{p}}^\top \hat{\mathbf{B}}_m \Sigma_{nn} \tag{2.10}$$

where the single mean point $\bar{\mathbf{p}}$ is projected to the planar basis $\hat{\mathbf{B}}_m$ before its relation with the plane normal in 2 degrees of freedom. Thus the 3 by 3 covariance of a single plane measurement from depth image is given by:

$$\Sigma_{mm} = \begin{bmatrix} \Sigma_{nn} & \Sigma_{dn}^\top \\ \Sigma_{dn} & \Sigma_{dd} \end{bmatrix} \tag{2.11}$$

2.4 State Estimation with Plane Measurements

The section will introduce the parameterization of a plane in the state vector. It will also relate the plane measurements from the ToF depth sensor with the system states. Note that a system state (or named as *state*) includes the motion states such as 6-DoF poses and

velocities, calibrations such as sensor bias, visual points and planes, as presented in Section 2.5 for more details. This section only describes the plane states and measurements.

Plane State Parameterization

A plane can be denoted by a unit-norm vector $\mathbf{n}$ as normal and a scalar d as the distance from the coordinate origin to this plane, and any 3D point $\mathbf{p}_k$ on the plane leads to $\mathbf{p}_k^T \cdot \mathbf{n} + d = 0$. Specifically all these quantities are expressed with respect to a gravity-aligned global coordinate frame $\{G\}$. Then a single plane state is represented by a 4 by 1 vector:

$$\mathbf{x}_{plane} = \begin{bmatrix} {}^G\mathbf{n}^\top & {}^Gd \end{bmatrix}^\top \tag{2.12}$$

Since plane normal $\mathbf{n}$ is a 3 by 1 vector with unit-norm constraint, the parameterization of plane normal is not minimal, so it cannot be directly applied to an EKF-based filtering framework. To solve this issue, a 2 by 1 error state $\mathbf{n}_e = \begin{bmatrix} k_1 & k_2 \end{bmatrix}^\top$ is introduced and the true plane normal state could be parameterized as the following:

$${}^G\mathbf{n} = {}^G\hat{\mathbf{n}} + \mathbf{B}_p \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \tag{2.13}$$

where ${}^G\hat{\mathbf{n}}$ represents an estimate of ${}^G\mathbf{n}$ and $\mathbf{B}_p$ is a 3 by 2 matrix with the 2 columns perpendicular to ${}^G\hat{\mathbf{n}}$. Note that $\mathbf{B}_p$ is the estimated state of planar basis in the global coordinate frame and evolves into its true state as filter updates, while the $\hat{\mathbf{B}}_m$ defined in Section 2.3 is a planar basis measurement computed from the ToF depth image. Moreover, an additive error is adopted for the state of plane distance. Therefore the error state representation for a plane is given by:

$$\tilde{\mathbf{x}}_{plane} = \begin{bmatrix} \mathbf{n}_e^\top & {}^G\tilde{d} \end{bmatrix}^\top = \begin{bmatrix} k_1 & k_2 & {}^G\tilde{d} \end{bmatrix}^\top \tag{2.14}$$

In the above parameterization, a plane error state has size 3, compatible with the degrees of freedom of a generic plane in $\mathbb{R}^3$, so this parameterization can be applied to the EKF-based filtering framework.

Plane Normal Measurement Model

Motivated by the noise model of plane normal measurement, the following constraints could be established to connect the measured plane normals with the EKF states:

$$\mathbf{z}_n = \mathbf{0} = \hat{\mathbf{B}}_m^\top \; {}^D_I\mathbf{R} \; {}^I_G\mathbf{R} \; {}^G\mathbf{n} + \mathbf{w}_n \tag{2.15}$$

Specifically z_n is the 2 by 1 projection of the measured normal onto $\hat{\mathbf{B}}_m$. $\hat{\mathbf{B}}_m$ represents the planar basis obtained during plane detection from the ToF depth image, ${}^D_I\mathbf{R}$ represents extrinsic between depth sensor coordinate frame $\{D\}$ and IMU coordinate frame $\{I\}$, ${}^I_G\mathbf{R}$ represents the current orientation of the global coordinate frame $\{G\}$ in IMU frame, and

${}^{G}\mathbf{n}$ is plane normal state. The noise of normal measurement is given by $\mathbf{w}_n$ and it is zero mean with covariance given by (2.9). If an estimate of filter states is available, the 2 by 1 projection of the plane normal measurement is estimated as:

$$\hat{\mathbf{z}}_n = \hat{\mathbf{B}}_m^\top \, {}^{D}_{I}\mathbf{R} \, {}^{I}_{G}\hat{\mathbf{R}} \, {}^{G}\hat{\mathbf{n}} \tag{2.16}$$

The residual of plane normal measurement used for state update could be obtained by:

$$\begin{aligned}\mathbf{r}_n = \mathbf{z}_n - \hat{\mathbf{z}}_n &= \hat{\mathbf{B}}_m^\top \, {}^{D}_{I}\mathbf{R} \left({}^{I}_{G}\mathbf{R} \, {}^{G}\mathbf{n} - {}^{I}_{G}\hat{\mathbf{R}} \, {}^{G}\hat{\mathbf{n}} \right) + \mathbf{w}_n \\ &\approx \mathbf{H}^n_{\mathbf{x}_I} \tilde{\mathbf{x}}_I + \mathbf{H}^n_{\mathbf{x}_{plane}} \tilde{\mathbf{x}}_{plane} + \mathbf{w}_n\end{aligned}$$

After linearization, the Jacobians $\mathbf{H}^n_{\mathbf{x}_I}$ and $\mathbf{H}^n_{\mathbf{x}_{plane}}$ relate the residual of plane normal measurement with the error states of current IMU pose $\mathbf{x}_I$ and the error states of plane $\mathbf{x}_{plane}$. The two error states are included in the state vector, so the plane normal measurement can be directly used for EKF update.

Plane Distance Measurement Model

To derive the measurement model for plane distance, we start to derive constraints to satisfy when the proposed normal plus distance parameterization are expressed with respect to different coordinate frame. In the context of our problem, we are interested in constraints between the depth sensor coordinate frame $\{D\}$ and global coordinate frame $\{G\}$ used to represent a plane measurement. If a point $\mathbf{p}_k$ is on a plane defined by $\mathbf{n}$ and d, the two following constraints are satisfied according to plane equations:

$${}^{D}\mathbf{n}^\top \, {}^{D}\mathbf{p}_k + {}^{D}d = {}^{G}\mathbf{n}^\top \, {}^{G}\mathbf{p}_k + {}^{G}d = 0 \tag{2.17}$$

Then the distance expressed in the depth sensor coordinate frame $\{D\}$ could be related to plane state parameterization in global coordinate frame $\{G\}$ and is further simplified to:

$${}^{D}d = {}^{G}d + {}^{G}\mathbf{n}^\top \left({}^{G}\mathbf{p}_I + {}^{I}_{G}\mathbf{R}^T \, {}^{I}\mathbf{p}_D \right) \tag{2.18}$$

Thus the plane distance measurement model is given by:

$$z_d = {}^{G}d + {}^{G}\mathbf{n}^\top \left({}^{G}\mathbf{p}_I + {}^{I}_{G}\mathbf{R}^T \, {}^{I}\mathbf{p}_D \right) + \mathbf{w}_d \tag{2.19}$$

Similar to normal measurement, estimates of plane distance measurement is obtained by:

$$\hat{z}_d = {}^{G}\hat{d} + {}^{G}\hat{\mathbf{n}}^\top \left({}^{G}\hat{\mathbf{p}}_I + {}^{I}_{G}\hat{\mathbf{R}}^T \, {}^{I}\mathbf{p}_D \right) \tag{2.20}$$

Thus the residual for plane distance measurement is given by:

$$\begin{aligned}r_d = z_d - \hat{z}_d &= z_d - {}^{G}\hat{d} + {}^{G}\hat{\mathbf{n}}^\top \left({}^{G}\hat{\mathbf{p}}_I + {}^{I}_{G}\hat{\mathbf{R}}^T \, {}^{I}\mathbf{p}_D \right) \\ &\approx \mathbf{H}^d_{\mathbf{x}_I} \tilde{\mathbf{x}}_I + \mathbf{H}^d_{\mathbf{x}_{plane}} \tilde{\mathbf{x}}_{plane} + \mathbf{w}_d\end{aligned}$$

Similarly to plane normal measurement, after linearization, the Jacobians $\mathbf{H}^d_{\mathbf{x}_I}$ and $\mathbf{H}^d_{\mathbf{x}_{plane}}$ relate the residual of plane distance measurement with the error states of current IMU pose $\mathbf{x}_I$ and the error states of plane $\mathbf{x}_{plane}$. Same as the plane normal, the plane distance measurement can also be directly used for EKF update.

2.5 System Integration

Overview of Estimation Algorithm in VIO System

Our VIO system relies on an MSCKF-based estimator and this section will give an overview of the underlying estimation algorithm of 6-DoF tracking. Our goal is to estimate the 6-DoF poses of a mobile device equipped with an IMU composed of a 3-axis accelerometer and a 3-axis gyroscope, a rolling-shutter (RS) camera and a ToF depth sensor with respect to a gravity-aligned global coordinate frame $\{G\}$. Specifically our framework estimates the following state vector:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_{I_1}^\top & \dots & \mathbf{x}_{I_N}^\top & \mathbf{f}_1^\top & \dots & \mathbf{f}_M^\top & \mathbf{p}_1^\top & \dots & \mathbf{p}_K^\top \end{bmatrix}^\top$$

where $\mathbf{x}_I$ represents the current IMU pose and its definition is given as following:

$$\mathbf{x}_I = \begin{bmatrix} {}^I\mathbf{q}_G^\top & {}^G\mathbf{p}_I^\top & {}^G\mathbf{v}_I^\top & \mathbf{b}_g^\top & \mathbf{b}_a^\top \end{bmatrix}^\top \tag{2.21}$$

The states $\mathbf{x}_{I_i}, i = 1 \dots N$ represent the latest N clones of IMU poses maintained in the sliding window. $\mathbf{f}_j, j = 1 \dots M$ are M visible points as visual features using inverse-depth parameterization. Furthermore, it includes K plane states as parameterized in Section 2.4.

Each received IMU measurement is used to propagate the estimated states and covariance [26]. For the feature points extracted from a received RGB image, a stochastic cloning of IMU pose containing orientation and position will be performed. A difference of Gaussian (DoG) detector is employed to generate the feature points, which are then temporally matched by 2-pt RANSAC[49]. This process produces feature tracks as visual measurements for state estimation. A feature track is processed by the two following approaches according to its track length. When the track length of a feature track is less than or equal to N, the feature track is triangulated into a 3D point from all its observations by Gauss-Newton minimization. After computing the residuals and Jacobians of all the observations and feature states are marginalized and all the observations of the features are used for filter update without maintaining feature states in the state vector. On the other hand, if the track length of a feature track is more than N, the feature track is processed in the same way as above for the first N feature measurements and then initialized into state vector. For any subsequent observation after first N observations it will be processed by using standard EKF-SLAM method. Since the images in our system are obtained from an RS camera, we follow the approach in [26] which proven to be an accurate yet efficient approach to consider RS effect. For mobile device with RS imaging, the RGB image is typically captured row-by-row and

we assume that timestamp of an image is chosen to be the time instant when middle row of the image is captured.

To keep the computational cost of processing feature tracks relatively bounded at each time instant, the total number of features processed by the two above methods is fixed. In addition, a Mahalanobis-gating test is performed before the observation of a feature track is used for filter update. The number of feature points maintained in the state vector is also fixed, and therefore those features that lost tracking are removed from the state. This allows a newly-observed feature with longer track length to be added into the state, and brings in newer constraints for 6-DoF pose estimation. Similarly, IMU poses $\mathbf{x}_{I_i}$ maintained in the sliding window are removed once all its associated feature tracks are consumed for filter update.

Integration of Plane Measurements into VIO System

The aforementioned framework of a VIO system only deals with two sources of sensor data: inertial measurements from IMU and feature tracks from camera. This section will introduce the method to cope with plane measurements obtained from the processing of ToF depth images in the existing filtering framework. Our current work assumes that the camera and ToF depth sensor have pre-synchronized clock. No explicit issues about the clock synchronization were found in our experiments, and the temporal alignment between the two sensors will be performed in future work.

RGB images and depth images are rarely emitted at the exactly same timestamp. Suppose that two consecutive RGB images are captured at timestamps t_k and t_{k+1} and a ToF depth image is captured at timestamp t_m, and the temporal ordering of these images is $t_k \leq t_m \leq t_{k+1}$. These measurements are always processed in the same order as the arrival of the RGB and depth images. Firstly the estimated state and covariance are propagated by using IMU measurements up to time t_m. Instead of performing a stochastic cloning of the states at time t_m as in processing RGB image, an EKF-SLAM style update is performed by using all the available plane measurements. After the state update with plane measurements, it follows the same process as the existing framework, propagating the state vector into the next image for a state update with visual measurements. Our proposed solution introduces a minimal change of the existing framework, and no extra states other than plane normal and distance are added specifically for the plane measurements. Fig. 2.4 illustrates the ordering of receiving and processing the visual and plane measurements. However, one might argue that the aforementioned assumption is not always guaranteed, and it is still possible that the plane measurements at t_m arrive later than the visual measurements at t_{k+1}, especially in a multi-threaded setup. However, according to the experiments, the out-of-order case is not common unless the ToF plane extraction takes a longer time than $\Delta t(m, k+1) = t_{k+1} - t_m$. In those cases, we should just ensure the efficiency of ToF plane extraction.

As described above, all the planes from ToF depth images follow the standard EKF-SLAM process. When a plane is observed for the first time, it is initialized into the state vector using the method in [101]. For any subsequent measurements that match the same

plane, a Mahalanobis-gating test is used to reject the ineligible plane tracks, and then we perform an EKF-SLAM update by the plane measurements. Similar to the visual feature, a plane state is also removed from the state vector once the plane tracking is lost. Given the method of processing plane measurements, the computational complexity of dealing with plane states for the state vector is cubic in the number of planes. However, due to the limited sensing range of the depth sensor on consumer-grade mobile devices such as smartphones, the number of planes to maintain in the state vector is small enough, and the experimental results in Section 2.6 show that our VIO system with plane measurements maintains real-time performance of the 6-DoF tracking on the smartphones.

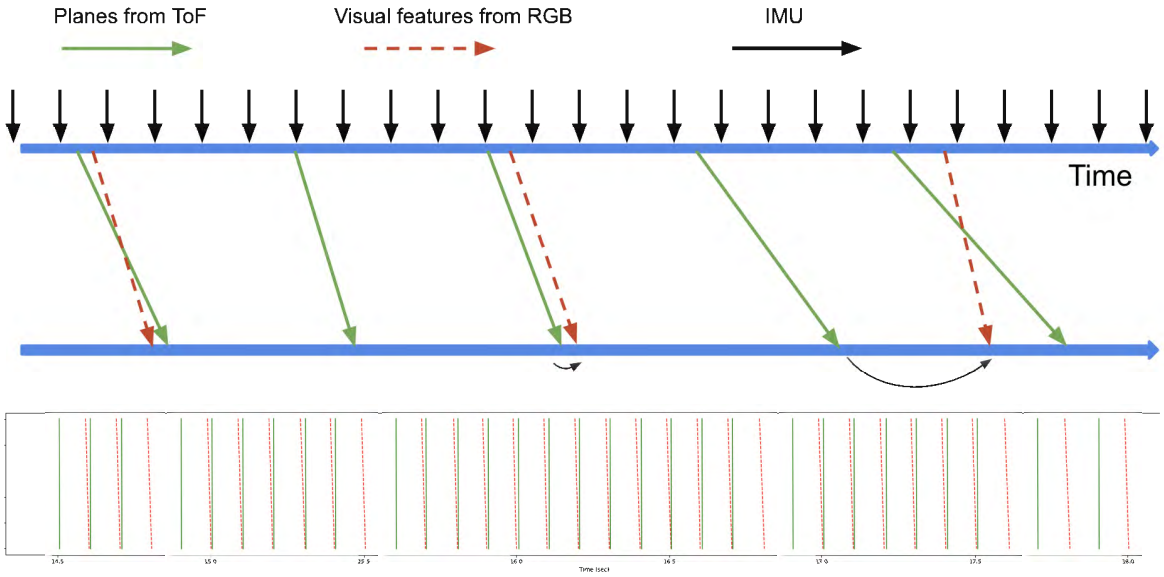


Figure 2.4: Top: the upper axis denotes the time of receiving sensor data, and the bottom axis denotes the time of starting a state update when the measurements computed from the sensor data are ready. The arc arrows denote that a vision-based state update can utilize the most recent 6-DoF pose that is estimated from the state update with plane measurements at a previous ToF depth image. Bottom: a short period of processing sensor data into feature measurements is clipped out of our experiments to depict the relations of RGB images and ToF depth images in practice.

2.6 Experiments

Experimental Setup

To benchmark the proposed solution, sensor data from a cusomer-grade mobile device are recorderd for offline processing. Each collected dataset contains IMU measurements, RGB images and depth images and is called a session below. To evaluate the proposed solution

in low-texture environments, many sessions used in benchmarking are collected with abundant structure planes and scarce point features (e.g white table and wall). Ground truth trajectories of each session is obtained from VICON system [91] and hand-eye calibration is performed so that trajectories from VICON and VIO estimates could be aligned.

Metrics and Experimental Results

The experiments of 6-DoF tracking in low-texture environments demonstrate that tracking lost and drifts frequently occur in the estimated positions without the ToF plane measurements. With the support of ToF depth sensor, our VIO system can obtain more accurate estimates of device positions, as shown in Fig. 2.5.

This figure also depicts no significant changes on the estimated device orientations, because the integration of gyro measurements in our system enables high-quality approximations of device orientations for a limited time, even after the position tracking is lost.

The evaluation of 6-DoF tracking is based on the root mean squared error (RMSE) values between the VIO estimated trajectory and the VICON ground truth trajectory for each of the testing sessions. The two trajectories have been aligned and equally sampled by timestamps before their comparison.

Global position RMSE is defined to compute the position errors of each pair of sample points at the same timestamp from the two trajectories. It evaluates the degree of fitting the two trajectories under the same coordinate frame.

The results in Table 2.1 show that the ToF planes effectively reduce the global position RMSE by more than 80%. The original VIO system suffers from tracking lost and drift in low-texture environments, resulting in unpredictably large errors (e.g. more than 10 meters) in comparison with ground truth.

Local position RMSE is defined to compute the position transformations of consecutive sample points at the same timestamp from the two trajectories. The global position RMSE requires the temporal and 6-DoF spatial alignment of the two trajectories so that they can be compared under the same coordinate frame, while this local position RMSE needs only the temporal alignment. If a trajectory has a large global position RMSE value but a small local position RMSE value, it means that the 6-DoF tracking encountered some drifts but is not totally lost in most of the time, and could be fixed by 3D mapping if in a normally textured environment.

Fig. 2.6 illustrates the example global position RMSE and local position RMSE. The results in Table 2.2 show that the sessions with ToF planes effectively also reduce the local position RMSE by 62.50%.

Our original VIO system has real-time performance on general smartphones. So the computational cost of the plane extraction from ToF depth images and the additional filter update by plane measurements in VIO has to be taken into account. A latency metric is defined as the elapsed time from receiving an image to publishing the 6-DoF pose at the exposure time of that image.

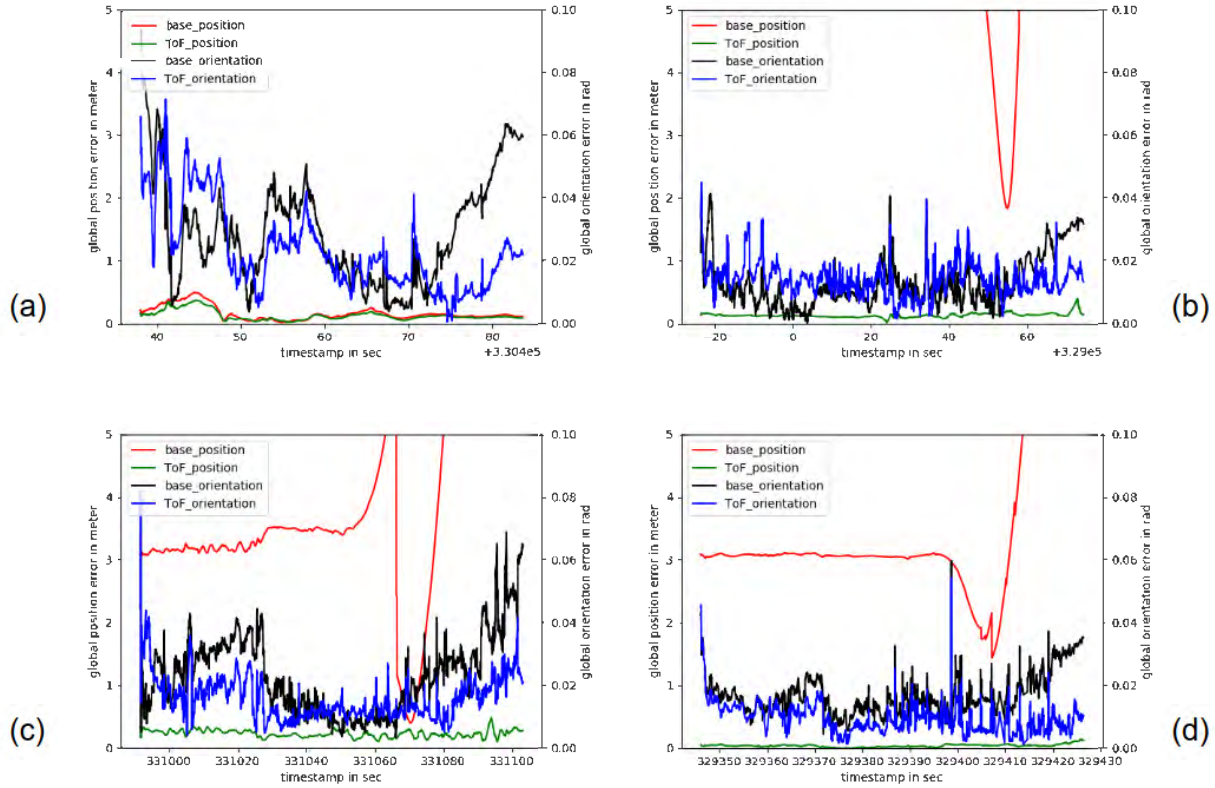


Figure 2.5: The global errors of the 6-DoF poses between VICON and VIO trajectories under the same coordinate frames in each comparison. The session (a) is captured from a normally textured environment with enough visual features, while the sessions (b-d) are captured from low-texture environments. We can observe that the global positions of the base cases in (b-d) could have errors larger than 5 meters, which means the 6-DoF tracking has been drifted and no valid 6-DoF poses are output.

According to the evaluation results in Table 2.3, the consuming time is increased by approximately 17.4%. But the average latency 72.809ms on the testing sessions means that this VIO system can still guarantee the real-time performance on general smartphones.

2.7 Conclusion

This chapter presents an effective solution of VIO degradation in low-texture environments. This system adopts a ToF depth sensor to identify the planar structure, builds error models for plane normal and distance, and applies the plane measurements to the filter update of the system states. This work shows that plane states and measurements can be successfully incorporated into an existing MSCKF framework, and a VIO system processing both visual

Global position RMSE

Session	Original w/o ToF [m]	Improved w ToF [m]	Percentage
01	12.2	2.1	-82.79%
02	16.6	1.52	-90.84%
03	1.1	0.0508	-95.38%
04	2.53	2.84	12.25%
05	11.8	0.266	-97.75%
06	20.8	0.321	-98.46%
07	0.243	0.255	4.94%
08	5.52	1.12	-79.71%
09	0.239	0.195	-18.41%
10	0.708	0.716	1.13%
11	1.56	0.496	-68.21%
overall	6.664	0.898	-86.52%

Table 2.1: The **Global** position RMSE values in meters obtained respectively from the original VIO without using ToF plane measurements and the improved VIO with ToF plane measurements. Note that each session runs for at least 20 times to avoid the influence of randomness in our VIO system, and the "overall" comes from the mean value of all the runs of all the sessions.

and plane feature measurements can keep the real-time performance of 6-DoF tracking on smartphones equipped with a ToF depth sensor.

In the future, the ToF depth sensors will be finely calibrated instead of using the default parameters. Plane measurements will also be applied to VIO initialization, so that this system does not rely on a successfully initialized VIO system from a normally textured environment. To alleviate the extra latency resulting from plane extraction, the VIO system should be able to automatically perceive the surrounding textureness and enable the on-demand depth sensor in low-texture environments.

Local position RMSE

Session	Original w/o ToF [m]	Improved w ToF [m]	Percentage
01	1.93	0.273	-85.85%
02	2.06	0.141	-93.16%
03	0.246	0.03	-87.80%
04	2.74	2.99	9.12%
05	1.26	0.125	-90.08%
06	4.11	0.124	-96.98%
07	0.429	0.445	3.73%
08	0.965	0.432	-55.23%
09	0.12	0.11	-8.33%
10	0.761	0.768	0.92%
11	0.367	0.183	-50.14%
overall	1.362	0.511	-62.50%

Table 2.2: The **Local** position RMSE values in meters obtained respectively from the original VIO without ToF plane measurements and the improved VIO with ToF plane measurements. Note that each session runs for at least 20 times to avoid the influence of randomness in our VIO system, and the "overall" comes from the mean value of all the runs of all the sessions.

Latency

Session	Original w/o ToF [ms]	Improved w ToF [ms]	Percentage
01	60.5	71.3	17.85%
02	59.6	71.7	20.30%
03	63.3	75.1	18.64%
04	60.5	75.5	24.79%
05	62.4	75.9	21.63%
06	61.7	75.2	21.88%
07	57.5	66.6	15.83%
08	61.4	69.6	13.36%
09	63.2	71.1	12.50%
10	66.9	74.9	11.96%
11	65.2	74.0	13.50%
overall	62.018	72.809	17.4%

Table 2.3: The latency in milliseconds from receiving an image to publishing the 6-DoF pose at the timestamp of that image. Note that each session runs for at least 20 times to avoid the influence of randomness in our VIO system, and the "overall" comes from the mean value of all the runs of all the sessions.

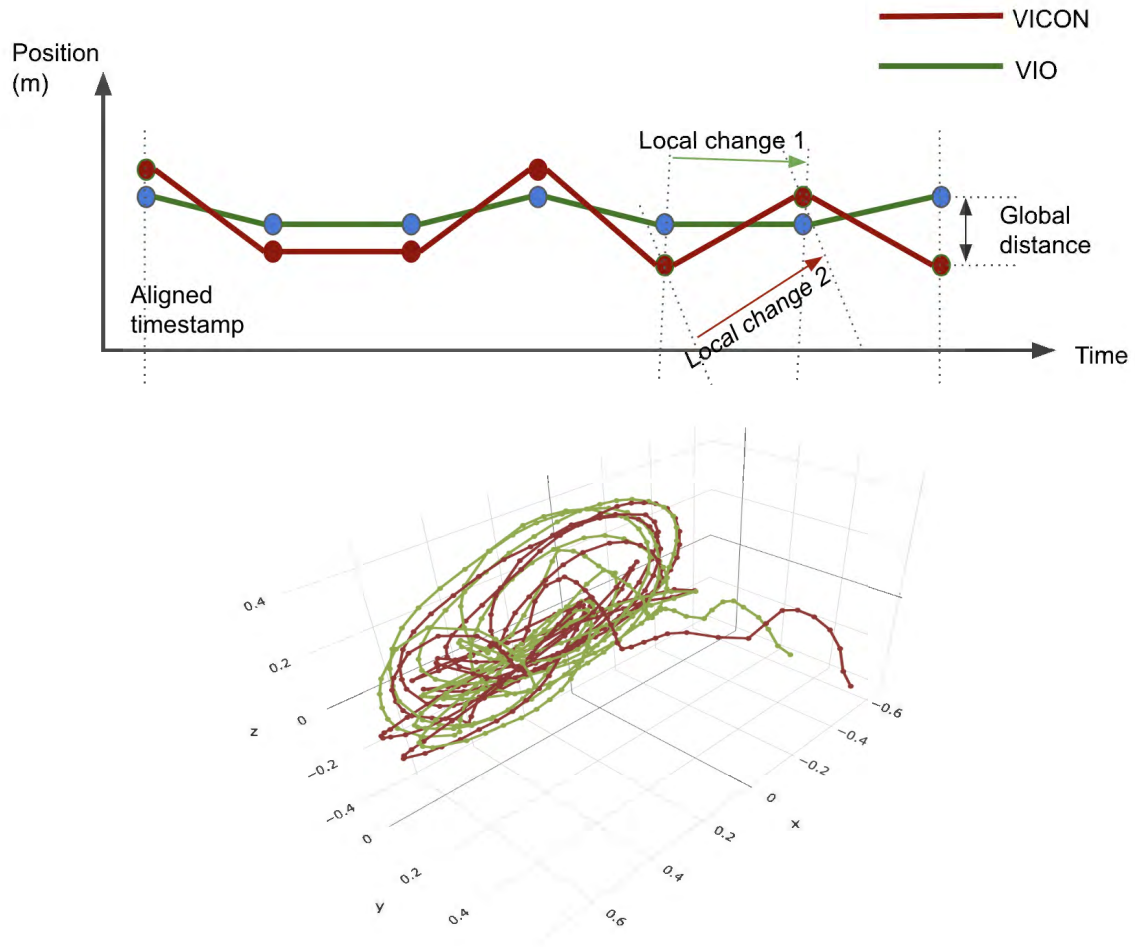


Figure 2.6: Top: the illustration of the global position RMSE and local position RMSE metrics in our evaluations. Bottom: an example pair of VICON and VIO trajectories from a practical session.

Chapter 3

Wireframe Matching

3.1 Introduction

3D reconstruction is an important task for many computer vision applications such as robotics and Augmented Reality. While computing power continues to increase, the burden of collecting data remains a fixed cost for the user. As such, many are looking towards reducing the data required at runtime to generate accurate reconstructions. One method to achieve this is to use large amounts of data, collected ahead of time, to provide a strong prior on how data collected at runtime should be interpreted [56, 27, 58, 57, 114]. In the extreme case this allows depth reconstruction from a single monocular image [22, 78, 77]. However these approaches can be fragile to minor changes in camera parameters such as angle or focal length [13, 28] and overcoming these issues may require an intractable amount of data. Instead we may look to spend additional compute cycles to ensure we make the most out of the data we collect at runtime, thereby reducing the need to collect redundant data.

Due to the inherent ambiguity in visual features, a significant amount of information is discarded due to an inability to accurately associate these features across images. Graph matching has been shown to be successful as a method for comparing images [80, 12, 59] or matching features in images [89], however, due to the high compute cost, it has only been applied in cases with relatively few (≈ 40) nodes. If realtime performance can be achieved, this potentially provides us with a better way to associate structures in an image, and reduce the amount of information lost.

In this chapter we combine the above approaches, utilizing a neural network trained to identify important structures (ie. wireframes) in the scene [113], and then apply Graduated Assignment (GA) graph matching [23] to match these structures between images, achieving improved performance over baseline feature matching. We further show that our simplification of GA is able to achieve realtime performance (30 FPS) on graphs with up to 300 nodes without a loss in accuracy compared to GA. We show that this approach can also be applied to achieve improved performance in standard feature matching without the use of a wireframe.

3.2 Related Works

Wireframe Detection

A wireframe consists of a set of straight lines and their intersected junctions and is analogous to line drawings commonly used in architecture design [35]. These lines and junctions represent some of the most fundamental elements that can be used to infer the geometric structure of the scene. They are also a compact representation, using far less information to describe the scene than point clouds or occupancy grids.

Wireframes can be detected using a simple line segment detector [92] and the junctions inferred through some simple heuristics [74]. Huang et al. [35] showed that these wireframes can be more accurately detected using a neural network trained on a well labeled dataset. Zhou et al. [113] further improved these results by switching to an end to end parsing scheme. We utilize the publicly available network from Zhou et al. [113] in our results going forward, however our method for wireframe matching is not dependent on the particular method of wireframe extraction used. An example wireframe detection is shown in Figure 3.1. Notice that the wireframe typically involves a sparse set of connections between junctions. That is, a given junction only has a few lines associated with it and is only directly connected with a small number of other junctions.

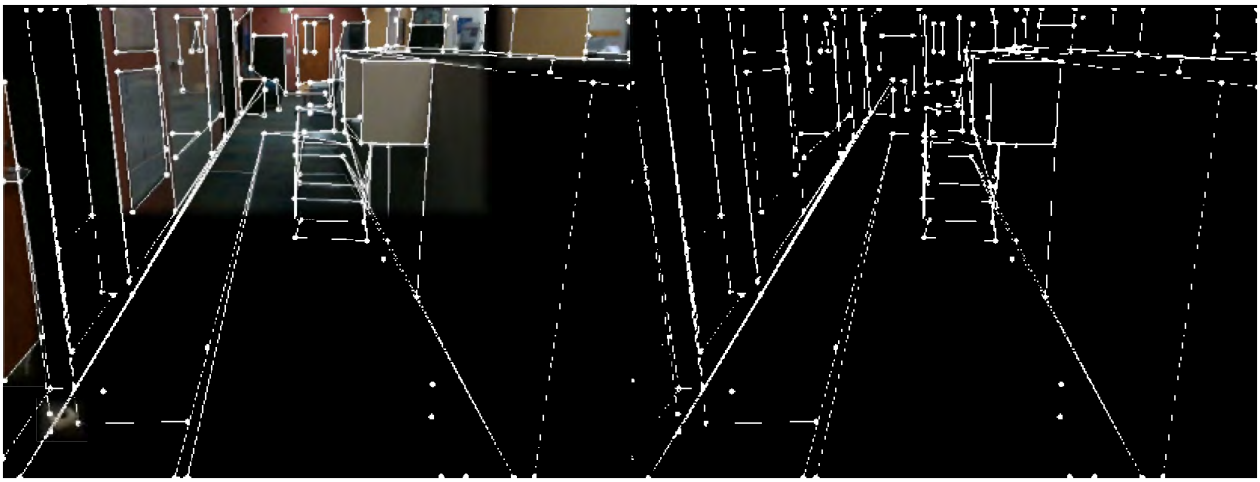


Figure 3.1: Example wireframe detection result overlayed on the RGB image (left) and overlayed on a black image (right).

Graph Matching

As we discuss later, a wireframe can be considered a graph linking visual features in an image. Therefore, it is reasonable to consider wireframe matching in images as a special case of graph matching. Graph matching is a very difficult problem with many applications.

Several different methods are employed that can achieve approximate solutions to the general problem. The most basic of these is the Hungarian algorithm [47] which solves the assignment problem given a measure of similarity of a variable to each assignment. This has been applied by several groups to give an approximate solution to the graph matching problem under various measures of similarity between nodes [39, 1, 38, 79].

While these algorithms can work well given a good similarity metric, the majority of these algorithms are "local" algorithms meaning they can only compare the similarity of a region around a given node. This has issues if there are repeated structures in the graph. More advanced approaches looks to solve an optimization problem (shown in Section 3.4) that maximizes the overall coherence of the graphs [23, 11, 110, 109, 50, 60, 89, 3]. Of these, the Graduated Assignment algorithm (GA) [23] has been used extensively in computer vision especially in matching image skeletons [80, 59, 12].

More recently, several algorithms have been shown to outperform GA [11, 110, 109, 60, 89]. Unfortunately these papers utilize an open source implementation [11] of GA that does not have good performance under the parameters defined in the original paper. Specifically, the open-source implementation performs a scaling on the similarity matrix that causes the algorithm to take very small steps at each iteration. While we cannot re-implement all the above algorithms, removing this scaling causes significant improvement to the GA algorithm for tests made by Zhou and De la Torre [109], leaving it on par with the best algorithms evaluated in the paper.

Feature Matching

The matching of wireframes can also be considered a special case of standard feature matching where the visual similarity of either the lines or junctions is used to match these features across images. Many methods exist that look at robustly describing the image patch around the feature in a way that can be easily compared across images [9, 51, 76, 62, 92, 106]. These approaches establish a distance metric between features, such as Hamming distance for binary descriptors [9]. A simple brute force matching approach is to match each feature with its closest feature under this metric. This is what we term "standard feature matching" in this chapter. While these methods work very well, these features suffer greatly in the presence of repeated textures; a common occurrence in man-made environments.

Similar in spirit to our approach is that of graph cuts [7] which can be used to efficiently find a labeling of pixels that minimizes an energy function. This has a wide variety of applications in the field of computer vision, but most relevant here is the application to stereo feature matching [43] and multi-view feature matching [44]. Rather than matching sparse features directly, these methods attempt to assign a depth or disparity label to each pixel. This requires a known transformation between images, which we may not have at the time of matching. It also requires a fixed number of disparity labels limiting the resolution of the reconstruction. Note that, while graph cuts can solve a wide variety of energy minimization problems, known as submodular functions [45], graph matching does not fall into this category [89].

3.3 Contribution

The contributions of this chapter are as follows:

1. We propose a simplification of the Graduated Assignment algorithm that, for certain tasks, can achieve realtime performance without a significant reduction in accuracy. Tasks involving 3D models, such as path planning and obstacle avoidance, often require realtime performance but, until now, this type of matching algorithm has been outside realtime performance for many scenarios.

2. We propose a novel wireframe matching algorithm using the simplified Graduated Assignment that utilizes the visual similarity of junctions and lines. This represents an important step towards automatic 3D wireframe reconstruction, which will enable compact models that can be used in robotics and augmented reality.

3. We demonstrate the applicability of the simplified Graduated Assignment algorithm to realtime feature matching. This allows the simplified Graduated Assignment algorithm to be applied, and increase the performance of, a wide variety of tasks involving feature matching such as simultaneous localization and mapping and 3d object tracking.

3.4 Problem Statement

A wireframe can easily be represented as a graph, where each junction represents a node of the graph, and each line represents an edge. The converse is also an option, where each line represents a node and each junction an edge. In either case, to optimize matching of wireframes images, we need to define the metric to optimize over. The simplest metric is to maximize matching nodes whose connected nodes are also connected in the corresponding image. This is a standard graph matching problem:

$$\max_{\mathbf{M}} S^{\mathrm{gm}}(\mathbf{M}) = \sum_{a=1}^{{}^1n} \sum_{i=1}^{{}^2n} \sum_{b=a+1}^{{}^1n} \sum_{j=i+1}^{{}^2n} \mathbf{C}_{abij} \mathbf{M}_{ai} \mathbf{M}_{bj}, \tag{3.1}$$

$$\text{s.t. } \forall a \sum_{i=1}^{{}^2n} \mathbf{M}_{ai} \leq 1, \forall i \sum_{a=1}^{{}^1n} \mathbf{M}_{ai} \leq 1, \forall ai\, \mathbf{M}_{ai} \in \{0,1\},$$

where S^{gm} is the similarity function we would like to maximize. If we denote the two graphs ${}^1\mathbf{G}$ and ${}^2\mathbf{G}$ then ${}^1n$ and ${}^2n$ are the number of nodes in ${}^1\mathbf{G}$ and ${}^2\mathbf{G}$ respectively. We can represent ${}^1\mathbf{G} \in \mathbb{R}^{{}^1n \times {}^1n}$ and ${}^2\mathbf{G} \in \mathbb{R}^{{}^2n \times {}^2n}$ as sparse matrices where ${}^1\mathbf{G}_{ab} = 1$ if node a is connected to node b and 0 otherwise. ${}^2\mathbf{G}_{ij}$ follows similarly. $\mathbf{C}_{abij}$ represents the compatibility of these edges. That is:

$$\mathbf{C}_{abij} = \begin{cases} 1 & {}^1\mathbf{G}_{ab}{}^2\mathbf{G}_{ij} = 1 \\ 0 & otherwise \end{cases}$$

The Matrix $\mathbf{M} \in \mathbb{R}^{{}^1n \times {}^2n}$ represents the correspondence of nodes, with $\mathbf{M}_{ai} = 1$ indicating the matching of node a in ${}^1\mathbf{G}$ to node i in ${}^2\mathbf{G}$. The inequality constraints in this optimization enforce that a node in ${}^1\mathbf{G}$ can only be matched to at most one node in ${}^2\mathbf{G}$. It is important to note here that the matrix C need not be computed directly. If ${}^1\mathbf{G}$ and ${}^2\mathbf{G}$ are represented as sparse matrixes, the similarity function S^{gm} can be computed very efficiently in $O({}^1l^2l)$ time (where ${}^1l$ and ${}^2l$ are the number of edges in ${}^1\mathbf{G}$ and ${}^2\mathbf{G}$ respectively) by iterating through the non-zero elements of ${}^1\mathbf{G}$ and ${}^2\mathbf{G}$.

An alternative method to the above would be to additionally maximize the visual similarity of the nodes:

$$\max_{\mathbf{M}} S^{\text{agm}}(\mathbf{M}) = \sum_{a=1}^{{}^1n} \sum_{i=1}^{{}^2n} \sum_{b=a+1}^{{}^1n} \sum_{j=i+1}^{{}^2n} \mathbf{C}_{ab,ij}\mathbf{M}_{ai}\mathbf{M}_{bj} + \alpha \sum_{a=1}^{{}^1n} \sum_{i=1}^{{}^2n} \mathbf{\Theta}^{\text{node}}_{ai}\mathbf{M}_{ai}, \tag{3.2}$$

$$\text{s.t. } \forall a \textstyle\sum_{i=1}^{{}^2n} \mathbf{M}_{ai} \leq 1, \forall i \textstyle\sum_{a=1}^{{}^1n} \mathbf{M}_{ai} \leq 1, \forall ai\, \mathbf{M}_{ai} \in \{0,1\},$$

where $\mathbf{\Theta}^{\text{node}}_{ai}$ represents the visual similarity of the node a in ${}^1\mathbf{G}$ to node i in ${}^2\mathbf{G}$. α is a weighting parameter to trade off between the importance of the two tasks. This becomes an attributed graph matching problem.

Lastly we could go further and maximize the visual similarity of the edges connecting the nodes.

$$\max_{\mathbf{M}} S^{\text{awgm}}(\mathbf{M}) = \sum_{a=1}^{{}^1n} \sum_{i=1}^{{}^2n} \sum_{b=a+1}^{{}^1n} \sum_{j=i+1}^{{}^2n} \mathbf{\Theta}^{\text{edge}}_{abij}\mathbf{M}_{ai}\mathbf{M}_{bj} + \alpha \sum_{a=1}^{{}^1n} \sum_{i=1}^{{}^2n} \mathbf{\Theta}^{\text{node}}_{ai}\mathbf{M}_{ai}, \tag{3.3}$$

$$\text{s.t. } \forall a \textstyle\sum_{i=1}^{{}^2n} \mathbf{M}_{ai} \leq 1, \forall i \textstyle\sum_{a=1}^{{}^1n} \mathbf{M}_{ai} \leq 1, \forall ai\, \mathbf{M}_{ai} \in \{0,1\},$$

where $\mathbf{\Theta}^{\text{edge}}_{abij}$ represents the visual similarity of the edge between nodes a and b in ${}^1\mathbf{G}$ to the edge between nodes i and j in ${}^2\mathbf{G}$. $\mathbf{\Theta}^{\text{edge}}_{abij}$ will only be non-zero when $\mathbf{C}_{abij}$ is non-zero. Similar to Equation 3.1, we do not need to compute $\mathbf{\Theta}^{\text{edge}}$ directly. Instead, we make the sparse matrices ${}^1\mathbf{G}$ and ${}^2\mathbf{G}$ contain the index of an edge in a edge similarity matrix of size ${}^1l \,\text{x}\, {}^2l$ and once again iterate through the non-zero elements of ${}^1\mathbf{G}$ and ${}^2\mathbf{G}$. Equation 3.3 is a form of attributed weighted graph matching.

As all of the above are generalizations of the inexact graph matching problem, they, unfortunately, fall into the class of NP-Hard problems [89]. As such, we look to find an approximate solution to this problem.

3.5 Algorithm Overview

Graduated Assignment is well described by Gold and Rangarajan [23] and has been shown to be capable of providing an approximate solution to the problem above. While GA is very efficient with a complexity of $O({}^1l^2l)$, where ${}^1l$ and ${}^2l$ are the number of edges in ${}^1\mathbf{G}$

and ${}^2\mathbf{G}$ respectively, the number of iterations required can still induce a high compute cost. We motivate a faster algorithm by looking at what reductions can be made to GA for the wireframe matching task. In this section we describe our simplified solution as it compares to the original algorithm. While the simplifications have the potential to reduce the accuracy of the final matching, we attempt to motivate why this accuracy reduction should be minimal. The solution here is described specifically as applied to Equation 3.3, however the solutions for Equations 3.1 and 3.2 follow similarly.

First we define the energy function as the negative of the similarity function ($E^{\text{awgm}} = -S^{\text{awgm}}$) and maximize the similarity function by minimizing the energy function. If we define an initial correspondence matrix $\mathbf{M}^0$ and take the first order Taylor Series approximation of this energy function we get:

$$E^{\text{awgm}}(\mathbf{M}) \approx E^{\text{awgm}}(\mathbf{M}^0) - \sum_{a=1}^{{}^1n} \sum_{i=1}^{{}^2n} \mathbf{Q}_{ai}(\mathbf{M}^0_{ai} - \mathbf{M}_{ai}) \tag{3.4}$$

$$\mathbf{Q}_{ai} = \alpha \mathbf{\Theta}^{\text{node}}_{ai} + \sum_{b=a+1}^{{}^1n} \sum_{j=i+1}^{{}^2n} \mathbf{\Theta}^{\text{edge}}_{abij} \mathbf{M}^0_{bj} \tag{3.5}$$

From this we find that we can descend on our energy function by maximizing:

$$\sum_{a=1}^{{}^1n} \sum_{i=1}^{{}^2n} \mathbf{Q}_{ai} \mathbf{M}_{ai} \tag{3.6}$$

By finding the $\mathbf{M}$ that maximizes (3.6) and repeatedly relinearizing about our new $\mathbf{M}$ we can repeatedly descend on this function until we achieve a local optimum. This, however, is still non-trivial due to the constraints we impose on $\mathbf{M}$ in Equation 3.3.

The Set Constraint

As with the original algorithm, to deal with the constraint $\mathbf{M}_{ai} \in \{0, 1\}$, we employ a continuation method known as simulated annealing or graduated non-convexity [75]. This involves relaxing the constraint to allow the elements of $\mathbf{M}$ to lie in the continuous range $[0, 1]$. A control variable β is used to slowly push the values of $\mathbf{M}$ to either 0 or 1. This is done by utilizing the softmax function:

$$x'_i = \frac{e^{\beta x_i}}{\sum_{j=1}^{n} e^{\beta x_j}} \tag{3.7}$$

By increasing β over time we approximate our original constraint. In GA, multiple iterations are used to descend on $E^{\text{awgm}}(\mathbf{M})$ for each value of β. For several reasons, this may not be necessary. First, we don't actually need to converge for a given value of β, we just need to get to a value of $\mathbf{M}$ where the next β still leaves us in a good local concavity. Second, we

don't necessarily need to converge to the minima at the last step either, as we can instead perform a greedy assignment step to fully enforce the constraint $\mathbf{M}_{ai} \in \{0, 1\}$. This involves taking the max of each row in $\mathbf{M}$, setting it to 1, and setting all other elements of $\mathbf{M}$ to 0. Lastly, we note that as the graph becomes more sparse, the number of nonlinear terms decreases, thus reducing the need for multiple iterations. As wireframes typically form very sparse graphs, we propose that in performing a single descent step for each value of β we do not sacrifice significant accuracy as compared to the iterative approach.

The Uniqueness Constraint

The inequality constraints on $\mathbf{M}$ ($\forall a \sum_{i=1}^{^2n} \mathbf{M}_{ai} \leq 1,\ \forall i \sum_{a=1}^{^1n} \mathbf{M}_{ai} \leq 1$) enforce that at each node in ${}^1\mathbf{G}$ is uniquely matched to, at most, a single node in ${}^2\mathbf{G}$. One way to enforce these constraints is to first note that if these were equality constraints ($\forall a \sum_{i=1}^{^2n} \mathbf{M}_{ai} = 1, \forall i \sum_{a=1}^{^1n} \mathbf{M}_{ai} = 1$) and $\mathbf{M}$ were a square matrix then $\mathbf{M}$ would be what is known as a doubly stochastic matrix. It is well known that any square matrix with all positive elements can be converted into a doubly stochastic matrix via the Sinkhorn-Knopp algorithm [81]. This involves alternating between normalizing all rows and normalizing all columns until the algorithm converges. Thus, if our matrix is square, we can simply add an additional row and column of "slack variables", and by applying the Sinkhorn-Knopp algorithm, we enforce our inequality constraint on our original matrix.

GA uses the Sinkhorn-Knopp algorithm with slack variables to both enforce the uniqueness constraints and determine outliers (the issue of non-square matrices is not addressed). For image feature matching there exist many good methods for determining outliers. As such, we propose only enforcing the constraints unilaterally and apply external outlier rejection to remove non-unique feature matches. It is noted by Gold and Rangarajan [23] that performing only one iteration of the Sinkhorn-Knopp algorithm is identical to only enforcing the constraint in one direction. This also removes the need for slack variables. It is worth noting that each Sinkhorn-Knopp iteration is only $O({}^1n^2n)$ rather than $O({}^1l^2l)$. That is, it's complexity grows with the number of nodes in the graphs rather than the number of edges. In a densely connected graph this change does not reduce the computational cost significantly as the number of edges would far exceed the number of nodes. As mentioned above, however, wireframes in a man-made environment are likely to be very sparse meaning that this step makes up a significant part of the algorithm's compute cost.

Additional Parameters

The recommended parameters for the original GA, given by Gold and Rangarajan [23], are: the initial annealing parameter $\beta_0 = 0.5$, the final annealing parameter $\beta_f = 10$, the rate of increase $\beta_r = 1.075$, the maximum number of descent iterations for a given β $t_2^{\max} = 4$, and the maximum number of Sinkhorn-Knopp iterations for each descent $t_3^{\max} = 30$. There are several changes to these parameters for our algorithm.

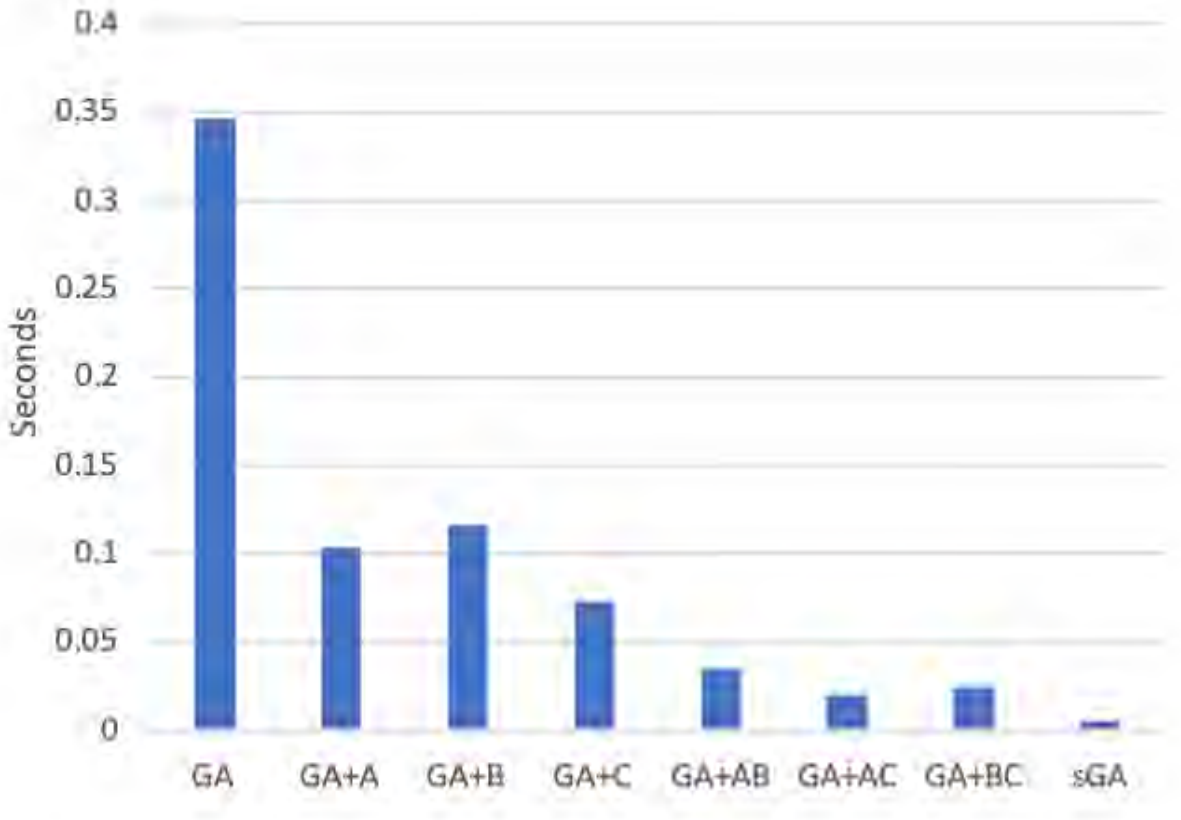


Figure 3.2: Effect of each of the simplifications from section 3.5 on the average runtime of wireframe matching on fr3. sGA is the full simplified algorithm with all the changes applied. "A" are the reductions proposed in the "Set Constraint" subsection. "B" are the reductions proposed in the "Uniqueness Constraint" subsection. "C" are the reductions proposed in the "Additional Parameters" subsection.

First, both t_2 and t_3 are effectively set to 1 due to the reductions proposed above. Second, we note that the rate of our simulated annealing algorithm is typically intended to be small to attempt to push us slowly towards a global optimum. If we accept that we are only looking to improve on our energy function and not necessarily achieve a global optimum, then we can increase this parameter significantly as compared to the original algorithm. Thus we increase the parameter β_r to 1.5. Lastly for our work, we initialize $\mathbf{M}$ to the node similarity matrix $\mathbf{\Theta}^{\text{node}}$. As we are not starting from an entirely random initialization of $\mathbf{M}$, we increase β_0 slightly to 1.0.

The full simplified GA is shown in Algorithm 1. The recommended parameters are the following: $\beta_0 = 1$, $\beta_f = 10$, $\beta_r = 1.5$.

These changes together have the effect of reducing the total worst case number of Sinkhorn-Knopp iterations (which are $O({}^1n^2n)$) by a factor of 1000, and the worst case

number descent steps (which are $O({}^1l^2l)$) by a factor of 30 as compared to the original GA with recommended parameters. The effect of each of these changes on runtime is shown in Figure 3.2.

Algorithm 1: Simplified Graduated Assignment For AWGM

$\beta \leftarrow \beta_0$
$\mathbf{M} \leftarrow \mathbf{\Theta}^{\text{node}}$
while $\beta < \beta_f$ **do**
  $\forall a \in {}^1n \, \forall i \in {}^2n$
  $\mathbf{Q}_{ai} \leftarrow \alpha \mathbf{\Theta}^{\text{node}}_{ai} + \sum_{b=a+1}^{{}^1n} \sum_{j=i+1}^{{}^2n} \mathbf{\Theta}^{\text{edge}}_{abij} \mathbf{M}^0_{bj}$
  $\forall a \in {}^1n \, \forall i \in {}^2n$
  $\mathbf{M}_{ai} \leftarrow e^{\beta \mathbf{Q}_{ai}}$
  $\forall a \in {}^1n \, \forall i \in {}^2n$
  $\mathbf{M}'_{ai} \leftarrow \frac{\mathbf{M}_{ai}}{\sum_{j=0}^{{}^2n+1} \mathbf{M}_{aj}}$
  $\forall a \in {}^1n \, \forall i \in {}^2n$
  $\mathbf{M}_{ai} \leftarrow \frac{\mathbf{M}'_{ai}}{\sum_{b=0}^{{}^1n+1} \mathbf{M}'_{bi}}$
  $\beta \leftarrow \beta * \beta_r$
end
$\mathbf{M} \leftarrow$greedy_assignment($\mathbf{M}$)

3.6 Applications

In this chapter we look at two applications of our simplified GA. While our primary interest is in the matching of image wireframes, we understand that the cost of wireframe extraction (when specialized hardware is unavailable) would likely outweigh the benefits of having a realtime matching algorithm. As such, we are also interested in the applicability of the simplified GA to standard feature matching. The use in standard feature matching (for which we use ORB features), would allow this algorithm to potentially improve a wide variety of algorithms that use feature matching in their pipeline. These algorithms include Simultaneous Localization And Mapping and 3D object tracking.

Wireframe Matching

We utilize the publicly available, pretrained network from Zhou et al. [113] for wireframe extraction. In this network the number of detected junctions is set to 250 for each image, and we keep all lines with a score of 0.9 or greater up to a maximum of 2500 lines. While the

network always produces 250 junctions, it is possible for these junctions to be in identical locations and so we remove duplicate junctions before proceeding. In order to support visual feature matching, we extract BREIF [9] descriptors around each junction, and binary line descriptors [92, 106] around each line. While either junctions or lines can be used as the nodes, we decided to use junctions as the nodes of the graph. The similarity matrix $\mathbf{\Theta}^{\text{node}} \in \mathbb{R}^{^1n \times {}^2n}$ is therefore computed by first finding the Hamming distance of the BRIEF descriptors from the first image to those from the second image. We truncate the distance to a maximum value $d^{\max} = 50$ for easier matrix scaling. This however gives us a measure of distance rather than a measure of similarity, and so we transform the matrix by: $\mathbf{\Theta}^{\text{node}}_{ai} = (d^{\max} - \Gamma^{\text{node}}_{ai})/d^{\max}$ where Γ^{node} is the matrix of descriptor distances. The reduced $({}^1l \times {}^2l)$ line similarity matrix follows similarly. As both the line and feature matches are similarly scaled and measure similar information, we set the α parameter to 1.

ORB Feature Matching

For our standard feature matching tests, we first detect a maximum of 300 ORB [76] features in each image. To convert our set of features in each image into a graph, we add an edge between any two features whose distance in image space is less that τ pixels. We then optimize using the energy function defined in Equation 3.2. This has the effect of optimizing under the premise that features that are close in one image, are likely to be close in the second image. This is true whenever the scene is relatively static and the features are far from the image plane compared to the translation between the two images. For our experiments using ORB features we set τ to 10 pixels on 640x480 resolution images and only consider matches whose hamming distance is less than 50. We do not compute edge similarity for this experiment, however many others have proposed a wide variety of similarity measures [89, 110, 11], that could also be used in practice.

3.7 Experiments

We compare the two versions of Graduated Assignment discussed in this chapter. We denote the original method as "GA", and our simplified GA as "sGA". Our baseline comparison in both of the above cases is standard brute force feature matching (denoted "BF"). As this does not use any line information we compare against two versions of the above algorithms. The first which we denote with the suffix "_Node", only uses node similarity (solving Equation 3.2 rather than 3.3). The suffix "_Both" denotes the use of both node and edge similarity. As we do not compute edge similarity for ORB feature matching, all the algorithms used in that experiment use only node similarity.

For GA we employ a slight deviation from the algorithm described by Gold and Rangarajan [23], which applies the Sinkhorn-Knopp algorithm directly despite that, in many cases, the number of rows and columns is not equal. Though they achieve good performance in practice, a rectangular matrix cannot simultaneously have each it's columns and each of

its rows sum to 1. To solve this, we employ a simple modification to the Sinkhorn-Knopp algorithm. Instead of normalizing *all* rows and columns, we normalize all rows and columns *except* the one row and column added as slack variables. This results in a matrix where all rows and all columns except the added row and column sum to one. In our experiments this achieves significantly better performance on both square and non-square matrices.

In addition we looked to compare against the the Dual Decomposition algorithm proposed by Torresani et al. [89]. We found, however, that running Dual Decomposition took several hours for a single image. Instead we turn to the next highest performing algorithm evaluated by Torresani et al. [89]: Max-Product Belief Propagation (denoted "BP"). Here we transform the problem by removing the inequality constraints in Equation 3.3 and instead add them as a large cost term. Thus the function we maximize via Belief Propagation is as follows:

$$\begin{aligned}\max_{\mathbf{M}} S^{\text{bp}}(\mathbf{M}) = &\sum_{a=1}^{{}^1n}\sum_{i=1}^{{}^2n}\sum_{b=a+1}^{{}^1n}\sum_{j=i+1}^{{}^2n}\mathbf{\Theta}^{\text{edge}}_{abij}\mathbf{M}_{ai}\mathbf{M}_{bj} \\ &+\alpha\sum_{a=1}^{{}^1n}\sum_{i=1}^{{}^2n}\mathbf{\Theta}^{\text{node}}_{ai}\mathbf{M}_{ai} - \eta\sum_{a=1}^{{}^1n}\sum_{i=1}^{{}^2n}\sum_{j=i+1}^{{}^2n}\mathbf{M}_{ai}\mathbf{M}_{aj} \\ &-\eta\sum_{a=1}^{{}^1n}\sum_{b=a+1}^{{}^1n}\sum_{i=1}^{{}^2n}\mathbf{M}_{ai}\mathbf{M}_{bi}, \quad (3.8)\end{aligned}$$

$$\text{s.t. } \forall ai\, \mathbf{M}_{ai} \in \{0,1\},$$

where η is a large constant. As stated by Torresani et al. [89], this problem is equivalent to Equation 3.3, however this formulation introduces a large number terms to the energy minimization problem.

Dataset

We evaluate our algorithms on two scenes from the TUM [82] dataset: the "fr2/large_no_loop" (which we denote fr2) and "fr3/long_office_household" (which we denote fr3) scenes. These scenes are indoor man-made structured environments, where we feel that wireframe reconstruction would be applicable. For each scene we sample two neighboring images from every 10 images. This results in a total of 335 test pairs for the first scene and 258 for the second. Notably, the rgb images in fr3 are rectified to have no lens distortion. The fr2 images, however, are not rectified but the distortion parameters are known. As the TUM datasets have ground truth trajectory information, we can compute the ground truth Fundamental Matrix and count the number of inliers whose Symmetric Epipolar Distance falls below a threshold. While this is not an exact measure of correct matches, it allows us to estimate the average improvement in the number of correct matches over the baseline.

fr2 Wireframe Matching Results

Algorithm	% Improv.	Avg. Improv.	Avg. FPS
sGA_Node	32.74%	12.60	267.52
sGA_Both	39.26%	15.24	259.74
GA_Node	34.74%	13.46	3.26
GA_Both	39.50%	15.32	3.34
BP_Node	8.84%	3.50	0.15
BP_Both	25.96%	10.39	0.15

fr3 Wireframe Matching Results

Algorithm	% Improv.	Avg. Improv.	Avg. FPS
sGA_Node	26.62%	14.77	235.95
sGA_Both	31.70%	17.43	228.63
GA_Node	28.12%	15.64	2.81
GA_Both	31.91%	17.61	2.89
BP_Node	6.28%	3.85	0.13
BP_Both	20.33%	11.75	0.13

Table 3.1: Wireframe matching results in two scenes from the TUM dataset

3.8 Results and Discussion

For each algorithm the average percent improvement is calculated as: $\frac{1}{K}\sum_{k=1}^{K}\frac{\iota_k-\iota_k^{\#}}{\iota_k^{\#}}$ where K is the number of images in the scene, ι_k is the number of inliers found by the algorithm for image k, and $\iota_k^{\#}$ is the number of inliers found by the baseline algorithm. Dividing by the number of inliers found in the baseline algorithm is an attempt to normalize for the difficulty of the matching, however for reference we also present the non-normalized results calculated as $\frac{1}{K}\sum_{k=1}^{K}\iota_k-\iota_k^{\#}$. To remove scenes with heavy motion blur or insufficient visual texture, we do not consider images where the brute force matching method fails to find at least 10 inliers.

The results for Wireframe matching are shown in Table 3.1. For fr2, the brute force matching algorithm correctly matched an average of 44 nodes out of an average of 117 nodes detected in each image. For fr3, the brute force matching algorithm correctly matched an average of 62 nodes out of an average of 127 nodes detected in each image. We see that, in both scenes, sGA has near identical performance to GA, increasing the number of correctly matched features by over 25% when using only node similarity and over 30% when using both node and edge similarity. We see that the BP algorithm takes significantly longer than both sGA and GA and only achieves a 20-25% improvement, even when using both node and edge similarity. The increased runtime is likely due to the large number of terms added

by transforming the uniqueness constraints.

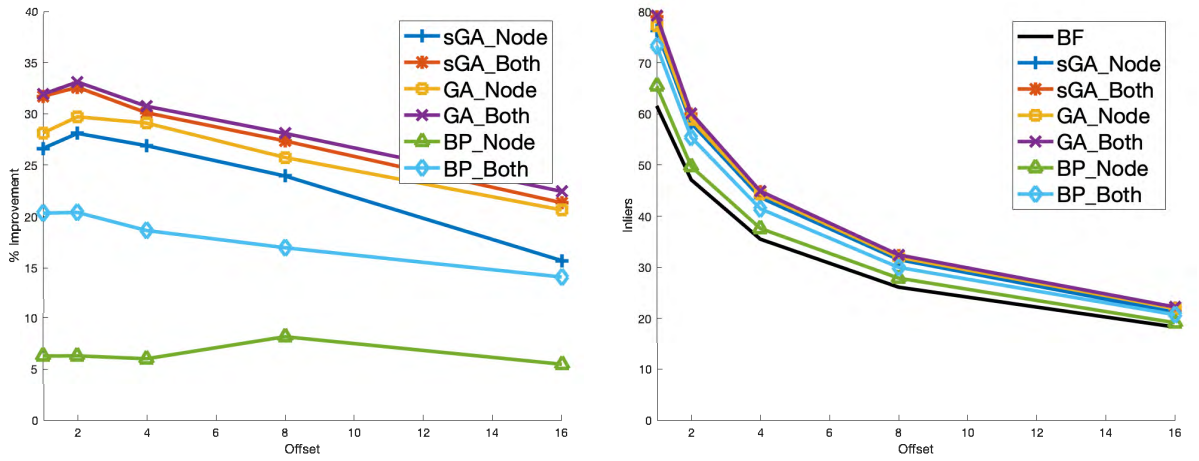


Figure 3.3: Effect of increasing the time between images on fr3. Offset indicates the number of images between the compared images in the sequence. We see that the total number of inliers (right) decreases, but the percent improvement over the baseline (left) is maintained.

We further tested to see if these results would hold as the rotation and translation between the images increased. For this we performed a similar sampling as described in section 3.7 except, instead of neighboring images, we chose images at an increased offset apart. As shown in Figure 3.3, the percent improvement does not change significantly as the offset is increased. This hints that, while our features' similarity matrix becomes less accurate as the offset increases, the wireframe extraction remains accurate, and therefore continues to enable improvement over the baseline. The small drop in accuracy by sGA_Node at the 16 image offset may indicate a greater susceptibility of sGA to noise in the similarity matrices than GA.

The results for ORB feature matching are shown in Table 3.2. For fr2, the brute force matching algorithm correctly matched an average of 132 nodes out of an average of 287 nodes detected in each image. For fr3, the brute force matching algorithm correctly matched an average of 175 nodes out of an average of 297 nodes detected in each image. In this experiment we see that sGA still achieves a 15% improvement, with the standard GA only giving a slightly better improvement. As the graph for ORB feature matching is somewhat less sparse than the wireframe case, this slight improvement may indicate that the additional iterations of GA are more important when there are more non-linear terms in the optimization problem. BP however fails to provide any benefit in this case. This could indicate a fragility to noise in the graph, or an inability to scale to large numbers of nodes.

fr2 ORB Matching Results

Algorithm	% Improv.	Avg. Improv.	Avg. FPS
sGA	23.05%	20.34	31.78
GA	24.08%	22.81	0.55
BP	1.05%	-4.95	0.01

fr3 ORB Matching Results

Algorithm	% Improv.	Avg. Improv.	Avg. FPS
sGA	15.51%	21.87	36.52
GA	17.49%	24.67	0.57
BP	-5.53%	-11.75	0.01

Table 3.2: ORB matching results in two scenes from the TUM dataset

3.9 Conclusion

We have demonstrated a realtime algorithm for the matching of image wireframes that achieves significant improvement over brute force feature matching. We have further shown how this simplification of the Graduated Assignment algorithm can be applied to the standard feature matching problem. In the future we plan to look at how this algorithm could be applied to utilize other types of structure in a scene. For example, this algorithm could be used to match semantic object labels in sequences of images. Additionally we would like to look into utilizing a GPU for parallelization of both the standard Graduated Assignment algorithm as well as this simplified algorithm.

Chapter 4

Applications of Wireframe Matching to Multi-Camera Rigs

4.1 Introduction

There have been many impressive results in both Simultaneous Localization and Mapping (SLAM) and 3D reconstruction using monocular camera systems. As we move towards a SLAM and 3D reconstruction system that can operate continuously and ubiquitously, it may be necessary to introduce multiple cameras to increase reliability. For example monocular camera systems rely on an inertial measurement unit (IMU) to determine the scale of the scene during 3D reconstruction. Unfortunately, due to the dependence of the IMU bias parameters on the state, the true scale is inherently unobservable [30]. Stereo and multi-camera rigs have been used very successfully to improve robustness and add reliable scale to mapping and reconstruction [66, 73, 83, 41, 52].

The goal of wireframe reconstruction is to enable robust and compact reconstructions of structured environments. As we move towards this goal we can look towards multi-camera rigs to present a reliable method for generating accurate 3D structures. In the previous chapter, we showed that by utilizing a modified graduated assignment algorithm we can improve feature and wireframe matching in monocular camera systems. In this chapter we present several methods for utilizing multiple cameras for matching wireframes across camera rigs. We present a method for merging multiple wireframe estimates from a camera rig into a single wireframe where the 3D positions of some nodes can be computed. We then improve the matching between rigs by encouraging features observed by multiple cameras within a new camera rig to be matched to the same feature in this merged wireframe. We further adapt the graduated assignment algorithm to the Iterative Closest Point algorithm for utilizing 3D constraints induced by the merged wireframe for feature matching and transform estimation. Lastly, we demonstrate how visual and geometric features can be combined to further improve matching. The quality of matching induced by each algorithm is evaluated on realistic image sequences.

These systems together present the basis of a multi-camera wireframe reconstruction method.

4.2 Related Work

3D Wireframe Detection

A wireframe consists of a set of straight lines and their intersected junctions that make up the structural components of the scene [35]. If the 3D positions of these lines and junctions can be estimated, the wireframe could represent a compact model of the world. Zhou et al. [114] extend the work of Huang et al. [35] to additionally predict the 3D positions of these lines and junctions under the Manhattan assumption. A related problem is that of room layout estimation which only estimates the lines and junctions corresponding to the walls and floors of the scene. Several networks estimate room layout from a single RGB image [115, 99, 100]. Both Zou et al. [115] and Yang et al. [99] employ a Manhattan assumption, while Yang et al. [100] employ a soft Manhattan assumption.

To our knowledge, this work presented in this thesis represents the first method for utilizing multiple cameras for 3D wireframe estimation and the first method to perform 3D wireframe estimation without a Manhattan assumption.

Multi-Camera SLAM

Multi-camera rigs have been used to great success in visual localization and mapping [66, 73, 87, 83, 41, 52]. Many of these algorithms do not detect feature matches between overlapping cameras [73, 87], instead relying on outlier rejection to prevent features from being matched to incorrectly in one or more cameras. Those that do compute matches between overlapping cameras typically only do so for the 3D constraints gained from this [66, 52] or to additionally remove outliers [83, 41]. Both these methods discard the additional visual descriptors that result from having seen the feature from multiple views. In this work, we present a novel algorithm for multi-camera feature matching that is able to utilize not only the visual information of multiple descriptors, but also the connectivity relationships that results from wireframe detection in multiple cameras.

Multi-Graph Matching

As discussed in the previous chapter, a wireframe can be considered a graph linking visual features in an image. Therefore, it is reasonable to consider the problem of matching multiple sets of wireframe detections as a multiple graph matching problem. Enforcing that matched nodes in each pair of graphs are consistently matched to the same node in every other graph is known as cycle consistency. Several papers exist that attempt to match multiple graphs while maintaining cycle consistency [98, 85, 97, 71], however the computational complexity

of these algorithms exceeds what is attainable in realtime for the number of nodes required for 3D reconstruction. The work of Zhou et al. [112] is able to operate in realtime, however it only optimizes for cycle consistency after pairwise matchings are complete, and still incurs higher computational complexity than our algorithm. Our work is a subset of the multi-graph matching problem where we seek to match multiple graphs to a single "keyframe" graph and only enforce cycle consistency in cycles involving the "keyframe" graph. Further, we fix matches between graph pairs where neither graph is the keyframe graph. This is because we can remove outlier matches between cameras in the rig due to the known camera extrinsics. This allows us to develop an algorithm that is linear in the number of cameras and linear in the number of overlaps (though the number of overlaps may be quadratic in the number of cameras in the rig) allowing the efficient matching of multiple graphs in realtime.

Iterative Closest Point

The use of multiple appropriately placed cameras allows us to immediately triangulate points in 3D, this enables the possibility of using 3D information rather than visual information for computing correspondences between camera rigs. One of the most common methods for this is through the Iterative Closest Point (ICP) algorithm [5]. Significant amounts of work have been done relating to ICP. Bouaziz et al. [6] introduce a set of robust cost functions are developed for the Iterative Closest Point algorithm that show increase robustness in the presence of outliers. Similar to these cost functions we utilize the Cauchy-loss to robustify our result. Tykkälä et al. [90] utilize a depth cost term is combined with a visual term to form an ICP error function, however no results are given with regards to the accuracy of this method. Luck et al. [63] use a stochastic simulated annealing method alongside a robust cost function to allow the algorithm to escape local minima. This method involves randomly searching for a new point outside of the local minima, which can be very slow if minima are far. In our work we apply a deterministic annealing strategy to Iterative Closest Point through the graduated assignment algorithm. This allows us to incorporate wireframe information, but also enables the application of simulated annealing to K-nearest neighbors Iterative Closest Point. Most similar to our approach here is that of Zhou et al. [109] which performs alternating minimization between graph matching and 2D transform estimation. In our work we go beyond this to perform SE(3) transform estimation under deterministic annealing in addition to the standard matching-transform alternation and demonstrate improved accuracy as a result. We also demonstrate how this can be used to fuse both geometric and visual information into a single optimization problem. An example of this iterative matching method is shown in 4.1.

4.3 Contribution

The contributions of this chapter are as follows:

1. An algorithm for combining multiple image wireframes into a single wireframe estimate.

2. An algorithm for efficiently incorporating matches across overlapping images into the wireframe matching problem to improve matching between camera rigs.

3. An extension of wireframe matching to the Iterative Closest Point (ICP) algorithm that is able to significantly improve matching between 3D points.

4. A joint matching and transform estimation algorithm that incorporates both visual and 3D constraints to improve feature matching.

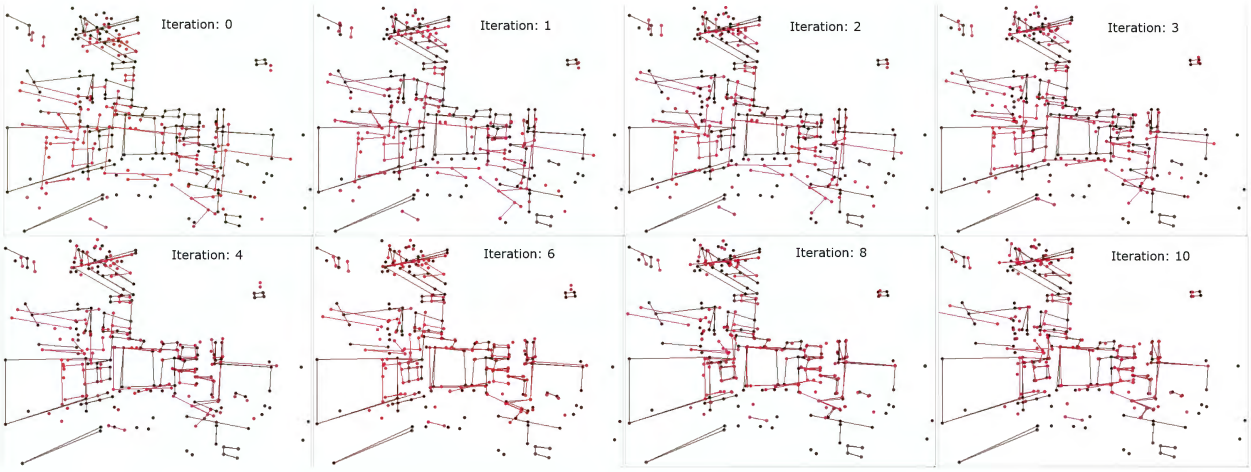


Figure 4.1: A visualization of the Iterative Wireframe Matching procedure. (red) The projection of a 3D wireframe onto the current image using the current transform estimate. (black) The observed wireframe for the current image. As the transform is updated with each iteration, the estimated wireframe more closely aligns with the observed wireframe.

4.4 Problem Statement

In this chapter we demonstrate how to match wireframe detections across rigs consisting of multiple cameras. To do this we first create a single estimated wireframe from the multiple cameras in a single camera rig. We will call this our "keyframe wireframe". We then attempt to compute a matching from a new camera rig to this keyframe wireframe. This is similar to the use of keyframe images in SLAM [42, 52, 102], and also to the use of a virtual "true" graph in multi-graph matching [112, 71]. As in the previous chapter, we will represent our wireframes as graphs, and attempt to match a set of graphs to our keyframe graph. We denote the keyframe graph as ${}^{\text{key}}\mathbf{G}$. It contains ${}^{\text{key}}n$ nodes. Each wireframe in our camera rig corresponds to a graph ${}^{1}\mathbf{G}...{}^{m}\mathbf{G}$ where m is the number of cameras in the rig. Each graph for the camera rig has ${}^{1}n...{}^{m}n$ nodes respectively. We can represent each graph as a sparse matrix where $\mathbf{G}_{ab} = 1$ if node a is connected to node b and 0 otherwise. We then

seek to find a matching between nodes in graphs ${}^{1}\mathbf{G}...{}^{m}\mathbf{G}$ to corresponding nodes in ${}^{\text{key}}\mathbf{G}$. A basic form of this problem would look like:

$$\max_{{}^{1}\mathbf{M}...{}^{m}\mathbf{M}} S^{\text{pair}}({}^{1}\mathbf{M}...{}^{m}\mathbf{M}) = \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n}\sum_{b=a+1}^{{}^{c}n}\sum_{j=i+1}^{{}^{\text{key}}n} {}^{c}\mathbf{C}_{abij}\,{}^{c}\mathbf{M}_{ai}\,{}^{c}\mathbf{M}_{bj} + \alpha_{\text{vis}}\sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n} {}^{c}\mathbf{\Theta}^{\text{node}}_{ai}\,{}^{c}\mathbf{M}_{ai} \tag{4.1}$$

s.t. $\forall ca \sum_{i=1}^{n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall ci \sum_{a=1}^{{}^{\text{key}}n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall cai\ {}^{c}\mathbf{M}_{ai} \in \{0,1\}$

The Matrices ${}^{1}\mathbf{M}...{}^{m}\mathbf{M}$ represent the correspondence of nodes, with ${}^{c}\mathbf{M}_{ai} = 1$ indicating the matching of node a in ${}^{c}\mathbf{G}$ to node i in ${}^{\text{key}}\mathbf{G}$. ${}^{c}\mathbf{\Theta}^{\text{node}}_{ai}$ represents the visual similarity of the node a in ${}^{c}\mathbf{G}$ to node i in ${}^{\text{key}}\mathbf{G}$. ${}^{c}\mathbf{C}_{abij}$ represents the edge compatibility of the edge between nodes a and b in ${}^{c}\mathbf{G}$ to the edge between nodes i and j in ${}^{\text{key}}\mathbf{G}$. That is:

$${}^{c}\mathbf{C}_{abij} = \begin{cases} 1 & {}^{c}\mathbf{G}_{ab}\,{}^{\text{key}}\mathbf{G}_{ij} = 1 \\ 0 & otherwise \end{cases}$$

α_{vis} is a weighting parameter to trade off between the importance of visual node similarity versus edge compatability. This solves for the pairwise matching between each ${}^{c}\mathbf{G}$ and the keyframe graph ${}^{\text{key}}\mathbf{G}$ and can be solved in $O(m\tau_{\mathbf{G}^{\text{key}}\mathbf{G}})$ time, where $\tau_{\mathbf{G}^{\text{key}}\mathbf{G}}$ is the time it takes to match a single graph ${}^{c}\mathbf{G}$ to ${}^{\text{key}}\mathbf{G}$. Due to noise in our similarity matrices and noise in the graphs themselves, this may not provide the true matching for each wireframe. As such, we can expect to do better by incorporating more information into the problem.

If we know that certain cameras in the camera rig overlap, then we can expect that certain nodes in these wireframes should correspond to the same node in the keyframe wireframe ${}^{\text{key}}\mathbf{G}$. This is known as cycle consistency. We define the set P to contain pairs of overlapping cameras. Here we assume we can compute known correspondence matrices between graphs in overlapping cameras. This assumption arises from the assumption that we have a known extrinsic calibration between cameras in the rig. This allows us perform outlier rejection on the matching computed across these cameras. The matching before outlier rejection can be computed by any method, however here we choose to use the pairwise wireframe matching approach from the previous chapter. Computing these known correspondence matrices takes $O(p\tau_{GG})$ time, where p is the number of overlapping camera pairs and τ_{GG} is the time it takes to match a single graph ${}^{d}\mathbf{G}$ to its overlapping partner ${}^{e}\mathbf{G}$. As a result, we can optimize for cycle consistency by optimizing the following:

$$\max_{{}^{1}\mathbf{M}...{}^{m}\mathbf{M}} S^{\text{cycle}}({}^{1}\mathbf{M}...{}^{m}\mathbf{M}) = \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n}\sum_{b=a+1}^{{}^{c}n}\sum_{j=i+1}^{{}^{\text{key}}n} {}^{c}\mathbf{C}_{abij}\,{}^{c}\mathbf{M}_{ai}\,{}^{c}\mathbf{M}_{bj} + \alpha_{\text{vis}}\sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n} {}^{c}\mathbf{\Theta}^{\text{node}}_{ai}\,{}^{c}\mathbf{M}_{ai}$$
$$+ \gamma \sum_{\{e,f\}\in P}\sum_{a=1}^{{}^{e}n}\sum_{b=1}^{{}^{f}n}\sum_{i=1}^{{}^{\text{key}}n} {}^{ef}\mathbf{M}_{ab}\,{}^{e}\mathbf{M}_{ai}\,{}^{f}\mathbf{M}_{bi} \tag{4.2}$$

s.t. $\forall ca \sum_{i=1}^{n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall ci \sum_{a=1}^{{}^{\text{key}}n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall cai\ {}^{c}\mathbf{M}_{ai} \in \{0,1\}$
${}^{ef}\mathbf{M}$ is the correspondence matrix resulting from the pairwise matching of overlapping cameras e and f and γ is a weighting parameter. Because ${}^{ef}\mathbf{M}$ is sparse, this computation only adds $O(pn^{\text{key}}n)$ computation. For wireframe matching, where the number of nodes is relatively large, and the number of overlapping cameras in a rig is typically relatively small, this adds additional computation on the same order as our original problem, which (if solved with graduated assignment) is $O(m^{\text{key}}ll)$ with ${}^{\text{key}}l >= {}^{\text{key}}n$ and $l >= n$.

Additionally, the use of overlapping multi-camera rigs, allows us to compute the relative 3D position of many of the nodes in our wireframe. This allows us to use an estimated transform between the current camera rig, and our keyframe camera rig, to compute distances between wireframe nodes. In cases where visual information is inaccurate (for example in the case of a light source attached to our camera causing visual features to change as the camera moves), this geometric distance metric could subsume our visual distance metric. By including the estimated transform in our optimization, we can iteratively compute both the wireframe matching and the estimated transform between the camera rigs. The resulting problem has the form:

$$\max_{\mathbf{T},{}^{1}\mathbf{M}...{}^{m}\mathbf{M}} S^{\text{icpgm}}(\mathbf{T}, {}^{1}\mathbf{M}...{}^{m}\mathbf{M}) = \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n}\sum_{b=a+1}^{{}^{c}n}\sum_{j=i+1}^{{}^{\text{key}}n} {}^{c}\mathbf{C}_{abij}\, {}^{c}\mathbf{M}_{ai}\, {}^{c}\mathbf{M}_{bj} + \alpha_{\text{geo}} \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n} {}^{c}\delta_{ai}^{\text{node}}(\mathbf{T})\, {}^{c}\mathbf{M}_{ai} \quad (4.3)$$

s.t. $\mathbf{T} \in SE(3)\, \forall ca \sum_{i=1}^{n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall ci \sum_{a=1}^{{}^{\text{key}}n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall cai\ {}^{c}\mathbf{M}_{ai} \in \{0,1\}$
$\mathbf{T}$ is the $SE(3)$ transformation representing the pose of the new camera rig relative to the keyframe, parameterized as a 4 by 4 transformation matrix. ${}^{c}\delta_{ai}^{\text{node}}$ is a measure of similarity between the 3D node position in the keyframe wireframe and the node position in camera c. This can be either a 3D-2D similarity measure, using the projection of the keyframe node onto camera c and the 2D position of the corresponding node in ${}^{c}\mathbf{G}$, or a 3D-3D similarity measure, using the 3D position of the keyframe node and computing the 3D position of nodes in the new camera rig using overlapping cameras.

If we remove the terms involving edge consistency from Equation 4.3 we are left with:

$$\max_{\mathbf{T},{}^{1}\mathbf{M}...{}^{m}\mathbf{M}} S^{\text{icp}}(\mathbf{T}, {}^{1}\mathbf{M}...{}^{m}\mathbf{M}) = \alpha_{\text{geo}} \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n} {}^{c}\delta_{ai}^{\text{node}}(\mathbf{T})\, {}^{c}\mathbf{M}_{ai} \quad (4.4)$$

s.t. $\mathbf{T} \in SE(3)\, \forall ca \sum_{i=1}^{n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall ci \sum_{a=1}^{{}^{\text{key}}n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall cai\ {}^{c}\mathbf{M}_{ai} \in \{0,1\}$
which is a standard problem in robotics, typically solved via the Iterative Closest Point algorithm [5] and this is where we will look for inspiration on solving Equation 4.3.

Finally we can combine all of the above together by incorporating both geometric and visual information:

$$\max_{\mathbf{T},{}^1\mathbf{M}...{}^m\mathbf{M}} S^{\text{joint}}(\mathbf{T}, {}^1\mathbf{M}...{}^m\mathbf{M}) = \sum_{c=1}^{m}\sum_{a=1}^{{}^cn}\sum_{i=1}^{{}^{\text{key}}n}\sum_{b=a+1}^{{}^cn}\sum_{j=i+1}^{{}^{\text{key}}n} {}^c\mathbf{C}_{abij}\,{}^c\mathbf{M}_{ai}\,{}^c\mathbf{M}_{bj} +$$

$$\sum_{c=1}^{m}\sum_{a=1}^{{}^cn}\sum_{i=1}^{{}^{\text{key}}n}(\alpha_{\text{vis}}\,{}^c\mathbf{\Theta}_{ai}^{\text{node}} + \alpha_{\text{geo}}^c\delta_{ai}^{\text{node}}(\mathbf{T}))\,{}^c\mathbf{M}_{ai} + \gamma\sum_{\{e,f\}\in P}\sum_{a=1}^{{}^en}\sum_{b=1}^{{}^fn}\sum_{i=1}^{{}^{\text{key}}n} {}^{ef}\mathbf{M}_{ab}\,{}^e\mathbf{M}_{ai}\,{}^f\mathbf{M}_{bi} \quad (4.5)$$

s.t. $\mathbf{T} \in SE(3)\,\forall ca\ \sum_{i=1}^{n} {}^c\mathbf{M}_{ai} \leq 1,\ \forall ci\ \sum_{a=1}^{{}^{\text{key}}n} {}^c\mathbf{M}_{ai} \leq 1,\ \forall cai\ {}^c\mathbf{M}_{ai} \in \{0,1\}$

While this requires both visual and geometric similarity to be computed, the fusion of information should result in a more accurate matching.

In the following sections we will describe how compute a 3D wireframe from a single camera rig and how to solve each of the above problems using the sGA algorithm presented in the previous chapter. We will also examine the impact each of these changes makes on the accuracy of our results.

4.5 Computing 3D Wireframes

This section describes how to combine multiple wireframes from a given camera rig into a single wireframe estimate. That is given a set of wireframe graphs ${}^1\mathbf{G}...{}^m\mathbf{G}$ our goal is to merge these graphs into a single estimated wireframe graph ${}^{\text{key}}\mathbf{G}$. ${}^{\text{key}}\mathbf{G}$ represents the keyframe wireframe for which the cameras in a new camera rig will be matched to. The first step to computing ${}^{\text{key}}\mathbf{G}$ is to compute correspondence matrices between wireframes in overlapping camera pairs. While various matching techniques can be used, we utilize the sGA method from the previous chapter. The next step is to perform outlier rejection. For this we form the fundamental matrix from the known camera extrinsics and remove all matches with a Symmetric Epipolar Distance greater than a threshold. This gives us a good estimate of the correspondences between overlapping cameras.

To build our graph from these correspondences, all nodes from all cameras are added to a single graph. At this point, the graph contains duplicates of corresponding nodes. To remove these we iteratively merge nodes that correspond, keeping track of all cameras they were observed in. The next step is to compute the connectivity matrix. This is done in a manner similar to occupancy mapping [64]. Each entry in the matrix will contain the log-odds estimate of occupancy. For each pair of nodes in each wireframe graph ${}^c\mathbf{G}$, the entry in the connectivity matrix will be incremented by the log of the observation probability if the nodes are connected in ${}^c\mathbf{G}$ or decremented by the log of the observation probability if the nodes are not connected in ${}^c\mathbf{G}$. All entries with a log-odds greater than a threshold are assumed to be connected, and below disconnected. This process is shown in Figure 4.2. Finally the 3D position of each node with more than a given threshold of observations is computed via least squares triangulation.

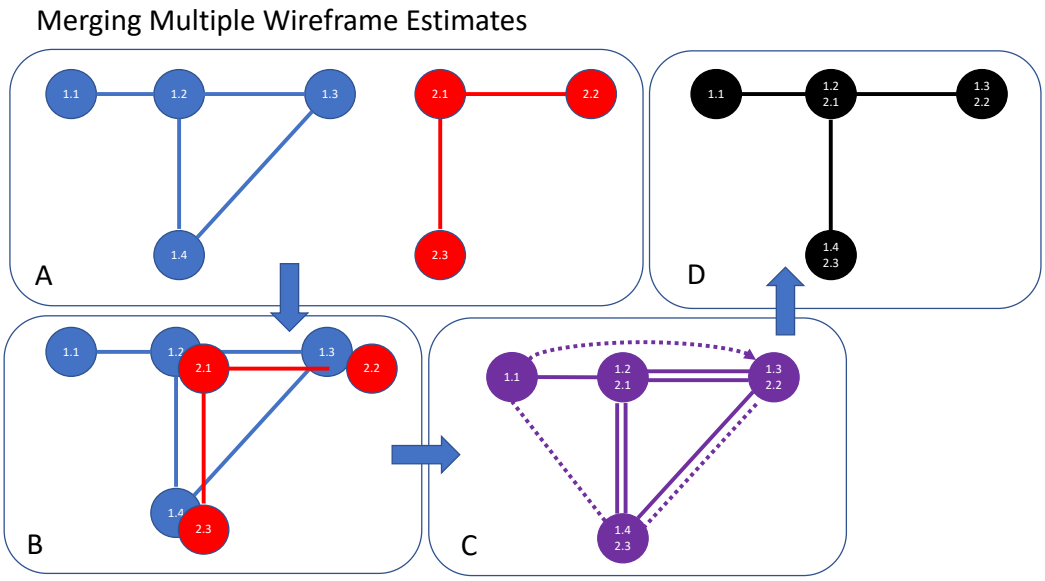


Figure 4.2: Process of merging multiple wireframe estimates. The individual wireframe estimates are shown in A. Solid lines represent nodes that are observed to be connected. In B all the nodes are added to a single wireframe. In C, matched nodes are merged, retaining connectivity information and a pointer to each observation. A dashed line is added between nodes which were observed to be unconnected in an image. In D, the connectivity information is merged. Note that where nodes were observed to be both connected and unconnected the observations are combined using log-odds and compared against a threshold, in this case resulting in the nodes being unconnected. The connectivity to a node with only one observation remains unchanged.

4.6 Multiple Wireframes and Cycle Consistency

The matching of multiple wireframes to a keyframe wireframe (solving Equation 4.1) proceeds similarly to the sGA method described in the previous chapter. As this problem is separable into multiple independent optimization problems, sGA can be applied independently to match each graph ${}^{c}\mathbf{G}$ to the keyframe graph ${}^{\text{key}}\mathbf{G}$. However, this separation does not hold when cycle consistency is taken into account. Instead the correspondence matrices must be solved for together in a single application of sGA. Despite this, only minor modifications to the sGA algorithm are required.

As before, we define the energy function as the negative of the similarity function ($E^{\text{cycle}} = -S^{\text{cycle}}$) and maximize the similarity function by minimizing the energy function. If we define an initial set of correspondence matrices as ${}^{1}\mathbf{M}^{0}...{}^{m}\mathbf{M}^{0}$ and take the first order Taylor Series approximation of this energy function we get:

$$E^{\text{cycle}}({}^{1}\mathbf{M}...{}^{m}\mathbf{M}) \approx E^{\text{cycle}}({}^{1}\mathbf{M}^{0}...{}^{m}\mathbf{M}^{0}) - \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n} {}^{c}\mathbf{Q}_{ai}({}^{c}\mathbf{M}^{0}_{ai} - {}^{c}\mathbf{M}_{ai}) \tag{4.6}$$

$${}^{c}\mathbf{Q}_{ai} = \alpha_{\text{vis}}\,{}^{c}\mathbf{\Theta}^{\text{node}}_{ai} + \sum_{b=a+1}^{{}^{\text{key}}n}\sum_{j=i+1}^{{}^{c}n} {}^{c}\mathbf{C}_{abij}\,{}^{c}\mathbf{M}^{0}_{bj} + \gamma \sum_{\{\{e,f\}\in P|e=c\}}\sum_{b=1}^{{}^{f}n}\sum_{i=1}^{{}^{\text{key}}n} {}^{ef}\mathbf{M}_{ab}\,{}^{f}\mathbf{M}^{0}_{bi} \tag{4.7}$$

s.t. $\forall ca\ \sum_{i=1}^{{}^{c}n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall ci\ \sum_{a=1}^{{}^{\text{key}}n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall cai\ {}^{c}\mathbf{M}_{ai} \in \{0,1\}$
From this we find that we can descend on our energy function by maximizing:

$$\sum_{c=1}^{m}\sum_{a=1}^{{}^{\text{key}}n}\sum_{i=1}^{{}^{c}n} {}^{c}\mathbf{Q}_{ai}{}^{c}\mathbf{M}_{ai} \tag{4.8}$$

s.t. $\forall ca\ \sum_{i=1}^{{}^{c}n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall ci\ \sum_{a=1}^{{}^{\text{key}}n} {}^{c}\mathbf{M}_{ai} \leq 1$, $\forall cai\ {}^{c}\mathbf{M}_{ai} \in \{0,1\}$

The constraints on this optimization are treated identically to sGA. A single β parameter is used across all correspondence matrices, however normalization is performed independently across each matrix. In addition the sGA algorithm from the previous chapter is modified slightly to an additive annealing parameter update rather than a multiplicative one. That is, at each iteration β is updated as $\beta = \beta + \beta_r$ rather than $\beta = \beta * \beta_r$ as this was generally shown to give better results for less computation. We denote algorithms in this chapter that optimize for cycle consistency with the suffix "-cy". The full algorithm, "sGA-cy", is shown in Algorithm 2.

4.7 Iterative Wireframe Matching

The triangulation of 3D points in the keyframe wireframe allows us to to perform wireframe matching using geometric distances between nodes. We compute this distance as the distance between the projection of the 3D points onto our camera rig and the 2D detected node positions. As our sGA algorithm uses a similarity metric, rather than a distance metric, we will convert this distance into a similarity metric by subracting the distance from a max distance and normalizing the values. Thus our final geometric similarity metric between node a in ${}^{c}\mathbf{G}$ and node i in ${}^{\text{key}}\mathbf{G}$ has the form:

$${}^{c}\delta^{\text{node}}_{ai} = \min(0, (d^{\text{geo}}_{\max} - ||({}^{c}\mathbf{z}_{a} - {}^{c}h({}^{c}\mathbf{T}_{\text{ext}}\mathbf{T}\mathbf{p}_{i}))||_2/d^{\text{geo}}_{\max}) \tag{4.9}$$

where ${}^{c}h(\mathbf{p})$ is the camera projection function incorporating the camera intrinsic matrix and distortion parameters. $\mathbf{p}_i$ is the triangulated 3D position of node i in ${}^{\text{key}}\mathbf{G}$ represented in homogeneous coordinates. ${}^{c}\mathbf{z}_a$ is the 2D position of node a in ${}^{c}\mathbf{G}$. ${}^{c}\mathbf{T}_{\text{ext}} \in SE(3)$ is the known camera extrinsic for camera c, parameterized as a 4 by 4 transformation matrix. $d^{\text{geo}}_{\max}$

Algorithm 2: Simplified Graduated Assignment For Multiple Wireframes with Cycle Consistency

$\beta \leftarrow \beta_0$

$\forall c \in m$:

$\quad {}^{c}\mathbf{M} \leftarrow {}^{c}\mathbf{\Theta}^{\text{node}}$

while $\beta < \beta_f$ **do**

$\quad \forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:

$\quad\quad {}^{c}\mathbf{Q}_{ai} \leftarrow \alpha_{\text{vis}} \, {}^{c}\mathbf{\Theta}^{\text{node}}_{ai} + \sum_{b=a+1}^{{}^{\text{key}}n} \sum_{j=i+1}^{{}^{c}n} {}^{c}\mathbf{C}_{abij} \, {}^{c}\mathbf{M}^{0}_{bj}$

$\quad\quad\quad +\gamma \sum_{\{\{e,f\} \in P | e=c\}} \sum_{b=1}^{{}^{f}n} \sum_{i=1}^{{}^{\text{key}}n} {}^{ef}\mathbf{M}_{ab} \, {}^{f}\mathbf{M}^{0}_{bi}$

$\quad \forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:

$\quad\quad {}^{c}\mathbf{M}_{ai} \leftarrow e^{\beta \, {}^{c}\mathbf{Q}_{ai}}$

$\quad \forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:

$\quad\quad {}^{c}\mathbf{M}'_{ai} \leftarrow \frac{{}^{c}\mathbf{M}_{ai}}{\sum_{j=0}^{n+1} {}^{c}\mathbf{M}_{aj}}$

$\quad \forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:

$\quad\quad {}^{c}\mathbf{M}_{ai} \leftarrow \frac{{}^{c}\mathbf{M}'_{ai}}{\sum_{b=0}^{{}^{\text{key}}n+1} {}^{c}\mathbf{M}'_{bi}}$

$\quad \beta \leftarrow \beta + \beta_r$

end

$\forall c \in m$

$\quad {}^{c}\mathbf{M} \leftarrow$greedy_assignment$({}^{c}\mathbf{M})$

is the maximum allowed distance. For those nodes where no 3D information is computed, we could chose to remove these nodes from the optimization, however this means that we have no chance of correctly matching these nodes. Instead, if a 3D position could not be found for node i in ${}^{\text{key}}\mathbf{G}$, ${}^{c}\delta^{\text{node}}_{ai}$ is set to 0 $\forall ca$. This enables us to still use the connectivity matrix to attempt to match these nodes.

In solving Equation 4.3, we draw inspiration from the Iterative Closest Point algorithm [5]. The Iterative Closest Point algorithm alternates between computing a matching between a series of points, and computing the least squares estimate of the transform given that matching. A basic form of this would be to alternate between using sGA to compute a matching using the given transform and computing the transform given this matching. This would be a direct application of ICP to multiple wireframe matching and is similar to the method proposed by Zhou et al. [109]. In this chapter we denote this algorithm "ICP-sGA".

Simplified Graduated Assignment, however, is already an iterative algorithm, and so it may be considered to perform transform estimation at each iteration of sGA. That is, for each value of β we update both the matrices ${}^{1}\mathbf{M}...{}^{m}\mathbf{M}$ and the transform estimate $\mathbf{T}$, using our current estimates of ${}^{1}\mathbf{M}...{}^{m}\mathbf{M}$ as weighted correspondences for transform estimation. We

denote this algorithm "sGA-tr". This method is akin to a weighted K-Nearest-Neighbors version of ICP where K is reduced as the number of iterations increases. This method applies some of the benefits of graduated non-convexity to the transform estimation [24] but also introduces some issues. At the beginning of sGA, when β is close to 0, all the elements of our correspondence matrices are nearly identical. This guarantees that our transform estimation will contain an extremely large number of outliers, which is known to give poor performance. Instead, we introduce a threshold t_K and only perform transform estimation using those correspondences with weight greater than this threshold. Due to the normalization applied during sGA, this effectively restricts K to a maximum value of $1/t_k$. As a result, transform estimation will not be performed until β is large enough. To further increase robustness we utilize a robust cauchy-loss function and require that a minimum number of correspondences t_n must have a weight greater than t_k before transform estimation can be performed. Thus, our transform estimation takes the form:

If $(\sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n}\mathbb{1}({}^{c}\mathbf{M}_{ai} > t_k)) > t_n$:

$$\min_{\mathbf{T}} E^{\text{trans}}(\mathbf{T}) = \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n}\log(1 - \mathbb{1}({}^{c}\mathbf{M}_{ai} > t_k)\,{}^{c}\delta_{ai}^{\text{node}}(\mathbf{T})\,{}^{c}\mathbf{M}_{ai}) \tag{4.10}$$

s.t. $\mathbf{T} \in SE(3)$

This problem is minimized using iterative linear approximation descent (Levenberg-Marquardt). Note that the optimization over $\mathbf{T}$ must be handled carefully as $\mathbf{T}$ is represented as a 4 by 4 matrix but the space of $SE(3)$ only has 6 degrees of freedom [67]. We perform this minimization in C++ using Ceres-Solver [2].

It turns out that a combination of these methods outperforms either method individually. In this combined method, we include the transformation estimation in the sGA loop, use the final matching to perform another transform estimation step, and then repeat sGA from scratch using this new transform estimate. This is repeated until convergence or a maximum number of iterations is reached. This joint method we term "Iterative Wireframe Matching" or "IWM". The full algorithm is shown in Aglorithm 3 and a visualization of this process is shown in Figure 4.1.

The same modifications from Section 4.6 can be used to additionally optimize for cycle consistency.

4.8 Joint Iterative Wireframe Matching

As may be expected, when both visual and geometric information is available, it is possible to combine both visual and geometric similarity into a single optimization problem and thereby improve the quality of the matching. The joint algorithm is shown in Algorithm 4. This joint algorithm changes very little from the algorithm in the previous section (Algorithm 3). We combine the terms for node similarity and trade off these terms using weighting parameters. We also include the terms for cycle consistency. Despite small size of these

Algorithm 3: Simplified Graduated Assignment For Iterative Wireframe Matching

$it \leftarrow 0$
while $it < it_{max}$ **do**
  $\beta \leftarrow \beta_0$
  $\forall c \in m$:
    ${}^{c}\mathbf{M} \leftarrow {}^{c}\mathbf{\Theta}^{\text{node}}$
  while $\beta < \beta_f$ **do**
    $\forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:
      ${}^{c}\mathbf{Q}_{ai} \leftarrow \alpha_{\text{geo}} \, {}^{c}\delta^{\text{node}}_{ai}(\mathbf{T}) + \sum_{b=a+1}^{{}^{\text{key}}n} \sum_{j=i+1}^{{}^{c}n} {}^{c}\mathbf{C}_{abij} \, {}^{c}\mathbf{M}^{0}_{bj}$
    $\forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:
      ${}^{c}\mathbf{M}_{ai} \leftarrow e^{\beta \, {}^{c}\mathbf{Q}_{ai}}$
    $\forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:
      ${}^{c}\mathbf{M}'_{ai} \leftarrow \frac{{}^{c}\mathbf{M}_{ai}}{\sum_{j=0}^{n+1} {}^{c}\mathbf{M}_{aj}}$
    $\forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:
      ${}^{c}\mathbf{M}_{ai} \leftarrow \frac{{}^{c}\mathbf{M}'_{ai}}{\sum_{b=0}^{{}^{\text{key}}n+1} {}^{c}\mathbf{M}'_{bi}}$
    if $(\sum_{c=1}^{m} \sum_{a=1}^{{}^{c}n} \sum_{i=1}^{{}^{key}n} \mathbb{1}({}^{c}\mathbf{M}_{ai} > t_k)) > t_n$ **then**
      $\min_{\mathbf{T}} E^{\text{trans}}(\mathbf{T}) = \sum_{c=1}^{m} \sum_{a=1}^{{}^{c}n} \sum_{i=1}^{{}^{\text{key}}n} \log(1 - \mathbb{1}({}^{c}\mathbf{M}_{ai} > t_k) \, {}^{c}\delta^{\text{node}}_{ai}(\mathbf{T}) \, {}^{c}\mathbf{M}_{ai})$
    end
    $\beta \leftarrow \beta + \beta_r$
  end
  $\forall c \in m$
    ${}^{c}\mathbf{M} \leftarrow$greedy_assignment$({}^{c}\mathbf{M})$
  $\min_{\mathbf{T}} E^{\text{trans}}(\mathbf{T}) = \sum_{c=1}^{m} \sum_{a=1}^{{}^{c}n} \sum_{i=1}^{{}^{\text{key}}n} \log(1 - \; {}^{c}\delta^{\text{node}}_{ai}(\mathbf{T}) \, {}^{c}\mathbf{M}_{ai})$
  $it \leftarrow it + 1$
end

changes, the impact on how the algorithm behaves may be significant. Specifically, the use of a visual similarity metric that does not depend on the transform between camera rigs may result in a matching that is not significantly changed with the transform. This can result in a local convexity that is very large with respect to the transform estimate and thus make the impact of transform estimation minimal. Performing the transform estimation inside the graduated non-convexity loop, however, may allow the algorithm to escape this local minima. As such we evaluate separately the performance of this combined similarity metric using each of the proposed geometric matching algorithms. We denote these

algorithms "vICP-sGA", "vsGA-tr", and "vIWM" respectively. When cycle consistency is introduced, these algorithms become "vICP-sGA-cy", "vsGA-tr-cy" and "Joint IWM" or "JIWM" respectively.

Here the choice of weighting parameter to trade off between the two similarity metrics is likely significant. While we simply determined these weighting parameters manually, it has been proposed that nonlinear inverse optimization may be used to learn these parameters from a given dataset [89].

4.9 Experiments

We evaluate our algorithms on two sequences from the EUROC MAV [8] dataset: "MH_01_easy" and "V1_01_easy", along with the first sequence from the KITTI [20] dataset: "00". These scenes utilize multiple (two) camera rigs and provide ground truth trajectory information for these rigs. We utilize the ground truth trajectories to compute the ground truth Fundamental Matrix and count the number of inlier matches whose Symmetric Epipolar Distance falls below a threshold. While this is not an exact measure of correct matches, it allows us to estimate the average improvement in the number of correct matches over the baseline. For each scene we sample two images from every 10 images. We test the results of our matching algorithms under varying temporal distance between images. Specifically for the EUROC sequences, which are captured at 20Hz, we test images that are 1, 5 and 10 images apart. For the KITTI 00 sequence, which is captured at 10Hz, we test images that are 1, 3, and 5 images apart. This separation between images is denoted "shift" in our results.

We perform three experiments on these datasets. In the first experiment, denoted "Cycle Matching" we evaluate the benefit of adding a cycle consistency cost to the sGA algorithm. That is, we evaluate the benefit of solving Equation 4.2 versus solving Equation 4.1 by evaluating the number of inliers induced by applying sGA to each problem.

In the second experiment, denoted "Geometric Matching", we evaluate methods to utilize a geometric cost term (Equation 4.9) for wireframe matching. Specifically we compare the three proposed methods of solving Equation 4.3 ("ICP-sGA","sGA-tr", and the combined method "IWM") to the standard ICP method (solving Equation 4.4). Again, we evaluate the number of inliers induced by these algorithms.

Finally, in the third experiment, denoted "Joint Matching" we evaluate how visual and geometric cost terms can be combined to improve matching. In this, we compare permutations of the above algorithms to solve Equation 4.5 (either with our without the cycle consistency term) against simply using a modified cost term for ICP that utilizes the visual similarity as well as the geometric similarity for the matching step.

It should be noted that the EUROC MAV sequences are captured in environments (a large machine shop and a room with mattress lines walls), that are not well represented by the training data of our wireframe detection network. As a result, we may expect to see reduced performance gains as compared to the KITTI sequence which contains outdoor urban environments. The KITTI dataset, however, contains larger transformations between

Algorithm 4: Simplified Graduated Assignment For Joint Iterative Wireframe Matching

$it \leftarrow 0$

while $it < it_{max}$ **do**

$\beta \leftarrow \beta_0$

$\forall c \in m$:

${}^{c}\mathbf{M} \leftarrow {}^{c}\mathbf{\Theta}^{\text{node}}$

while $\beta < \beta_f$ **do**

$\forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:

$${}^{c}\mathbf{Q}_{ai} \leftarrow {}^{c}(\alpha_{\text{vis}}\mathbf{\Theta}^{\text{node}}_{ai} + \alpha_{\text{geo}}\delta^{\text{node}}_{ai}(\mathbf{T})) + \sum_{b=a+1}^{{}^{\text{key}}n}\sum_{j=i+1}^{{}^{c}n} {}^{c}\mathbf{C}_{abij}\,{}^{c}\mathbf{M}^{0}_{bj} + \gamma \sum_{\{\{e,f\}\in P|e=c\}}\sum_{b=1}^{{}^{f}n}\sum_{i=1}^{{}^{\text{key}}n} {}^{ef}\mathbf{M}_{ab}\,{}^{f}\mathbf{M}^{0}_{bi}$$

$\forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:

$${}^{c}\mathbf{M}_{ai} \leftarrow e^{\beta\,{}^{c}\mathbf{Q}_{ai}}$$

$\forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:

$${}^{c}\mathbf{M}'_{ai} \leftarrow \frac{{}^{c}\mathbf{M}_{ai}}{\sum_{j=0}^{n+1}{}^{c}\mathbf{M}_{aj}}$$

$\forall c \in m \, \forall a \in {}^{\text{key}}n \, \forall i \in n$:

$${}^{c}\mathbf{M}_{ai} \leftarrow \frac{{}^{c}\mathbf{M}'_{ai}}{\sum_{b=0}^{{}^{\text{key}}n+1}{}^{c}\mathbf{M}'_{bi}}$$

if $(\sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{key}n} \mathbb{1}({}^{c}\mathbf{M}_{ai} > t_k)) > t_n$ **then**

$$\min_{\mathbf{T}} E^{\text{trans}}(\mathbf{T}) = \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n} \log(1 + \mathbb{1}({}^{c}\mathbf{M}_{ai} > t_k)\,{}^{c}\delta^{\text{node}}_{ai}(\mathbf{T})\,{}^{c}\mathbf{M}_{ai})$$

end

$\beta \leftarrow \beta + \beta_r$

end

$\forall c \in m$

${}^{c}\mathbf{M} \leftarrow$greedy_assignment$({}^{c}\mathbf{M})$

$$\min_{\mathbf{T}} E^{\text{trans}}(\mathbf{T}) = \sum_{c=1}^{m}\sum_{a=1}^{{}^{c}n}\sum_{i=1}^{{}^{\text{key}}n} \log(1 + {}^{c}\delta^{\text{node}}_{ai}(\mathbf{T})\,{}^{c}\mathbf{M}_{ai})$$

$it \leftarrow it + 1$

end

images, which may reduce performance of our Geometric Matching algorithms. The rgb images in KITTI 00 are rectified to have no lens distortion. The images of the EUROC sequences are not rectified but the distortion parameters are known. For these sequences wireframe detection is performed on the distorted images and the junction positions are undistorted for use in Geometric and Joint Matching experiments. These sequences used

together represent a range of possible situations for the evaluation of our algorithms.

As mentioned in Section 4.6, we have modified sGA to use an additive annealing parameter update. As a result the new β_r will vary slightly from the original result. In this work we set $\beta_r = 0.5$. Note that this results in a little over twice as many iterations as compared to the previous chapter. This is to ensure that the Geometric Matching methods have enough iterations to compute the transformation when this transformation is large. If, the transformation is known to be small or the geometric matching is not used, β_r may be increased, reducing the number of iterations. For consistency these parameters are used across all experiments in this chapter. The remaining parameters, which are either new or unchanged from the previous chapter, are as follows: $\beta_0 = 1.0$, $\beta_f = 10$ $\gamma = 1.0$, $\alpha_{\text{vis}} = 1.0$, $t_k = 0.2$,$t_n = 10$. For Geometric Matching Experiments α_{geo} is set to 1.0, however for Joint Matching Experiments α_{geo} is set to 0.5 (except for "vis-ICP" which had the best performance at $\alpha_{\text{geo}} = 0.3$). For both Geometric and Joint matching experiments $d^{\text{geo}}_{\max}$ is set to 320 pixels and it_{max} is set to 15.

For all experiments where visual similarity is used, we employ BRISK [51] feature descriptors extracted around the image location of each node. To compute image similarity we use the function: ${}^{c}\mathbf{\Theta}^{\text{node}}_{ai} = (d^{\text{vis}}_{\max} - {}^{c}D^{\text{node}}_{ai})/d^{\text{vis}}_{\max}$ where ${}^{c}D^{\text{node}}_{ai}$ is the hamming distance between descriptors of node a in ${}^{c}\mathbf{G}$ and node i in ${}^{\text{key}}\mathbf{G}$ and $d^{\max}$ is set to 200. For the keyframe wireframe where nodes have multiple descriptors associated with them, only one descriptor is used.

4.10 Results and Discussion

For each algorithm the average percent improvement is calculated as: $\frac{1}{K}\sum_{k=1}^{K}\frac{\iota_k - \iota_k^{\#}}{\iota_k^{\#}}$ where K is the number of sample pairs in the sequence, ι_k is the total number of inliers found by the algorithm for pair k, and $\iota_k^{\#}$ is the total number of inliers found by the baseline algorithm for the same pair. Dividing by the number of inliers found in the baseline algorithm is an attempt to normalize for the difficulty of the matching. For reference we also present the average number of inliers found by each algorithm. To remove scenes with heavy motion blur or insufficient visual texture, we do not consider images where the baseline method fails to find at least 20 total inliers across both images in the camera rig.

Cycle Matching Experiments

The percent improvement and average inlier results for the Cycle Matching experiments are shown in Figure 4.3 and Table 4.1 respectively. We observe that the addition of the cycle consistency terms to the optimization problem does increase performance as compared to the standard sGA result. Across all sequences, the impact of including cycle consistency terms increases with the separation between images, where the matching is more difficult. We also see a greater impact of cycle consistency on the KITTI sequence as compared to the EUROC sequences. As the KITTI sequence is better represented by our wireframe network

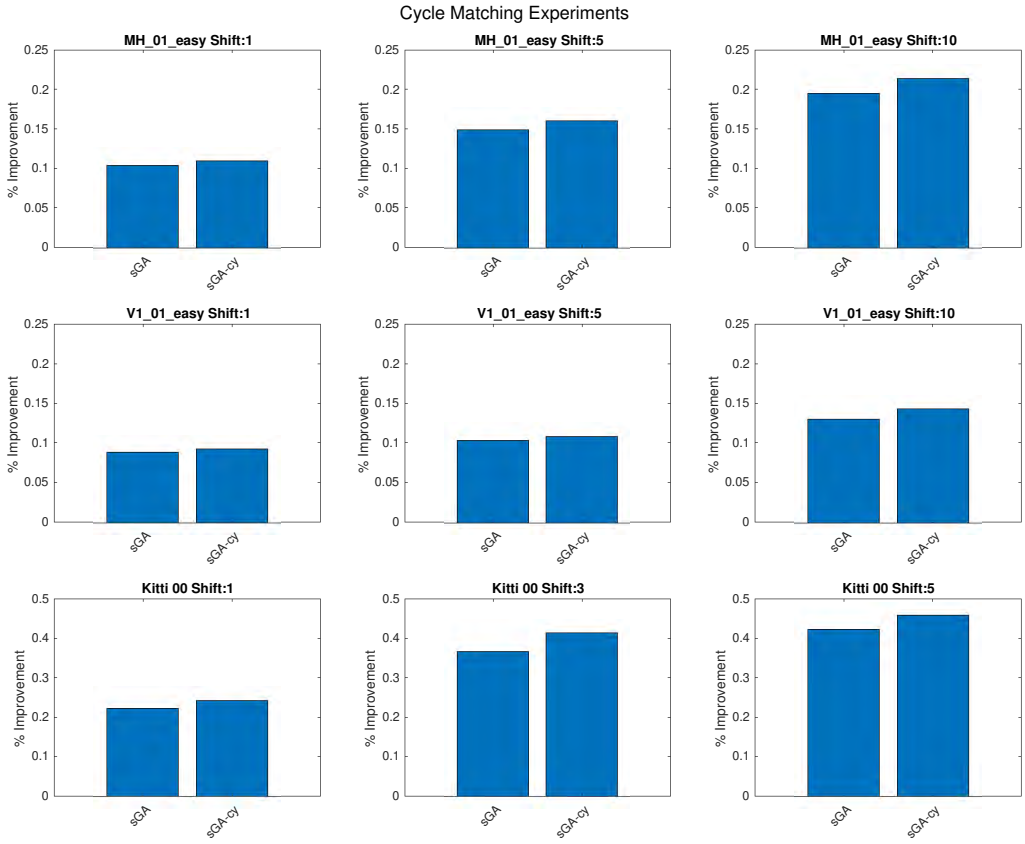


Figure 4.3: Average percent improvement of inliers for wireframe matching algorithms on three sequences for varying levels of image separation ("shift"). Each bar represents the improvement of the associated algorithm over a matching that only uses visual feature similarity for feature matching. "sGA" is the simplified Graduated Assignment method from Chapter 3. "sGA-cy" is the proposed method of optimizing for cycle consistency across multiple images.

training dataset, this increased impact may be the result of more accurate detection of wireframes by the network. An alternate explanation would be the increased lighting changes of the outdoor scene cause reduced performance of visual matching, making the structural wireframe detection more important. Either of these explanations is also consistent with the increased performance difference between sGA and the baseline visual matching on the KITTI sequence. The average frames per second (FPS) for each algorithm is shown in Table 4.2. We observe here that, using our algorithm, adding cycle consistency terms does very little to impact average run time of the sGA algorithm, making this a practical way to optimize for this constraint. Note that this does not include the cost to compute pairwise

Cycle Matching Results

dataset	shift	vis	sGA	sGA-cy
MH_01_easy	1	177.12	194.56	**195.54**
MH_01_easy	5	147.36	166.88	**168.42**
MH_01_easy	10	128.14	149.68	**151.64**
V1_01_easy	1	110.76	120.12	**120.54**
V1_01_easy	5	79.56	87.44	**87.93**
V1_01_easy	10	64.00	71.70	**72.50**
Kitti 00	1	106.25	128.74	**130.64**
Kitti 00	3	67.77	90.65	**93.34**
Kitti 00	5	63.75	88.38	**90.56**

Table 4.1: Average number of inliers for wireframe matching algorithms on three sequences for varying levels of image separation ("shift"). "vis" represents a matching that only uses visual feature similarity for feature matching. "sGA" is the simplified Graduated Assignment method from Chapter 3. "sGA-cy" is the proposed method of optimizing for cycle consistency across multiple images. The best results for each sequence are shown in bold.

matches as this will depend on the algorithm used, however example pairwise sGA timing results are given in the previous chapter.

Geometric Matching Experiments

Figure 4.4 and Table 4.3 show the percent improvement and average inlier results for the Geometric Matching experiments. We observe that all of the proposed geometric matching algorithms are able to achieve highly significant performance increases as compared to the direct application of the Iterative Closest Point algorithm. This may be unsurprising in a few ways. First, sparse feature matching is not a good candidate application for Iterative Closest Point as the sparsity of the problem induces very large local concavities. Second, the need for features to be triangulated before a geometric distance can be computed means that those features which could not be triangulated have no chance of being matched correctly by the standard ICP algorithm. Our algorithms overcome these difficulties by optimizing for edge consistency, a metric of similarity independent of transform. This both increases the likelihood that the algorithm will converge to a good local optimum, but also provides a metric of similarity to those features that would not have a similarity metric otherwise.

We see that IWM further outperforms the separate methods ICP-sGA and sGA-tr. As with the cycle consistency, this improvement increases with image separation hinting that the combined method is more important as the difficulty of the matching increases. We also again see a greater improvement on the KITTI dataset. This may be the result of the

Algorithm Runtime (FPS) Results

Algorithm	MH_01_easy	V1_01_easy	Kitti 00
vis	624.33	1184.50	621.99
sGA	29.70	65.32	26.16
sGA-cy	27.27	60.50	24.62
ICP	55.56	123.06	53.71
sGA-tr	26.09	51.94	22.91
ICP_sGA	2.65	10.31	4.04
IWM	2.33	9.13	3.84
vICP	123.67	385.28	169.10
vICP-sGA	4.25	13.85	6.75
vsGA-tr	23.81	48.36	21.47
vIWM	4.38	14.35	6.53
ICP-sGA-cy	3.90	15.27	6.16
vsGA-tr-cy	22.39	45.42	20.36
JIWM	3.97	13.31	6.20

Table 4.2: Average FPS for wireframe matching algorithms on three sequences. This average is computed across all images and all shift values for a given sequence.

larger transformations between images on the KITTI dataset requiring a larger region of convergence than what is provided by ICP-sGA or sGA-tr individually.

The average FPS for these algorithms is shown in Table 4.2. We observe that adding transform estimation in the form of sGA-tr has very little impact on the run time as compared to the standard sGA method. Note however that the chosen parameters for sGA have been increased to enable sGA-tr to perform well and the standard sGA method can perform well at over twice this speed if the parameters are changed. sGA-tr notably still maintains realtime performance for these datasets. The IWM method, though having the best accuracy of these three algorithms, also has the slowest run time as it performs sGA-tr repeatedly.

Joint Matching Experiments

Figure 4.5 and Table 4.4 show the percent improvement and average inlier results for the Joint Matching experiments. In these results we observe that direct application of ICP using a combined visual and geometric similarity term does yield a small improvement over just using visual similarity for matching. This improvement is significantly increased, however, by the introduction of sGA. As before the IWM (vIWM and JIWM) methods generally outperform the ICP-sGA (vICP-sGA and vICP-sGA-cy) and sGA-tr (vsGA-tr and vsGA-tr-cy) methods, with this improvement increasing with image separation. Similarly the methods

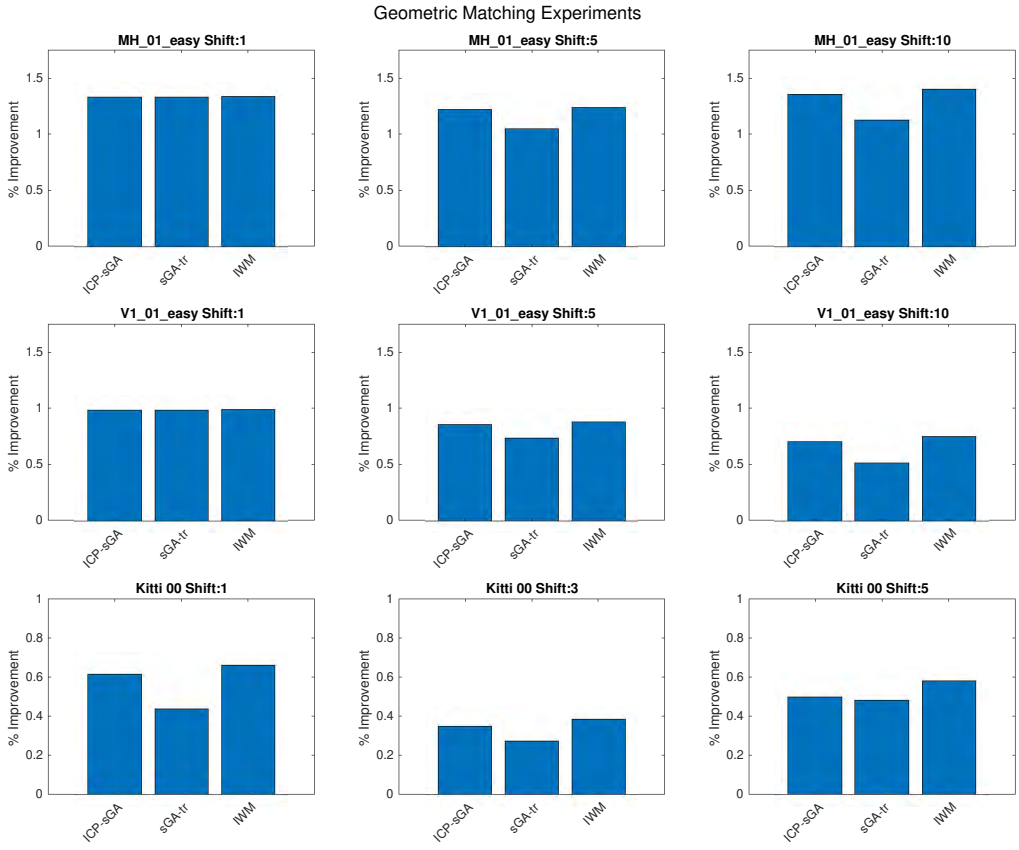


Figure 4.4: Average percent improvement of inliers for wireframe matching algorithms on three sequences for varying levels of image separation ("shift"). Each bar represents the improvement of the associated algorithm over a standard ICP result. "ICP-sGA" uses sGA for the association of points in the ICP loop rather than nearest-neighbors. "sGA-tr" is a modified sGA algorithm which optimizes for the transform inside the sGA algorithm. "IWM" is the proposed algorithm combining of ICP-sGA and ICP-tr.

which enforce cycle consistency outperform those that do not, with this improvement increasing with image separation. However, unlike the Geometric Matching experiments, the sGA-tr methods here outperform the ICP-sGA methods. In investigating this we observed that ICP-sGA converged very quickly (often in only two iterations). This likely indicated that the matching resulting from visual similarity terms often push ICP-sGA directly to a local optimum that is difficult to escape from. The inclusion of transform estimation inside sGA, however, allows transform estimation to be performed when the assignment matrix is smoother, potentially resulting in a better local optimum.

The average FPS for these algorithms is shown in Table 4.2. Notably the vIWM and

Geometric Matching Results

dataset	shift	ICP	ICP-sGA	sGA-tr	IWM
MH_01_easy	1	49.46	122.28	122.33	**122.63**
MH_01_easy	5	38.71	101.79	91.95	**103.26**
MH_01_easy	10	29.73	84.62	71.57	**86.78**
V1_01_easy	1	42.05	89.95	89.94	**90.19**
V1_01_easy	5	28.28	64.80	58.72	**66.08**
V1_01_easy	10	20.74	49.58	41.22	**50.45**
Kitti 00	1	49.28	81.65	72.98	**83.84**
Kitti 00	3	27.01	41.93	38.47	**42.72**
Kitti 00	5	23.60	42.23	40.37	**42.64**

Table 4.3: Average number of inliers for wireframe matching algorithms on three sequences for varying levels of image separation ("shift"). "ICP" is the result of the direct application of the Iterative Closest Point algorithm to wireframe matching. "ICP-sGA" uses sGA for the association of points in the ICP loop rather than nearest-neighbors. "sGA-tr" is a modified sGA algorithm which optimizes for the transform inside the sGA algorithm. "IWM" is the proposed algorithm combining of ICP-sGA and ICP-tr.

.

JIWM methods have faster runtimes than IWM. This is due to the aforementioned decrease in the number of outer loop iterations performed by these methods. When realtime performance is required the vsGA-tr-cy method is likely the method of choice due to the relatively low added cost and high matching accuracy. When realtime performance is not required and the importance of matching correctly is high, the JIWM method remains a viable solution.

4.11 Conclusion

We present a series of algorithms for matching wireframes across multi-camera rigs. These algorithms intelligently incorporate known extrinsics calibration information to improve wireframe matching results. These algorithms are capable of optimizing cycle consistency across pairs of overlapping images, optimizing geometric similarity, and jointly optimizing visual and geometric similarities. By comparing permutations of these components under realistic image sequences, we are able to evaluate the benefit of each component. In the future we hope to utilize the algorithms presented in this chapter to perform large scale wireframe reconstructions.

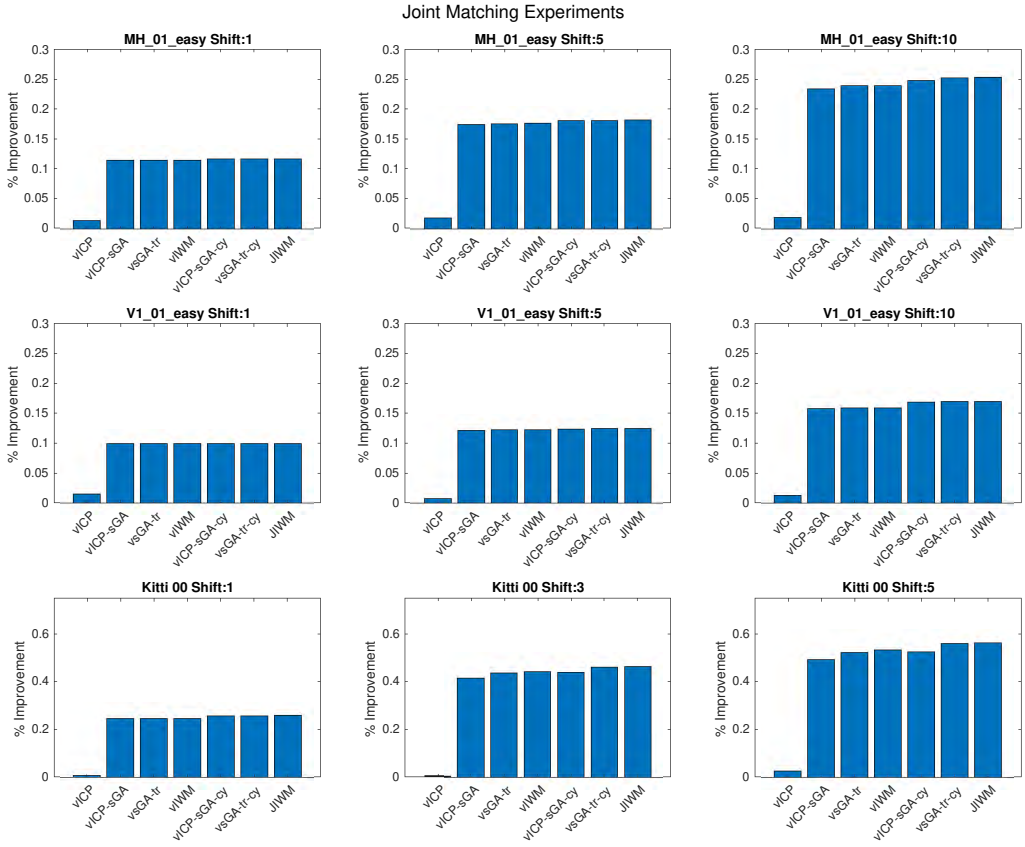


Figure 4.5: Average percent improvement of inliers for wireframe matching algorithms on three sequences for varying levels of image separation ("shift"). Each bar represents the improvement of the associated algorithm over a matching that only uses visual feature similarity for feature matching. "vICP" is the direct application of Iterative Closest Point using the joint similarity term. "vICP-sGA" uses sGA for the association of points in the ICP loop rather than nearest-neighbors. "vsGA-tr" is a modified sGA algorithm which optimizes for the transform inside the sGA algorithm. "vIWM" is the combination of vICP-sGA and vsGA-tr. "vICP-sGA-cy" is vICP-sGA with the additional optimization of cycle consistency. "vsGA-tr-cy" is vsGA-tr with the additional optimization of cycle consistency. "JIWM" is the proposed joint matching algorithm which is vIWM with the additional optimization of cycle consistency.

Joint Matching Results

dataset	shift	vis	vICP	vICP-sGA	vsGA-tr	vIWM	vICP-sGA-cy	vsGA-tr-cy	JIWM
MH_01_easy	1	177.12	179.72	196.25	196.26	196.25	**196.68**	**196.68**	**196.68**
MH_01_easy	5	147.36	149.89	169.65	169.78	169.84	170.50	170.56	**170.60**
MH_01_easy	10	128.14	130.41	153.41	153.88	153.88	154.83	155.15	**155.21**
V1_01_easy	1	110.76	112.32	121.24	121.23	121.25	121.28	**121.29**	121.28
V1_01_easy	5	79.56	80.23	88.83	88.88	88.88	89.00	**89.04**	89.03
V1_01_easy	10	64.00	64.82	73.26	73.38	73.35	73.83	73.96	**73.99**
Kitti 00	1	106.25	107.48	131.05	131.21	131.24	132.12	132.35	**132.40**
Kitti 00	3	67.77	68.61	93.48	94.64	95.08	95.12	96.18	**96.44**
Kitti 00	5	63.75	65.79	91.97	93.71	94.05	94.04	95.75	**96.01**

Table 4.4: Average number of inliers for wireframe matching algorithms on three sequences for varying levels of image separation ("shift"). "vis" represents a matching that only uses visual feature similarity for feature matching. "vICP" is the direct application of Iterative Closest Point using the joint similarity term. "vICP-sGA" uses sGA for the association of points in the ICP loop rather than nearest-neighbors. "vsGA-tr" is a modified sGA algorithm which optimizes for the transform inside the sGA algorithm. "vIWM" is the combination of vICP-sGA and vsGA-tr. "vICP-sGA-cy" is vICP-sGA with the additional optimization of cycle consistency. "vsGA-tr-cy" is vsGA-tr with the additional optimization of cycle consistency. "JIWM" is the proposed joint matching algorithm which is vIWM with the additional optimization of cycle consistency. The best results for each sequence are shown in bold.

Chapter 5

Final Remarks

This thesis has identified and begun to address several problems relating to Simultaneous Localization and Mapping (SLAM) and 3D reconstruction in the context of Augmented Reality. These problems include the lack of features in low texture environments, changing features due to lighting conditions and dynamic objects, and the inefficient representation of 3D models in the scene.

By utilizing planar structures identified in depth images, we are able to incorporate depth information efficiently and effectively to improve performance in low-texture environments and reduce the likelihood of tracking failure in these situations. This algorithm was specifically designed to have minimal impact on the amount of required computation, while still improving accuracy under the limited depth sensing range of mobile time of flight sensors. While this is a step towards reducing the failure cases of SLAM in structured environments, to reduce these cases completely it is likely that longer range depth sensors will need to be used, and additional types of structure incorporated into our estimation problem.

This thesis further considers the image wireframe as a type of structure for improving SLAM and 3D reconstruction. The image wireframe represents those lines and their intersections that make up the structural components of the scene. A neural network is used to detect these lines and intersections, which has demonstrated high precision and recall for this task. A high precision indicates that wireframe features are less likely to be the result of non-structural components such as shadows or creases in fabric. As a result, the use of these features may be a solution to SLAM failures that occur when normal point features change over time. A high recall indicates that the majority of the structural components in the scene are identified by the neural network. This indicates that the wireframe can be used as a compact representation of the overall structure of the scene.

A requirement for enabling both these tasks is the accurate matching of wireframes across images. This thesis presents a method for accurately and efficiently matching wireframes across images by exploiting the connectivity information provided by the image wireframe. This method is extended to provide increased accuracy using the additional information provided by multiple cameras when available. It is also shown how geometric constraints can be used in place of visual similarity when visual similarity in unreliable, such as in the

case of a light source attached to the camera device.

This thesis additionally presents a basic method for merging image wireframes into estimated 3D wireframes given a matching between the junctions in the wireframe. It remains to be demonstrated, however, the accuracy of a large scale wireframe reconstruction that would result from this method. It is also open to future work how to convert an estimated 3D wireframe into a representation usable by Augmented Reality devices. We hypothesize that minimum clique finding in the graph representation of the wireframe, followed by Delauny Triangulation run on each clique would represent an efficient method of converting a 3D wireframe into a compact mesh representation commonly used in Augmented Reality. We hope to address this in future work.

Lastly, it is important to note that while the image wireframe detection network has both high precision and recall, it is by no means perfect. As there are inherent ambiguities in the information about structure provided by a single monocular camera image, the network must rely on some other context clues in the image to determine when structure is likely present in the scene. The depth image, however, provides significantly more information about the 3D structure of the scene. As we look towards the future, it is likely that best solutions will result from the combination of depth and traditional RGB cameras in a big data framework. In looking towards the goal of ubiquitous Augmented Reality we continue to work towards the integration of these systems in order to enable the reliable detection of structure in a variety of environments.

Bibliography

[1] Ibrahim SI Abuhaiba. "Offline signature verification using graph matching". In: *Turkish Journal of Electrical Engineering & Computer Sciences* 15.1 (2007), pp. 89–104.

[2] Sameer Agarwal, Keir Mierle, et al. *Ceres Solver*. `http://ceres-solver.org`.

[3] HA Almohamad and Salih O Duffuaa. "A linear programming approach for the weighted graph matching problem". In: *IEEE Transactions on pattern analysis and machine intelligence* 15.5 (1993), pp. 522–525.

[4] Joan Batlle, E Mouaddib, and Joaquim Salvi. "Recent progress in coded structured light as a technique to solve the correspondence problem: a survey". In: *Pattern recognition* 31.7 (1998), pp. 963–982.

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

[6] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. "Sparse iterative closest point". In: *Computer graphics forum*. Vol. 32. 5. Wiley Online Library. 2013, pp. 113–123.

[7] Yuri Boykov, Olga Veksler, and Ramin Zabih. "Fast approximate energy minimization via graph cuts". In: *IEEE Transactions on pattern analysis and machine intelligence* 23.11 (2001), pp. 1222–1239.

[8] Michael Burri et al. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* (2016). DOI: `10.1177/0278364915620033`. eprint: `http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html`. URL: `http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract`.

[9] Michael Calonder et al. "Brief: Binary robust independent elementary features". In: *European conference on computer vision*. Springer. 2010, pp. 778–792.

[10] Andrea Corti et al. "A metrological characterization of the Kinect V2 time-of-flight camera". In: *Robotics and Autonomous Systems* 75 (2016), pp. 584–594.

[11] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. "Balanced graph matching". In: *Advances in Neural Information Processing Systems*. 2007, pp. 313–320.

[12] Cecilia Di Ruberto. "Recognition of shapes by attributed skeletal graphs". In: *Pattern Recognition* 37.1 (2004), pp. 21–31.

[13] Tom van Dijk and Guido de Croon. "How Do Neural Networks See Depth in Single Images?" In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2183–2191.

[14] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *European Conference on Computer Vision*. Springer, 2014, pp. 834–849. DOI: 10.1007/978-3-319-10605-2_54.

[15] Ali Erol et al. "Vision-based hand pose estimation: A review". In: *Computer Vision and Image Understanding* 108.1-2 (2007), pp. 52–73.

[16] Renato F. Salas-Moreno et al. "Dense planar SLAM". In: Sept. 2014, pp. 157–164. DOI: 10.1109/ISMAR.2014.6948422.

[17] Sergi Foix, Guillem Alenya, and Carme Torras. "Lock-in time-of-flight (ToF) cameras: A survey". In: *IEEE Sensors Journal* 11.9 (2011), pp. 1917–1926.

[18] W. Förstner and K. Khoshelham. "Efficient and Accurate Registration of Point Clouds with Plane to Plane Correspondences". In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. Oct. 2017, pp. 2165–2173. DOI: 10.1109/ICCVW.2017.253.

[19] Dorian Gálvez-López and J. D. Tardós. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Transactions on Robotics* 28.5 (Oct. 2012), pp. 1188–1197. ISSN: 1552-3098. DOI: 10.1109/TRO.2012.2197158.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[21] Patrick Geneva et al. "LIPS: LiDAR-Inertial 3D Plane SLAM". In: Oct. 2018. DOI: 10.1109/IROS.2018.8594463.

[22] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. "Unsupervised Monocular Depth Estimation With Left-Right Consistency". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[23] Steven Gold and Anand Rangarajan. "A graduated assignment algorithm for graph matching". In: *IEEE Transactions on pattern analysis and machine intelligence* 18.4 (1996), pp. 377–388.

[24] Steven Gold et al. "New algorithms for 2D and 3D point matching: pose estimation and correspondence". In: *Pattern recognition* 31.8 (1998), pp. 1019–1031.

[25] Chao X Guo et al. "Efficient Visual-Inertial Navigation using a Rolling-Shutter Camera with Inaccurate Timestamps." In: *Robotics: Science and Systems*. Citeseer. 2014.

[26] Chao Guo et al. "Efficient Visual-Inertial Navigation using a Rolling-Shutter Camera with Inaccurate Timestamps". In: July 2014. DOI: 10.15607/RSS.2014.X.057.

[27] Osian Haines and Andrew Calway. "Estimating Planar Structure in Single Images by Learning from Examples." In: *ICPRAM (2)*. 2012, pp. 289–294.

[28] Lei He, Guanghui Wang, and Zhanyi Hu. "Learning depth from single images with deep neural network embedding focal length". In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4676–4689.

[29] Martial Hebert. "Active and passive range sensing for robotics". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE. 2000, pp. 102–110.

[30] Joshua Hernandez, Konstantine Tsotsos, and Stefano Soatto. "Observability, identifiability and sensitivity of vision-aided inertial navigation". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2319–2325.

[31] Wolfgang Hess et al. "Real-Time Loop Closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278.

[32] Armin Hornung et al. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees". In: *Autonomous Robots* (2013). Software available at `http://octomap.github.com`. DOI: `10.1007/s10514-012-9321-0`. URL: `http://octomap.github.com`.

[33] M. Hsiao et al. "Keyframe-based dense planar SLAM". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 5110–5117. DOI: `10.1109/ICRA.2017.7989597`.

[34] Ming Hsiao, Eric Westman, and Michael Kaess. "Dense Planar-Inertial SLAM with Structural Constraints". In: May 2018, pp. 6521–6528. DOI: `10.1109/ICRA.2018.8461094`.

[35] Kun Huang et al. "Learning to parse wireframes in images of man-made environments". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 626–635.

[36] Shahram Izadi et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011, pp. 559–568.

[37] David Jiménez et al. "Modeling and correction of multipath interference in time of flight cameras". In: *Image and Vision Computing* 32.1 (2014), pp. 1–13.

[38] Salim Jouili, Ines Mili, and Salvatore Tabbone. "Attributed graph matching using local descriptions". In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer. 2009, pp. 89–99.

[39] Salim Jouili and Salvatore Tabbone. "Graph matching based on node signatures". In: *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer. 2009, pp. 154–163.

[40] Michael Kaess. "Simultaneous localization and mapping with infinite planes". In: *Proceedings - IEEE International Conference on Robotics and Automation* 2015 (June 2015), pp. 4605–4611. DOI: 10.1109/ICRA.2015.7139837.

[41] Bernd Kitt, Andreas Geiger, and Henning Lategahn. "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme". In: *2010 ieee intelligent vehicles symposium*. IEEE. 2010, pp. 486–492.

[42] Georg Klein and David Murray. "Parallel tracking and mapping for small AR workspaces". In: *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE. 2007, pp. 225–234.

[43] Vladimir Kolmogorov and Ramin Zabih. "Computing visual correspondence with occlusions using graph cuts". In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 2. IEEE. 2001, pp. 508–515.

[44] Vladimir Kolmogorov and Ramin Zabih. "Multi-camera scene reconstruction via graph cuts". In: *European conference on computer vision*. Springer. 2002, pp. 82–96.

[45] Vladimir Kolmogorov and Ramin Zabin. "What energy functions can be minimized via graph cuts?" In: *IEEE transactions on pattern analysis and machine intelligence* 26.2 (2004), pp. 147–159.

[46] Dimitrios G. Kottas and Stergios I. Roumeliotis. "Efficient and consistent vision-aided inertial navigation using line observations". In: IEEE, May 2013, pp. 1540–1547. DOI: 10.1109/ICRA.2013.6630775.

[47] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[48] Robert Lange et al. "Time-of-flight range imaging with a custom solid state image sensor". In: *Laser Metrology and Inspection*. Vol. 3823. International Society for Optics and Photonics. 1999, pp. 180–191.

[49] Margarita Chli Laurent Kneip and Roland Siegwart. "Robust Real-Time Visual Odometry with a Single Camera and an IMU". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 16.1–16.11. DOI: 10.5244/C.25.16.

[50] D Khuê Lê-Huu and Nikos Paragios. "Alternating direction graph matching". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 4914–4922.

[51] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. "BRISK: Binary robust invariant scalable keypoints". In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2548–2555.

[52] Stefan Leutenegger et al. "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[53] RA Lewis and Alan R Johnston. "A scanning laser rangefinder for a robotic vehicle". In: *IJCAI* 2 (1977), pp. 762–768.

[54] M. Li and A. I. Mourikis. "Vision-aided Inertial Navigation with Rolling-Shutter Cameras". In: *International Journal of Robotics Research* 33.11 (Sept. 2014), pp. 1490–1507.

[55] Rui Li, Zhenyu Liu, and Jianrong Tan. "A survey on 3D hand pose estimation: Cameras, methods, and datasets". In: *Pattern Recognition* 93 (2019), pp. 251–272.

[56] Beyang Liu, Stephen Gould, and Daphne Koller. "Single image depth estimation from predicted semantic labels". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1253–1260.

[57] Chen Liu et al. "Planenet: Piece-wise planar reconstruction from a single rgb image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2579–2588.

[58] Chen Liu et al. "Planercnn: 3d plane detection and reconstruction from a single image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4450–4459.

[59] Juntao Liu, Wenyu Liu, and Caihua Wu. "Objects similarity measurement based on skeleton tree descriptor matching". In: *2007 10th IEEE International Conference on Computer-Aided Design and Computer Graphics*. IEEE. 2007, pp. 96–101.

[60] Zhi-Yong Liu and Hong Qiao. "GNCCP—Graduated nonconvexityand concavity procedure". In: *IEEE transactions on pattern analysis and machine intelligence* 36.6 (2013), pp. 1258–1267.

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

[62] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[63] Jason Luck, Charles Little, and William Hoff. "Registration of range data using a hybrid simulated annealing and iterative closest point algorithm". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 4. IEEE. 2000, pp. 3739–3744.

[64] Hans P. Moravec. "Sensor fusion in certainty grids for mobile robots". In: *Sensor devices and systems for robotics*. Springer, 1989, pp. 253–276.

[65] A. I. Mourikis and S. I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. Apr. 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.

[66] Raul Mur-Artal and Juan D Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[67] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[68] Matthias Nießner et al. "Real-time 3D reconstruction at scale using voxel hashing". In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), pp. 1–11.

[69] Irina Nurutdinova and Andrew Fitzgibbon. "Towards Pointless Structure from Motion: 3D Reconstruction and Camera Parameters from General 3D Curves". In: IEEE, Dec. 2015, pp. 2363–2371. DOI: 10.1109/ICCV.2015.272.

[70] Edwin Olson, John Leonard, and Seth Teller. "Fast iterative alignment of pose graphs with poor initial estimates". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE. 2006, pp. 2262–2269.

[71] Deepti Pachauri, Risi Kondor, and Vikas Singh. "Solving the multi-way matching problem by permutation synchronization". In: *Advances in neural information processing systems*. 2013, pp. 1860–1868.

[72] Kaustubh Pathak et al. "Fast 3D Mapping by Matching Planes Extracted from Range Sensor Point-Clouds". In: Nov. 2009, pp. 1150–1155. DOI: 10.1109/IROS.2009.5354061.

[73] Taihú Pire et al. "S-PTAM: Stereo parallel tracking and mapping". In: *Robotics and Autonomous Systems* 93 (2017), pp. 27–42.

[74] Srikumar Ramalingam et al. "Manhattan junction catalogue for spatial reasoning of indoor scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3065–3072.

[75] Anand Rangarajan and Rama Chellappa. "Generalized graduated nonconvexity algorithm for maximum a posteriori image estimation". In: *[1990] Proceedings. 10th International Conference on Pattern Recognition*. Vol. 2. IEEE. 1990, pp. 127–133.

[76] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.

[77] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. "3-d depth reconstruction from a single still image". In: *International journal of computer vision* 76.1 (2008), pp. 53–69.

[78] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. "Learning depth from single monocular images". In: *Advances in neural information processing systems*. 2006, pp. 1161–1168.

[79] S Serratosa. "Speeding up fast bipartite graph matching through a new cost matrix. Int". In: *Journal of Pattern Recognition* 29.2 (2015).

[80] Daniel Sharvit et al. "Symmetry-based indexing of image databases". In: *Proceedings. IEEE Workshop on Content-Based Access of Image and Video Libraries (Cat. No. 98EX173)*. IEEE. 1998, pp. 56–62.

[81] Richard Sinkhorn. "A relationship between arbitrary positive matrices and doubly stochastic matrices". In: *The annals of mathematical statistics* 35.2 (1964), pp. 876–879.

[82] J. Sturm et al. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. Oct. 2012.

[83] Ke Sun et al. "Robust stereo visual inertial odometry for fast autonomous flight". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 965–972.

[84] Q. Sun et al. "RGB-D SLAM in Indoor Environments With STING-Based Plane Feature Extraction". In: *IEEE/ASME Transactions on Mechatronics* 23.3 (June 2018), pp. 1071–1082. ISSN: 1083-4435. DOI: 10.1109/TMECH.2017.2773576.

[85] Paul Swoboda et al. "A convex relaxation for multi-graph matching". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11156–11165.

[86] Petri Tanskanen et al. "Semi-direct EKF-based monocular visual-inertial odometry". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6073–6078. DOI: 10.1109/IROS.2015.7354242.

[87] Sarah Tariq and Frank Dellaert. "A multi-camera 6-dof pose tracker". In: *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2004, pp. 296–297.

[88] Yuan Tian et al. "Occlusion and Collision Aware Smartphone AR Using Time-of-Flight Camera". In: *International Symposium on Visual Computing*. Springer. 2019, pp. 141–153.

[89] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. "Feature correspondence via graph matching: Models and global optimization". In: *European conference on computer vision*. Springer. 2008, pp. 596–609.

[90] Tommi Tykkälä, Cédric Audras, and Andrew I Comport. "Direct iterative closest point for real-time visual odometry". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE. 2011, pp. 2050–2056.

[91] *Vicon Motion Systems Ltd UK registered no. 1801446*. 2020. URL: https://www.vicon.com/.

[92] Rafael Grompone Von Gioi et al. "LSD: A fast line segment detector with a false detection control". In: *IEEE transactions on pattern analysis and machine intelligence* 32.4 (2008), pp. 722–732.

[93] Friedrich M Wahl. "A coded light approach for depth map acquisition". In: *Mustererkennung 1986*. Springer, 1986, pp. 12–17.

[94] J. Weingarten and R. Siegwart. "EKF-based 3D SLAM for structured environment reconstruction". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Aug. 2005, pp. 3834–3839. DOI: 10.1109/IROS.2005.1545285.

[95] Thomas Whelan et al. "ElasticFusion: Dense SLAM without a pose graph". In: Robotics: Science and Systems. 2015.

[96] Thomas Whelan et al. "Kintinuous: Spatially Extended KinectFusion". In: *Proceedings of RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*. July 2012.

[97] Junchi Yan et al. "Consistency-driven alternating optimization for multigraph matching: A unified approach". In: *IEEE Transactions on Image Processing* 24.3 (2015), pp. 994–1009.

[98] Junchi Yan et al. "Multi-graph matching via affinity optimization with graduated consistency regularization". In: *IEEE transactions on pattern analysis and machine intelligence* 38.6 (2015), pp. 1228–1242.

[99] Shang-Ta Yang et al. "Dula-net: A dual-projection network for estimating room layouts from a single rgb panorama". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3363–3372.

[100] Shichao Yang, Daniel Maturana, and Sebastian Scherer. "Real-time 3D scene layout from a single image using convolutional neural networks". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 2183–2189.

[101] Yulin Yang et al. "Tightly-Coupled Aided Inertial Navigation with Point and Plane Features". In: May 2019. DOI: 10.1109/ICRA.2019.8794078.

[102] Georges Younes et al. "Keyframe-based monocular SLAM: design, survey, and future directions". In: *Robotics and Autonomous Systems* 98 (2017), pp. 67–88.

[103] Hongsheng Yu and Anastasios Mourikis. "Edge-based visual-inertial odometry". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2017, pp. 6670–6677. DOI: 10.1109/IROS.2017.8206582.

[104] Hongsheng Yu and Anastasios Mourikis. "Vision-aided inertial navigation with line features and a rolling-shutter camera". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2015, pp. 892–899. DOI: 10.1109/IROS.2015.7353477.

[105] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time." In: *Robotics: Science and Systems*. Vol. 2. 9. 2014.

[106] Lilian Zhang and Reinhard Koch. "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency". In: *Journal of Visual Communication and Image Representation* 24.7 (2013), pp. 794–805.

[107] Xing Zheng et al. "Photometric patch-based visual-inertial odometry". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, June 2017, pp. 3264–3271. DOI: 10.1109/ICRA.2017.7989372.

[108] Bill Zhou et al. "Real-Time Hand Model Estimation from Depth Images for Wearable Augmented Reality Glasses". In: *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE. 2019, pp. 269–273.

[109] Feng Zhou and Fernando De la Torre. "Deformable graph matching". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2013, pp. 2922–2929.

[110] Feng Zhou and Fernando De la Torre. "Factorized graph matching". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 127–134.

[111] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[112] Xiaowei Zhou, Menglong Zhu, and Kostas Daniilidis. "Multi-image matching via fast alternating minimization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4032–4040.

[113] Yichao Zhou, Haozhi Qi, and Yi Ma. "End-to-end wireframe parsing". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 962–971.

[114] Yichao Zhou et al. "Learning to reconstruct 3D Manhattan wireframes from a single image". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7698–7707.

[115] Chuhang Zou et al. "Layoutnet: Reconstructing the 3d room layout from a single rgb image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2051–2059.