Camera Marker Networks for Pose Estimation and Scene Understanding in Construction Automation and Robotics

by

Chen Feng

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Civil Engineering)
in the University of Michigan
2015

Doctoral Committee:

Associate Professor Vineet R. Kamat, Chair
Associate Professor SangHyun Lee
Assistant Professor Carol C. Menassa
Professor Atul Prakash

Dedication

To My Parents,
Who bestowed me life, raised me with love, educated me with wisdom,
And support me pursuing my dream from thousands of miles away.

To My Darling,
Who doubles my joys, divides my sorrows, calms me down, and cheers me up,
Our hearts are tied together no matter how far apart.

Acknowledgments

When I was seeking Ph.D. research opportunities five years ago, I was told that a "right" Ph.D. advisor is one of the most important factors of an enjoyable and fruitful Ph.D. life. I feel extremely fortunate and grateful to have Professor Vineet Kamat as my "right" advisor in not only academia but also career development and work-life balance. He is very gifted, far-sighted, patient, flexible and comprehensive in terms of his academic guidance. After four years I still remember his metaphor that doing research in most cases is not like going straight to a destination but more like having a spiral around the direction and walking up around it towards destination. He also pays a great amount of attention in sharpening my soft skills for public presentations, academic collaborations, technology transfer, and so on.

I am also deeply appreciative to my committee members Professor Atul Prakash, Professor SangHyun Lee, and Professor Carol Menassa. Dr. Prakash's suggestions in my preliminary exam helped me rethink and improve the selection of my dissertation contents. Dr. Lee provided many constructive discussions, inspirations, and encouragements for my research. Dr. Menassa's advices on storyboarding of research contents stimulated further reflection and re-organization of my research logic. Their contributions are indispensable to this dissertation.

During my doctoral research, I am honored to collaborate with and be taught by some excellent faculty members and researchers. The joyful and productive collaboration with Professor Wes McGee was my first hands-on experience with industrial robotics, and our discussions about architecture, construction, and robotics are full of imagination and ambition. Dr. Yuichi Taguchi

and Dr. Srikumar Ramalingam were hosts for my intern at Mitsubishi Electric Research Laboratories. Since then, we established good friendship and maintained research collaborations, resulting in several top robotics publications. These two collaborations led to chapter 2 and 5 of this dissertation. Professor Edwin Olson is an extremely talented and energetic researcher and educator who taught me mobile robotics and directly inspired my work on fast plane extraction. His AprilTag algorithm also played important roles in my research. Professor Silvio Savarese taught me computer vision and offered valuable suggestions to improve my work on fast plane registration. Dr. John Everett taught me many core construction courses, and his doctoral dissertation is a great introduction to construction automation. The knowledge and thinking methods I acquired from them laid solid foundations for this dissertation.

I am also sincerely grateful to all my friends and colleagues Suyang Dong, Manu Akula, Sanat Talmaki, Yong Xiao, Kurt Lundeen, Aaron Willette, Yingze Bao, Nicholas Fredricks, Zhiyuan Zuo, Ehsan Rezazedeh Azar, Kyle Anderson, Seungjun Ahn, Srinath Sridhar, and many others. Their assistance is essential for this dissertation. Even more precious for me are their friendships.

While writing this dissertation at my Northwood apartment in Ann Arbor, reviewing what I have learnt and achieved, and getting ready for a new expedition, I cannot help myself recollecting those memories starting from five years ago around this time, when I was writing my undergraduate thesis at a library in Beijing, and preparing for the upcoming journey to the University of Michigan. Without the firm support from my parents, Meijun Feng and Zhurong Deng, this journey would not even take shape and I can never be grateful enough for them. Most luckily, my darling, Shasha Li, joined me in this journey after two years. We joked about how she could delay this dissertation, but the truth is, I would not have a stable mood to conduct my research and finish this dissertation without her companionship, encouragements, and love.

Table of Contents

List of Tables

List of Figures

Abstract

The construction industry faces challenges that include high workplace injuries and fatalities, stagnant productivity, and skill shortage. Automation and Robotics in Construction (ARC) has been proposed in the literature as a potential solution that makes machinery easier to collaborate with, facilitates better decision-making, or enables autonomous behavior. However, there are two primary technical challenges in ARC: 1) unstructured and featureless environments; and 2) differences between the as-designed and the as-built. It is therefore impossible to directly replicate conventional automation methods adopted in industries such as manufacturing on construction sites. In particular, two fundamental problems, pose estimation and scene understanding, must be addressed to realize the full potential of ARC.

This dissertation proposes a pose estimation and scene understanding framework that addresses the identified research gaps by exploiting cameras, markers, and planar structures to mitigate the identified technical challenges. A fast plane extraction algorithm is developed for efficient modeling and understanding of built environments. A marker registration algorithm is designed for robust, accurate, cost-efficient, and rapidly reconfigurable pose estimation in unstructured and featureless environments. Camera marker networks are then established for unified and systematic design, estimation, and uncertainty analysis in larger scale applications.

The proposed algorithms' efficiency has been validated through comprehensive experiments. Specifically, the speed, accuracy and robustness of the fast plane extraction and the marker registration have been demonstrated to be superior to existing state-of-the-art algorithms. These algorithms have also been implemented in two groups of ARC applications to demonstrate the proposed framework's effectiveness, wherein the applications themselves have significant social and economic value. The first group is related to in-situ robotic machinery, including an autonomous manipulator for assembling digital architecture designs on construction sites to help improve productivity and quality; and an intelligent guidance and monitoring system for articulated machinery such as excavators to help improve safety. The second group emphasizes human-machine interaction to make ARC more effective, including a mobile Building Information Modeling and way-finding platform with discrete location recognition to increase indoor facility management efficiency; and a 3D scanning and modeling solution for rapid and cost-efficient dimension checking and concise as-built modeling.

Chapter 1
Introduction

"Let's start with the three fundamental Rules of Robotics."—Isaac Asimov

The objective of this research was to develop, implement and validate novel computer vision based technologies to provide rapidly reconfigurable, infrastructure-independent, robust, reliable and accurate 6 Degree-Of-Freedom (DOF) pose (position and orientation) estimation, 3 dimensional (3D) scene reconstruction and understanding solutions, for various perception and navigation applications of Automation and Robotics in Construction (ARC). These technologies have the potential to help the construction industry improve safety, increase productivity and accelerate the transition of skill-intensive construction jobs to knowledge-intensive ones.

This research is timely and critical. The automatic perception and navigation ability has been highlighted as a basic capability that will impact the automation of many industries including construction by the National Robotics Initiative (Christensen et al. 2009). With such ability, construction machines could be easier to control and collaborate with, require less human supervision to avoid collisions, and even autonomously operate on complex worksites. Workers and managers can also benefit from wearable and mobile devices with such ability to facilitate construction, operation and management tasks. These could alleviate issues like relatively poor safety, stagnant productivity, and skilled labor shortage in both construction and similarly

affected industries such as manufacturing, mining, and shipbuilding, where robotics is poised to revolutionize next-generation processes if key technical challenges related to localization, navigation, manipulation, obstacle avoidance, and collaboration can be successfully resolved.

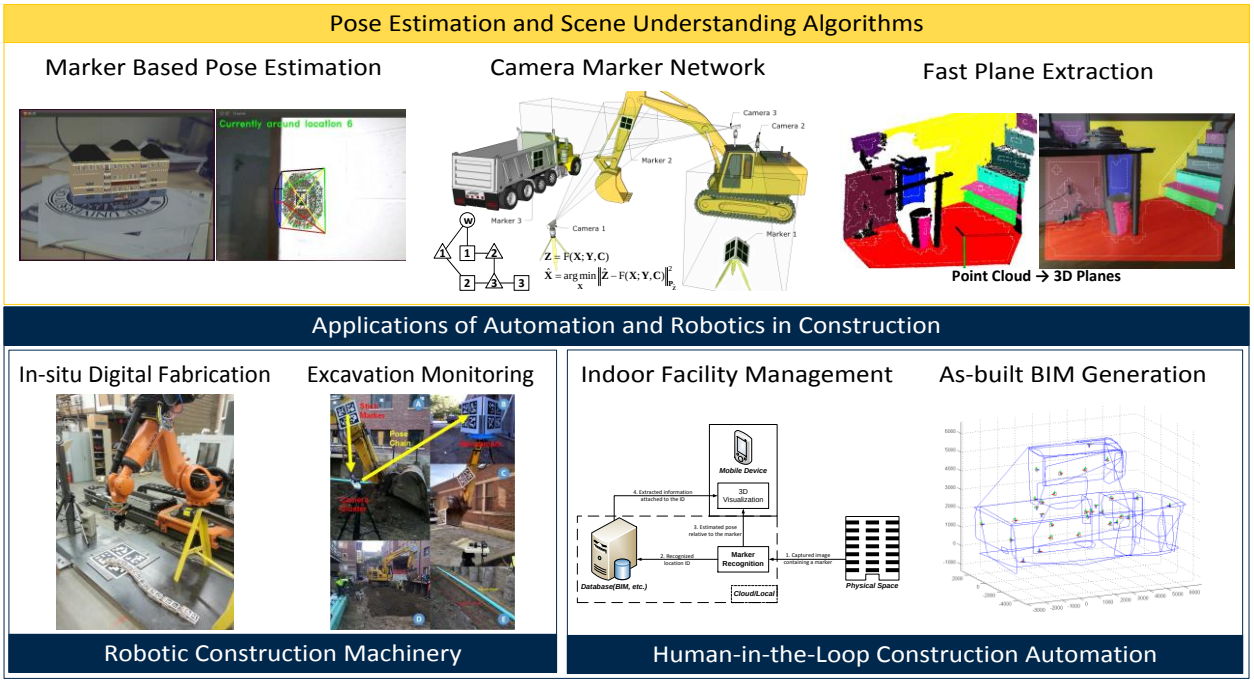


Figure 1-1: Overview of the research.

As shown in Figure 1-1, the research objective was achieved by first investigating general pose estimation and scene understanding algorithms in computer vision and robotics, either by exploring and advancing existing ones or developing new ones. These algorithms were then applied in the ARC domain to explore their potential and validate their effectiveness.

1.1 Importance of the Research

The inability to automatically perceive surrounding environments and estimate poses of either large scale mobile manipulators' key components (e.g., end-effectors of manipulator arms or undercarriage tracks of mobile cranes), or mobile devices used by civil engineering inspectors,

has a significant negative impact on the safety and productivity in industries such as manufacturing, construction, and shipbuilding. This incapability also contributes to the shortage of skilled labor in these industries.

Firstly, high rates of workplace injuries and fatalities in unstructured manufacturing environments (e.g., construction sites, shipyards) have been affecting many industries. According to the 2013 Census of Fatal Occupational Injuries (CFOI) report (Bureau of Labor Statistics 2013), the construction industry had the largest number of fatal occupational injuries, and in terms of rate ranked the fourth highest among all industries. Among all the causes for the 796 fatal injuries in the U.S. construction industry in 2013, the cause of being struck by an object or equipment comprised 10 percent. This percentage is even higher in other industries such as agriculture (19%), forestry (63%), and mining (23%). For example, besides directly causing fatal injuries on worksites, mobile manipulators such as excavators can also inadvertently strike buried utilities, thus disrupting life and commerce, and pose physical danger to workers, bystanders, and building occupants. Such underground strikes happen with an average frequency of about once per minute in the U.S., reported by the Common Ground Alliance, the nation's leading organization focused on excavation safety. More specifically, excavation damage is the third biggest cause of breakdowns in U.S. pipeline systems, accounting for about 17% of all incidents, leading to nearly 25 million dollars annual utility interruptions (US DOT PHMSA 2015). Similar statistics abound in other related industries.

When mobile manipulators are continuously aware of their poses in unstructured environments, and are monitoring their surroundings such as recognizing nearby human workers' poses and actions, such machines can make decisions to avoid striking human workers, for example by sending alerts to their operators or even temporarily taking over the controls to prevent accidents.

Thus, machine self-awareness can help decrease the possibilities of the abovementioned injuries and fatalities and improve safety on worksites. Similarly, with continuous tracking of the pose of an end-effector (e.g., a bucket of an excavator), an intelligent excavator can perform collision detection with an existing map of underground utilities and issue its operator a warning if the end-effector's distance to any buried utilities exceeds predefined thresholds.

In addition to the safety concerns, there are also increasing concerns of relatively stagnant productivity rates and skilled labor shortage in manufacturing and construction industries. From a recent survey, 83% U.S. construction firms are reported to be in shortage of skilled workers (Associated General Contractors of America 2014). Similarly, the construction sector in the United Kingdom is reported to be in urgent need of 20% more skilled workers and thus 50% more training provision by 2017, to deliver projects in planning (LCCI/KPMG 2014). For example, earthwork is a typical affected activity. Currently precise excavation grade control is provided by employing grade-checkers to accompany excavators during appropriate operations. Grade-checkers specialize in surveying and frequently monitor the evolving grade profile. The evolving grade profile is compared to the target profile and this information is communicated by the grade-checker to the excavator operator. The operator reconciles this information and adjusts the digging strokes accordingly. This process is repeated until the target profiles are achieved. Employing grade-checkers is not only dangerous but also results in a significant loss in excavation productivity due to frequent interruptions required for surveying the evolving profile (Feng et al. 2015).

When a mobile manipulator can continuously track its end-effector's pose and capture its 3D surroundings on worksites, such information can be combined together with the digital design of a task, either to assist human operators to complete the task faster and more efficiently, or to

eventually finish the task autonomously. For example, an intelligent excavator being able to track the pose of its bucket can guide its operator to dig trenches or backfill according to designed profiles more easily and accurately with automatic grade-checks. This can eventually lead to fully autonomous in-situ machines, such as deployment of robotic arms or unmanned aerial vehicles on construction sites and shipyards for autonomous assembly and fabrication (Helm et al. 2012; Willmann et al. 2012; Feng et al. 2014). When such machines become more intelligent, due to the transition from skill-based to knowledge-based control, it can be expected to save time in training operators, and thus to mitigate skilled labor shortages and also improve productivity.

In summary, the inability to automatically reconstruct and perceive surrounding scenes and estimate poses not only affects the safety and productivity of the corresponding processes, but also increases the demand for skilled labor in those industries. In particular, the inability to estimate pose in unstructured environments and the lack of rapidly reconfigurable solutions are primary obstacles that need to be overcome. There is thus a clear and critical need for new methods to support accurate and reliable real-time scene understanding and 6DOF pose estimation with rapid and flexible configurations for intelligent guidance and control of machines and mobile devices adopted in relevant industries.

1.2 Background of the Research

1.2.1 Perception and Navigation

In the general context of automation and robotics, pose estimation and scene understanding belong to perception and navigation problems. **Perception** is the process of interpreting sensor data in order to acquire information and develop knowledge of the environment; and **navigation**

is the process of determining the current pose of an object of interest and discovering subsequent actions leading the object to its target pose without collision with other objects. These two definitions imply that perception and navigation are very closely related to each other: fulfillment of the goal of navigation (such as localization, path planning and collision avoidance) usually requires support of perception to provide knowledge of the environment; on the other hand, better perception results can be achieved with improved navigation since sensor data are registered into an improved spatially consistent framework.

Many different types of sensors can be used for perception and navigation, ranging from Global Position System (GPS) receivers, Inertial Measurement Units (IMU), sonars, and optical encoders to generalized vision sensors including cameras, depth cameras (such as stereo vision cameras, Microsoft Kinect, etc.) and laser range finders/lidars. This research will mainly focus on the generalized vision sensors because of both the richness of information that they can capture and the various insufficiencies of other sensors.

Perception includes many different types of problems. To find out a certain object from vision sensor data such as images or point clouds, object detection and recognition are needed. To pinpoint the position or identify the region of that object in those data, object segmentation is necessary. If the vision sensor data are continuously updated, then object tracking is required. 3D reconstruction can provide a 3D geometric description of the environment either from a sequence of images or from registration of a collection of point clouds into a unified coordinate frame.

Many different research communities have spent enormous efforts on perception. Photogrammetry is probably one of the earliest, with focus on theories of 3D reconstruction from aerial imagery and applications in cartography. Computer vision sprouts from digital image

processing and builds on different perception research domains mentioned above. Robotics researchers study visual perception to increase the automation level for machinery and robots. Other industries such as manufacturing, automobile, augmented reality (AR) have all taken advantage of progress in perception.

Navigation in robotics often includes multiple goals: pose estimation, localization, path planning and collision avoidance. Path planning often needs either 2D or 3D maps of the environment as algorithm input. Collision avoidance requires detection of obstacles in the environment. Yet before these two goals, pose estimation or localization has the first priority in order to achieve successful navigation. The ability to recover a user's pose (i.e., position and orientation within a certain coordinate frame) is critical in many engineering domains such as AR, robotics, context-aware computing, and computer vision. In AR, for example, this task is termed as the "registration" problem (Azuma 1997). In robotics, this task is closely related to "Simultaneous Localization and Mapping" (SLAM) (Klein and Murray 2007; Thrun 2008). In computer vision, "Structure from Motion" (SfM) algorithms are designed to solve this problem with little or no prior knowledge about the environment (Sturm and Triggs 1996; Snavely et al. 2006; Bao and Savarese 2011). Context-aware engineering applications also face a similar problem where the positioning part is more relevant (Akula et al. 2011). In cinematography, the problem is called "move matching".

1.2.2 Automation and Robotics in Construction

ARC is comprised of two major categories: hard and soft ARC (Balaguer 2004). Just as "every construction chore has physical components and information components" (Everett and Slocum 1994), hard ARC focuses mainly on construction tasks which contain a large portion of physical processing, such as robotics for brick laying, interior finishing, road paving, etc.; while soft ARC

concentrates mostly on construction tasks which typically require higher level information processing, such as document management, progress monitoring, safety monitoring, maintenance and inspection, and as-built Building Information Modeling (BIM).

1.2.2.1 Challenges

Even though hard ARC had been actively studied in the 1990s, ARC research has been shifting towards the soft ARC side since the last decade. From a previous research trend study (Son H. et al. 2010) about papers published in the proceedings of the International Symposium on Automation and Robotics in Construction (ISARC), a huge net decrease of hard ARC related papers from about 70% to 35% was observed. This trend highlights the importance of incorporating more soft ARC techniques into the hard ARC side, which means more automatic information processing abilities should be developed for construction machinery or devices to increase their level of automation and thus to make them easier to use (Balaguer 2004). This is because that on top of perception and navigation challenges inherited from general automation and robotics such as speed, accuracy and robustness of algorithms, part of the reasons for this decrease could be the following unique challenges in civil engineering and similar industries.

Unstructured and featureless environment: unlike traditional manufacturing, where robotic solutions benefit from the structured layout of the environment (e.g., factory assembly line), construction robots face unique challenges that arise from the unstructured, dynamic, and sometimes featureless environment of the work site, as shown in Figure 1-2, as well as the uncertainty and evolving sequence of occurring on-site events. This challenges any intended construction robots to not only replicate basic human motion, but also be capable of accurately and reliably sensing and adapting to environmental changes, and making decisions based on the

evolving state of the environment. Examples of ARC applications under such environments include the on-site robotic assembly in Chapter 5 and the excavation monitoring in Chapter 6.

Figure 1-2: Unstructured environments with repeated features or featureless[1] characteristics.

Difference between the as-designed and the as-built: many buildings or civil infrastructures have different extent of discrepancies between their designs and as-built results, since many issues are not anticipated or simply unpredictable during the design phase. This poses another layer of challenges when trying to incorporating the design as prior knowledge for perception and navigation algorithms for ARC. Examples of ARC applications related to such challenges include facility management in Chapter 7 and the as-built modeling in Chapter 8.

1.2.2.2 Principle and Methodology

Due to the abovementioned challenges, many perception and navigation algorithms designed for traditional manufacturing robots cannot be directly applied in ARC out of the box. When fully autonomous construction robots seem to lack the required algorithmic foundations and practical feasibilities, semi-automation in construction enabled by Human Machine Interaction (HMI) is

[1] The two photos come from Buildipedia.com.

identified to be “preferential in the mobile and non-standardized construction environment” (Han C. 2011). Previous work about either interior finishing robot (Kahane and Rosenfeld 2004b; Navon 2000), where human operators need to manually transfer the robot between workstations, or infrastructure inspection and maintenance robot (Kim and Haas 2000), where manual editing and correction of automatic crack sealing error is needed, had followed this principle. This research was also guided by the same principle, as shown in applications in later chapters.

Besides the HMI principle, to efficiently address those challenges, many ARC methodologies have been proposed as guidelines to identify construction tasks and develop robotics and automation solutions for them. Everett (1991) described a hierarchical taxonomy of construction field operations, in which two important levels of construction operation, activity and basic task, are proposed. While many hard ARC research had focused on activity level automation, i.e. whose output “results in a recognizable, completed unit of work with spatial limits and/or dimensions” (Everett and Slocum 1994), Everett (1991) proposed to conduct ARC research on the level of the basic task—fundamental elements that build up construction activities, since technology advancement on this level could be applied to many different construction activities, as opposed to automation on activity level. This research followed the same idea and advanced it by changing the perspective of basic task level automation from construction worker to autonomous/semi-autonomous construction machinery.

In Everett's hierarchical taxonomy of construction field operations (Everett 1991), the basic task level—including connect, cover, cut, dig, finish, inspect, measure, place, plan, position, spray and spread—is the one recommended for most easy introduction of construction automation. Since basic task is the fundamental element of construction field work, successful automation on one basic task could more easily benefit many different construction activities.

However, when ARC researchers actually try to automate these basic tasks, one issue they will encounter is likely to be the sub-problem overlap. For example, to automate the “connect” basic task, the first question for the designer to ask might be “how to identify the objects to be connected”. Thus object detection and recognition is a sub-problem for this basic task. On the other hand, to automate the “cut” basic task, the same sub-problem of object detection and recognition must be addressed since the robot needs to know what object needs to be cut. Similarly, the question “where and in what pose should the object be positioned” must be answered for the robot to automate both “position” and “place” basic tasks.

It is thus interesting to note that the basic tasks were summarized and abstracted from construction activities from the perspective of a human worker or manager. It is indeed natural, obvious and easy to assign commands made up from these basic tasks to human workers, whereas commands for construction robots require specification of additional detailed information in forms that machines understand.

Therefore, inspired by the modularization thinking in Everett's methodology and the identification of overlapping sub-problems, to efficiently automate basic tasks, their common sub-problems should be investigated and automated first. By further examining these sub-problems, one can realize that most of them are related to the information processing. Hence, the construction basic task automation methodology in this research is as follows:

1. For each basic task, identify input and output information;
2. Find each commonly needed type of information and define an atomic function which outputs that information;
3. Prioritize all atomic functions and selectively automate them;

4. Automate or semi-automate basic tasks which require information output by automated atomic functions.

This methodology is in line with the previous trend analysis stating that more automatic information processing abilities (the atomic functions) must be possessed by construction machinery and devices. Guided by this methodology, firstly the commonly needed information is analyzed. From Table 1–1 one can see that information such as position and orientation, object identity and geometric description of the environment are commonly needed. Moreover almost all autonomous mobile robots need this information to navigate themselves to their destination. Thus the corresponding atomic functions, i.e., pose estimation and scene understanding, which belong to perception and navigation for ARC, are chosen to be investigated in this research.

Table 1–1: Commonly needed information for each construction basic task.

Basic Task	Object Identity	Position and/or Orientation	Area/Region/Shape/Boundary
Connect	√	√	
Cover	√		√ (Region to be covered)
Cut	√	√ (pose of cutting tool)	
Dig		√	√ (Region to be dug)
Finish			√ (Region to be finished)
Inspect	√	√	
Measure	√	√	
Place	√	√	
Plan		√	
Position	√	√	
Spray		√ (pose of spraying tool)	√ (Region to be sprayed)
Spread		√ (pose of spreading tool)	√ (Region to be spread)

1.2.3 Previous Work

In the ARC community, perception and navigation have been studied since the 1990s (Everett 1991; Beliveau et al. 1996; Forsberg et al. 1997; Shohet and Rosenfeld 1997). Recently modern computer vision techniques are being introduced into ARC community, including 3D

reconstruction from unordered image sets for construction visualization and progress monitoring (Golparvar-Fard et al. 2009), object detection and tracking for automatic productivity estimation (Rezazadeh Azar and McCabe 2012), 3D human skeleton reconstruction for construction occupational disease analysis (Han and Lee 2013), and planar structure extraction from surveyed point clouds of buildings for as-built BIM (Zhang et al. 2012). Building on recent advancements in robotic navigation and control, researchers from robotics community also have been making efforts towards autonomous robots that can perform certain simplified construction tasks, such as structure or brick assembly by quadrotors (Lindsey et al. 2012; Willmann et al. 2012).

Many researchers have realized that to increase the level of autonomy for construction robots, the mapping and navigation abilities of the robot are essential (Beliveau et al. 1996; Forsberg et al. 1997; Shohet and Rosenfeld 1997). However, the accuracy of SLAM algorithm was found to be insufficient at that time for many construction tasks which require direct manipulation of construction materials or tools (Shohet and Rosenfeld 1997). Some researchers even suggested removing the autonomous navigation functionality and transferring robots between workstations manually, then performing either a coarse-to-fine calibration (Kahane and Rosenfeld 2004a) or carrying out an additional vision-based real-time quality assurance step (Navon 2000).

As core functions of either mapping or navigation, two types of pose estimation techniques have been extensively studied, i.e. traditional non-visual-sensor-based methods, and newly emerging visual-sensor-based methods, briefly introduced as follows.

1.2.3.1 Non-visual-sensor-based Methods

Among the first type, GPS is mainly used in outdoor open areas. GPS signals are easily blocked by obstacles (e.g., buildings) that result in decreased accuracy or even failure of localization,

known as the “urban canyon” effect (Cui and Ge 2003; Groves 2011). Wireless Local Area Network (WLAN) based methods also require large number of footprints for calibration (Aziz et al. 2005). Ultra-Wide Band (UWB) based methods generally have high cost (Teizer et al. 2008; Khoury and Kamat 2009). Radio Frequency Identification (RFID) based methods usually depend on large infrastructure (i.e., sufficient RFID tags must be available) and also requires special tag readers (Sanpechuda and Kovavisaruch 2008; Andoh et al. 2012). IMU has tracking drift issues that require error correction (Akula et al. 2011). Most of these methods (except for IMU) are dependent on certain installed tracking infrastructure. Besides, none of them can easily provide direct orientation information (angular sensors in IMU such as gyroscope, electrical compass or accelerometer have problems such as tracking drift or sensitivity to magnetic environment changes), which makes them not optimal for the aforementioned industrial application scenarios.

1.2.3.2 Visual-sensor-based Methods

On the contrary, the second type of methods directly outputs orientation along with position information, by analyzing images captured from visual-sensors (e.g., cameras, lidars). Based on their different assumptions/requirements on the surrounding environment, these algorithms can be classified into two groups: known vs. unknown environment (Lepetit and Fua 2005). The only unknown in the first group is the sensor's pose. While in the second group, both the environment and the sensor's pose have to be estimated, i.e. the SLAM problem (Thrun 2008). Generally, SLAM-based methods are inherently infrastructure-independent due to minimal assumptions. Traditional 2D SLAM methods rely on lidar measurements, while the range limits, cost and even the weight of lidars are disadvantages for their large-scale outdoor applications. Although emerging visual SLAM algorithms (Davison et al. 2007; Klein and Murray 2007; Engel et al.

2014) try to avoid these drawbacks by using ordinary cameras, they have limitations including small range or inadequate accuracy and robustness.

On the other hand, the first group of methods assumes the environment is fully or partially known, thus providing more accurate, reliable and robust pose estimation. These pieces of known appearance and geometry information are so-called markers, which could be pre-designed planar or non-planar objects. Thus, they are also referred to as marker-based pose estimation, which have been extensively studied in many areas such as AR and robotics (Olson 2011), including context-aware computing (Feng and Kamat 2012; Feng and Kamat 2013) and in-situ digital fabrication (Feng et al. 2014).

1.2.4 Limitations of Previous Work

The current state of knowledge has three critical limitations that preclude the application of pose estimation and scene understanding of construction machinery or devices to increase their level of autonomy:

- Lack of Rapidly Reconfigurable and Sensor-Infrastructure-Independent Methods
- Lack of Reliable Visual-Sensor-Based Methods in Complex Environments
- Lack of Systematic Design and Error Analysis for Industrial Applications

1.2.4.1 Lack of Rapidly Reconfigurable and Sensor-Infrastructure-Independent Methods

Traditional non-visual-sensor-based methods have various limitations for large-scale mobile manipulator applications, including inadequate accuracy and robustness, high cost, slow reconfiguration, and infrastructure dependency. Within this type of methods, robotic total stations provide the most accurate (millimeter level) position estimation given a clear line of sight. However, it can only track one target's 3D position at a time, which makes real-time 6DOF

pose estimation intractable, in addition to its relative high cost and payload. Real time kinematic (RTK) GPS provides 3D position estimation at centimeter level accuracy, but it also has relative high cost and payload, and inherits the common GPS issues noted above (Akula et al. 2011). UWB methods provide sub-meter level position accuracy, but are expensive to setup the infrastructure. WLAN methods provide meter level position accuracy, but are slow to configure and calibrate the infrastructure. Due to low positioning accuracy, orientation estimation is not directly available using UWB or WLAN. Although IMU can be integrated for orientations, the aforementioned tracking drift and magnetic environment sensitivity issues make it less robust and appealing. Thus, it is clear that because of these limitations, non-visual-sensor-based methods are non-optimal for the pose estimation in guidance and control of large scale mobile manipulators. To overcome such limitations, visual-sensor-based methods are increasingly studied and have the potential to bridge these gaps.

1.2.4.2 Lack of Reliable Visual-Sensor-Based Methods in Complex Environments

Visual-sensor-based methods differ from other methods by the ability to instantaneously and non-intrusively capture massive amounts of information as images of the environment. Thus, these methods have the potential to interpret sensors' surroundings and estimate sensors' poses without any hardware infrastructure as needed in GPS (satellites), UWB, or WLAN, i.e., they are inherently infrastructure-independent. However, the challenge in these methods is mainly the robust and accurate interpretation of images. Current visual SLAM methods commonly assume that:

1. The working environment has abundant visual features; and
2. This environment is completely or at least mostly static in both appearance and geometric structure.

But these two assumptions do not normally hold in complex industrial environments. For example, on construction sites, workers and machines are constantly and frequently moving, which makes the sites highly dynamic instead of static. In addition, many of such surroundings are feature-less in terms of visual appearance, for instance, shipyards before finishing have almost the same appearance everywhere on the walls and ceilings. Moreover, repeated visual features commonly exist in such environments, which decreases the robustness of the interpretation and hence impede the pose estimation accuracy. Besides, there are also challenges such as the computational burden of real-time image interpretation and the lack of scale estimation in visual SLAM using a monocular camera (i.e., it estimates the camera's position in an undefined distance unit). Thus, it is clear that although the visual-sensor-based methods (especially visual SLAM) have the potential to overcome limitations of non-visual-sensor-based methods, presently they are still in the early research phase and not readily feasible for applications in complex industrial environments where construction machinery are operated.

1.2.4.3 Lack of Systematic Design and Error Analysis for Industrial Applications

It is not sufficient to only estimate the pose of a key component of a mobile manipulator. The accuracy and uncertainty of the estimated pose is critical for the following reasons. Firstly, the uncertainty provides a measure of the confidence level of the estimated pose, which is necessary for many downstream applications (e.g., deciding buffer size for collision avoidance). Secondly, it serves as a tool for evaluating the stability of the pose estimation system under different system configurations, and thus provides further guidance to avoid critical configurations that lead to unstable pose estimation.

Current visual-sensor-based methods normally have neither systematic uncertainty analysis nor practical accuracy evaluation. Usually, visual SLAM methods' position accuracies are from

comparisons with GPS positioning as ground truth, while the orientation accuracies are omitted or evaluated qualitatively. This is insufficient since the orientation accuracy also affects downstream decision-making and GPS might not provide accurate enough ground truth in urban areas, especially in GPS-denied regions. In addition, although a few marker-based methods applied Monte-Carlo simulation and made some empirical observations (Luhmann 2009), neither visual SLAM nor marker-based methods have systematic analysis in terms of the relationship between estimation stability and system configuration to improve pose estimation system design.

1.3 Research Objectives

As previously stated, the overall objective of this research was to develop, implement and validate novel computer vision based technologies to provide rapidly reconfigurable, infrastructure-independent, robust, reliable and accurate 6 DOF pose estimation, 3D scene reconstruction and understanding solutions, for various applications of ARC. The more specific objectives of this research were as follows.

- Develop algorithms of real-time scene understanding in 3D point clouds, to enable more accurate 3D scene reconstruction, and to enable semantic recognition of different geometric elements (e.g., walls, floors, ceilings, stairs, etc.), thus facilitating as-built BIM generation and mobile robot perception.
- Develop algorithms of accurate and robust real-time marker-based pose estimation, to serve as core algorithmic components in camera marker networks.
- Develop algorithms of pose estimation using camera marker networks that has little or no hardware infrastructure dependency and thus can be rapidly applied in large scale applications.

- Design and implement generic software frameworks of the new methods for further industrial applications and prototypes.
- Evaluate accuracy and precision of the new methods in virtual and real-world scenarios.
- Validate effectiveness and investigate potential of the new methods through industrial application prototypes in robotic construction machinery, including autonomous in-situ robotic assembly, using vision-guided mobile manipulators to digitally fabricate curved walls more efficiently; as well as intelligent excavation monitoring using a camera marker network for articulated machine pose estimation to improve excavation safety and productivity.
- Validate effectiveness of the new methods through applications in construction automation with human-in-the-loop, including indoor facility management using mobile devices and camera marker networks to increase inspection efficiency, as well as camera marker networks assisted 3D scene reconstruction and geometric element recognition for cost-efficient and more reliable and accurate as-built BIM generation.

The end results of pursuing these objectives are three general scene understanding and pose estimation algorithms, corresponding software frameworks for both soft and hard ARC applications, and the four specific ARC applications mentioned above.

1.4 Research Methodology

The methodology of this research is first to investigate, adapt existing pose estimation and scene understanding algorithms and develop new ones when necessary, then with the help of the domain knowledge from construction and civil engineering, to apply those fundamental algorithms in appropriate ARC applications. Figure 1-1 above shows the overview of such algorithms-to-applications methodology in this research.

One of the advantages of this methodology is that due to consideration of the prior knowledge from the application domain, i.e., construction or civil engineering, existing general pose estimation or scene understanding algorithms can be modified and adapted as needed to better fit targeted application requirements. Especially when algorithms developed for general computer vision or robotics applications make assumptions that do not hold in construction scenarios, new algorithms are well-motivated for development.

Another advantage of this methodology is that the new algorithms developed in this research are not limited to only construction or civil engineering, but applicable also in other engineering domains. For example, the fast plane extraction can accelerate scene understanding for general robotics problems such as point-plane based SLAM (Taguchi et al. 2013) or autonomous unmanned aerial vehicle (UAV) control. The marker based pose estimation can improve the stability of desktop AR. The camera marker network can also be applied for jobsite machinery productivity analysis. Thus, both ARC community and general computer vision and robotics communities can benefit from these algorithms.

1.5 Dissertation Outline

This dissertation is a compilation of peer-reviewed scientific manuscripts which document this research of the development of novel scene understanding and pose estimation algorithms as well as the designing and implementation of ARC applications adopting those algorithms.

There are mainly three parts in this dissertation. Part I, including chapter 2, 3 and 4, describe the general scene understanding and pose estimation algorithms. Chapter 2 describes a novel scene understanding algorithm that extracts planes from depth images in real-time. Chapter 3 describes a novel marker based pose estimation algorithm that enables fast, accurate and robust 6DOF pose

estimation between a camera and a planar marker. Chapter 4 describes the abstract model of pose estimation using a network of cameras and markers and corresponding mathematical theories for pose estimation and error analysis.

Part II, including chapters 5 and 6, describe two applications of pose estimation in robotic construction machinery. Chapter 5 describes an in-situ digital assembly application using a vision-guided mobile robotic manipulator. Chapter 6 describes an articulated machine pose estimation application using a camera marker network for excavation monitoring and guidance.

Part III, including chapters 7 and 8, describe two construction automation applications with human-in-the-loop. Chapter 7 describes indoor facility management applications using a dynamic camera marker network using markers as spatial indices to link physical locations and associated information. Chapter 8 describes a reliable and accurate as-built BIM generation application using an RGBD camera marker network to both reconstruct 3D point clouds and recognize plane based 3D parametric models.

The dissertation concludes with Chapter 9, which summarizes the significance and contributions of this research, and discusses future work directions. Since each chapter from 2 to 8 is written as a self-contained paper, some information appears in multiple chapters for the sake of completeness. All chapters have been written such that they can be easily understood and successfully replicated by a technically literate audience from diverse domains with basic understandings of 3D computer vision and robotics.

Part I: General Scene Understanding and Pose Estimation Algorithms

"The world is flat."—Thomas Friedman

This part includes three fundamental algorithms centered on planes addressing scene understanding and pose estimation problems. In many cases 3D point clouds of the environment are not enough for robotics applications, since they are generally noisy, redundant, and without explicit semantics of the scene. For compact and semantic 3D modeling, fitting primitives in 3D point clouds has attracted many research interests. In particular, planes are one of the most important primitives, since man-made structures consist of many planes. Thus Chapter 2 describes a fast plane extraction algorithm from depth images.

Planes not only enable compact 3D modeling, but can also facilitate 6DOF pose estimation. This is because the relative pose between a camera plane and a plane with a marker is encoded in a so-called homography matrix, which can be estimated given geometric correspondences between the two planes. Chapter 3 investigated two major groups of methods to establish or maintain correspondences and developed a more accurate and robust pose estimation algorithm.

With more markers or cameras in large scale, an observation network is naturally formed. If depth cameras are used, 3D planes become a new kind of observation. Chapter 4 abstracted such networks in a unified framework, developed general solution and performed systematic error analysis.

Chapter 2
Fast Plane Extraction in Organized Point Clouds

"Divide each difficulty into as many parts as is feasible and necessary to resolve it."

—René Descartes

2.1 Introduction

As low-cost depth cameras and 3D sensors have emerged in the market, they have become a popular choice in various robotics and computer vision applications. 3D point clouds obtained by such sensors are generally noisy and redundant, and do not provide semantics of the scene. For compact and semantic modeling of 3D scenes, primitive fitting to the 3D point clouds has attracted a lot of research interests. In particular, planes are one of the most important primitives, since man-made structures mainly consist of planes.

In this chapter, an efficient plane extraction algorithm applicable to organized point clouds, such as depth maps obtained by Kinect sensors, is presented. This algorithm first constructs a graph by dividing a point cloud into several non-overlapped regions with a uniform size in the image space. The algorithm then performs a bottom-up, agglomerative hierarchical clustering (AHC) on the graph: It repeats (1) finding the region that has the minimum plane fitting mean squared error (MSE) and (2) merging it with one of its neighbors such that the merge results in the minimum plane fitting MSE. It is shown that the clustering process can be done with the complexity log-linear in the number of initial nodes, enabling real-time plane extraction. To refine the boundaries of the clustered regions, the clustering process is followed by pixel-wise

region growing. In experiments, this algorithm is compared with state-of-the-art algorithms. This algorithm achieves real-time performance (runs over 35 Hz) for 640 by 480 pixel depth maps, while providing the accuracy comparable to the state-of-the-art algorithms. Some example results are shown in Figure 2-1. Extracted planes are superimposed with different colors on the RGB image (black means non-planar region). White dash lines show the segmentation boundaries before the region-grow-based refinement. Initial node size of 10 by 10 detects most of the planes in the scene (top-left), whose 3D view is shown (bottom-left). Initial node size of 4 by 4 reveals more segments in a smaller scale such as stairs and table leg (top-right), while that of 20 by 20 focuses on major large planar structures such as floors and walls (bottom-right).

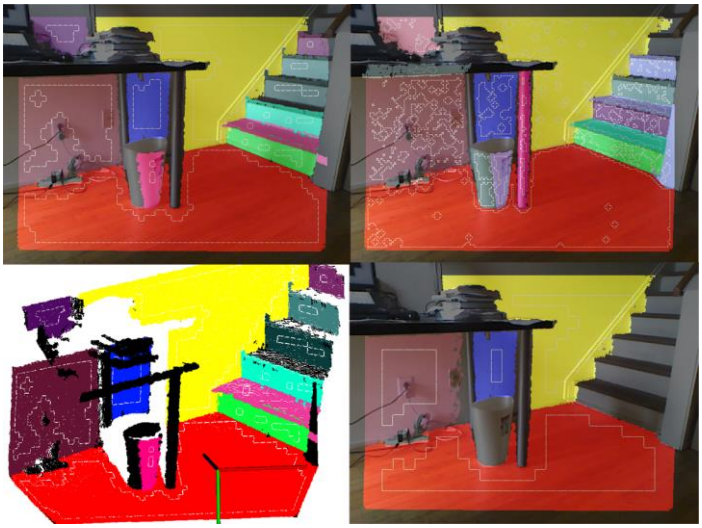


Figure 2-1: Plane extraction results generated with different initial node sizes.

2.1.1 Contributions

There are following contributions for this chapter:

- An efficient plane extraction algorithm based on agglomerative clustering is presented.
- The complexity of the clustering algorithm is analyzed and shown to be log-linear in the number of initial nodes.

- Real-time performance is demonstrated with the accuracy comparable to state-of-the-art algorithms.

The following sections of this chapter will explain the details of this proposed Plane Extraction using Agglomerative Clustering (dubbed as PEAC). Section 2.2 will explain the related work on plane extraction, and related applications. Section 2.3 will give an overview of PEAC including an analogy to line segment extraction and the differences when generalizing to three dimensions. Section 2.4 and 2.5 will explain the two main phases of PEAC. The PEAC's performance is then demonstrated by various experiments in section 2.6. Finally conclusions are drawn in section 2.7.

2.2 Related Work

2.2.1 Plane Extraction

Several different algorithms have been proposed for plane extraction from 3D point clouds. RANSAC-based methods (Schnabel et al. 2007) have been widely used. These methods usually follow the paradigm of iteratively applying RANSAC algorithm on the data while removing inliers corresponding to the currently found plane instance. Since RANSAC requires relatively long computation time for random plane model selection and comparison, several different variants were developed. Oehler et al. (2011) performed Hough transformation and connected component analysis on the point cloud first as pre-segmentation and then applied RANSAC to refine each of the resulting "surfels" (2s per 640 by 480 points). Several algorithms (Taguchi et al. 2013; Hulik et al. 2012; Lee et al. 2012) applied RANSAC on local regions of the point cloud (which decreases the data size considered in each RANSAC run to increase speed) and then grew the region from the locally found plane to the whole point cloud (0.2s (Taguchi et al. 2013) or 0.1s (Hulik et al. 2012) per 640 by 480 points; 0.03s (Lee et al. 2012) per 320 by 240 points).

Region-grow-based methods are another popular choice. Häehnel et al. (2003) and Poppinga et al. (2008) grew points by both point-plane distance threshold and MSE threshold (0.2s per 25,344 points). Holz et al. (2012) grew points by their surface normal deviation (0.5s per 640 by 480 points), which requires per-point normal estimation. A similar but much slower variant is voxel grow (Deschaud and Goulette 2010). Instead of growing points, Geogiev et al. (2011) first extracted line segments from each scan line of the data and then grew the line segments across scan lines (0.05s per 18,100 points in MATLAB).

There are other methods which do not belong to the two groups. Holz et al. (2011) first clustered the point cloud in the normal space and further clustered each group by its distance to the origin (0.14s per 640 by 480 points). To avoid per-point normal estimation, Enjarini et al. (2012) designed the gradient of depth feature for plane segmentation, which could be rapidly computed. Graph-based segmentation using self-adaptive threshold was also used (Strom et al. 2010; Wang et al. 2013) (0.17s per 148,500 points in Strom's paper). Although the PEAC proposed in this chapter also uses a graph to represent data relation, it differs from the previous methods as follows: 1) no RGB information is used; 2) no per-point normal estimation is required; and more importantly, 3) dynamic edge weights are used instead of static ones which fix the merging order as in (Strom et al. 2010).

2.2.2 Applications

Planes have been used in various applications in robotics, computer vision, and 3D modeling. Compact and semantic modeling of scenes provided by planes is useful in indoor and outdoor 3D reconstruction, visualization, and Building Information Modeling (BIM) (Zhang et al. 2012). Extracting a major plane is a common strategy for table-top manipulation (Holz et al. 2011), because it helps segment objects placed on the plane. Planes have been also used for SLAM

(Weingarten and Siegwart 2006; Pathak et al. 2010; Trevor et al. 2012) and place recognition (Fernandez-Moral et al. 2013) systems as landmarks, because planes are more robust to noise and more discriminative than points. However, at least three planes whose normals span $\mathbb{R}^3$ are required to compute the 6 DOF camera pose. To avoid the degeneracy due to the insufficient number of planes, Taguchi et al. (2013) used both points and planes as landmarks in their SLAM system. Salas-Moreno et al.'s SLAM system (2013) that uses objects as landmarks extracted a ground plane and used it as a soft constraint to align the poses of objects with respect to the ground plane. All of the above works can benefit from the fast plane extraction in this chapter.

2.3 Algorithm Overview

Figure 2-2 illustrates how PEAC processes each frame of an organized point cloud. Each frame of an organized point cloud is processed from left to right. (a) shows the graph initialization with each node colored by its normal; black dot and line showing graph node and edge; red 'x', black 'o', and red dot showing node rejected by depth discontinuity, missing data, and too large plane fitting MSE, respectively. (b) and (c) show the two core operations of the AHC. Regions with random colors in (b) and (c) show graph nodes merged at least once. Black lines in (c) show all edges coming out from the node A, in which the thick line shows the edge to the node B that gives the minimum plane fitting MSE when merging the node A with one of its neighbors. Colored regions in (d) show the extracted coarse planes, which are finally refined in (e) if required by the application.

An organized point cloud is defined to be a set of 2D indexed 3D points $\mathcal{F} = \{\mathbf{p}_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})^{\mathsf{T}}\}$, $i = 1, \cdots, \mathsf{M}, j = 1, \cdots, \mathsf{N}$ where the 2D indices (i, j) and $(i \pm 1, j \pm 1)$ reflect the 3D proximity relationship between points $\mathbf{p}_{i,j}$ and $\mathbf{p}_{i\pm1,j\pm1}$ if they lie on the same

surface (this index space is dubbed as image space). Usually it can be obtained from a depth map produced by devices such as a Kinect sensor, time-of-flight camera, structured light scanning system, and even rotating the scanning plane of a laser range finder.

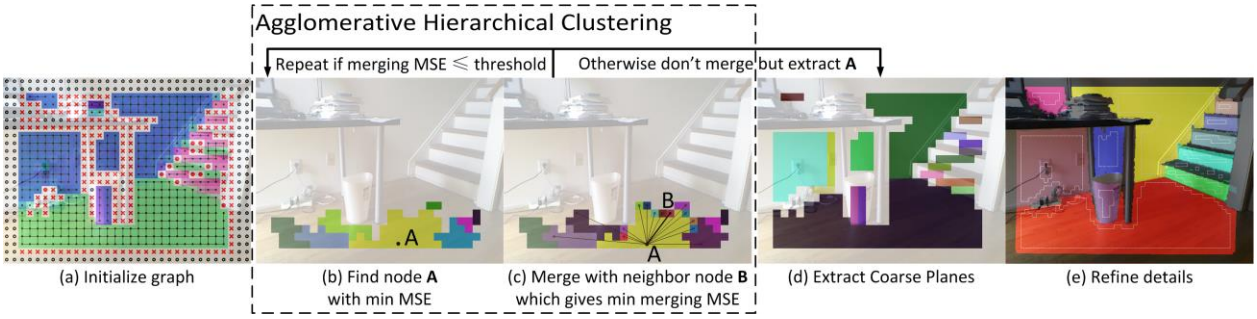


Figure 2-2: The PEAC algorithm overview.

2.3.1 Line Segment Extraction as an Analogy

Before moving into the details of PEAC, a line segment extraction algorithm called *line regression* is briefly discussed, as summarized in (Nguyen et al. 2005) and implemented in April Robotics Toolkit (Olson 2010). It is widely used for extracting line features from 2D point sequences obtained from a laser range finder, and inspired us to generalize its idea to 3D case for fast plane extraction. As illustrated in Figure 2-3 (blue dots show the 2D points; circles labeled with letters show the nodes in a linked list; brackets show the groups of points represented by the nodes; thick line indicates that merging node g with its left neighbor ef gives a smaller line fitting MSE than merging it with its right neighbor h), every W consecutive points ($W = 3$ in this figure) in the sequence are grouped into nodes[2], forming a double linked list. Then AHC is performed on this linked list by repeating (1) finding the node g with the minimum line fitting MSE and (2) merging this node g with either its left or right neighbor that gives the minimum merging MSE. If the minimum merging MSE is larger than a predefined threshold, which can

[2] Note that "node" and "segment" are used interchangeably to represent a set of data points.

usually be decided by the noise characteristics of the sensor, then the merging is canceled and the node g can be extracted as a line segment. When using a binary heap to find the minimum MSE node, log-linear time complexity $O(n \log n)$ can be achieved for this algorithm, where n is the number of points in the sequence. Note that by applying the idea of integral images, as used in (Holzer et al. 2012; Holz et al. 2011), merging two nodes and calculating the resulting line fitting MSE become constant time operations.

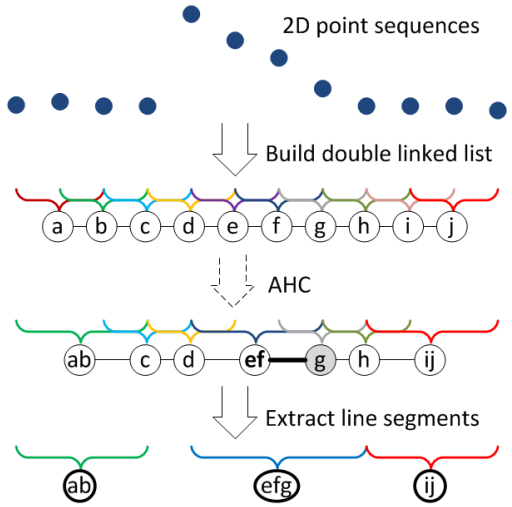


Figure 2-3: Line regression algorithm.

2.3.2 Differences When Generalizing to 3D

Inspired by the use of point's neighborhood information given by the point's order of the sequence, one wish to generalize the 2D line regression to 3D plane extraction in an organized point cloud, where the neighborhood information is stored in the 2D indices. However, this generalization is nontrivial, because of the following two major differences.

2.3.2.1 Non-Overlapping Nodes

As opposed to the line regression, initial nodes (and thus any two nodes during/after merging) should have no identical points, i.e., for any two nodes $\mathcal{B}_s, \mathcal{B}_t \subset \mathcal{F}$, $\mathcal{B}_s \cap \mathcal{B}_t = \varnothing$. This

requirement is due to the fact that after several merging steps, the 3D points belonging to a certain node $\mathcal{B}_s$ will form an irregular shape instead of maintaining its initial rectangular shape in the image space, as shown in Figure 2-2(b). Thus, if allowing different nodes to have identical points, it is difficult to efficiently handle the overlapping points when merging two nodes, even with the help of integral images. While in the line regression, merging two neighboring line segments will still result in a line segment represented by a start and end index in the point sequence, which makes overlapping nodes feasible. It is important to notice that the overlapping nodes enable the line regression algorithm to automatically split line segments at their boundaries; since nodes containing points at different line segments tend to have larger line fitting MSE than others (e.g., nodes c, d, and h in Figure 2-3), their merging attempts will be delayed and finally rejected. The non-overlapping requirement in PEAC results in losing that advantage of automatically detecting boundaries of planes. Section 2.4.1 will describe how to overcome the disadvantage by removing bad nodes in the initialization step. Section 2.5 will also describe a pixel-wise region growing algorithm to refine the boundaries of planes.

2.3.2.2 Number of Merging Attempts

In the line regression, merging a node with its neighbor is a constant time operation with at most two merging attempts, either to its left or right neighbor. In this generalized case, the number of merging attempts is larger, since nodes are initially connected to at most 4 neighbors to form a graph, and after several merging steps, they can be connected to a larger number of neighbors. In section 2.4.2, the average number of merging attempts in PEAC will be experimentally analyzed and shown that it stays small in practice; therefore, the merging step can be done in a constant time, resulting in the complexity of $O(n \log n)$ similar to the line regression.

2.4 Fast Coarse Segmentation

The PEAC algorithm consists of three major steps, as shown in Figure 2-2 and Algorithm 2–1: The algorithm first initializes a graph and then performs AHC for extracting coarse planes, which are finally refined. If the application only requires rough segmentation of planar regions, e.g., detecting objects in a point cloud, then the final refinement step may be skipped, which could increase the frame rate to more than 50Hz for 640 by 480 points.

First the notations are clarified. $\mathcal{F}$ denotes a complete frame of an organized point cloud of M rows and N columns. $\mathcal{B},\mathcal{C}$ represent coarse and refined segmentation respectively, i.e., each element $\mathcal{B}_k$ / $\mathcal{C}_l$ of $\mathcal{B}$ / $\mathcal{C}$ is a segment—a set of 3D points $\mathbf{p}_{i,j}$. Meanwhile Π,Π' are sets of plane equations corresponding to $\mathcal{B},\mathcal{C}$, respectively. Also note that each node v of a graph G is a set of 3D points and each undirected edge uv denotes the neighborhood of segments u,v in the image space.

Algorithm 2–1: Fast Plane Extraction.

1: **function** FastPlaneExtraction($\mathcal{F}$)
2: $G \leftarrow$ InitGraph($\mathcal{F}$)
3: $(\mathcal{B},\Pi) \leftarrow$ AHCluster(G)
4: $(\mathcal{C},\Pi') \leftarrow$ Refine($\mathcal{B},\Pi$)
5: **return** $(\mathcal{C},\Pi')$

2.4.1 Graph Initialization

As mentioned in section 2.3.2, PEAC has a requirement of non-overlapping node initialization, represented in lines 3 to 5 of Algorithm 2–2. This step uniformly divides the point cloud into a set of initial nodes of the size $\mathsf{H}\times\mathsf{W}$ in the image space. The requirement causes PEAC to lose

the advantage of automatically detecting boundaries of planes. To properly segment planes using AHC under this restriction, the following types of nodes and corresponding edges are removed from the graph, which is illustrated using an example in Figure 2-4 (‘o’ shows nodes with missing data point; ‘x’ shows nodes with depth discontinuity; black dot shows nodes with too large plane fitting MSE; and ‘B’ shows nodes located at the boundary region between two connected planes):

1. **Nodes Having High MSE:** Non-planar regions lead to high plane fitting MSE, which are simply removed.
2. **Nodes Containing Missing Data:** Because of the limitation of the sensor, some regions of the scene might not be sensed correctly, leading to missing data (e.g., the glass window behind the shutter).
3. **Nodes Containing Depth Discontinuities:** These nodes contain two sets of points lying on two surfaces that are not close in 3D but are close in the image space (usually one surface partially occludes the other, e.g., the monitor occludes the wall behind). If principle component analysis (PCA) is performed on points belonging to this node for plane fitting, the fitted plane will be nearly parallel to the line-of-sight direction and thus still have a small MSE. Merging this "outlier" node with its neighbor node will have bad effect on the plane fitting result because of the well-known issue of over-weighting outliers in least-squares methods.
4. **Nodes at Boundary Between Two Planes:** These nodes contain two sets of points close to each other in 3D but lying on two difference planes (e.g., the corner of the room), which will decrease the plane fitting accuracy if they are merged to one of the planes.

Algorithm 2–2: Graph Initialization.

function InitGraph($\mathcal{F}$)
  $G \leftarrow (V \leftarrow \varnothing, E \leftarrow \varnothing)$
  for $i \leftarrow 1, \lceil \frac{\mathsf{M}}{\mathsf{H}} \rceil$ **do** ▷ initialize nodes
    for $j \leftarrow 1, \lceil \frac{\mathsf{N}}{\mathsf{W}} \rceil$ **do**
      $v_{i,j} \leftarrow \{\mathbf{p}_{k,l}\} \subset \mathcal{F}, k = (i-1)\mathsf{H}+1, \cdots, \min(i\mathsf{H}, \mathsf{M}), l = (j-1)\mathsf{W}+1, \cdots, \min(j\mathsf{W}, \mathsf{N})$
      if RejectNode($v_{i,j}$) **then**
        $v_{i,j} \leftarrow \varnothing$
      $V \leftarrow V \bigcup \{v_{i,j}\}$
  for each $v_{i,j} \in V$ **do** ▷ initialize edges
    if $\neg$RejectEdge($v_{i,j-1}, v_{i,j}, v_{i,j+1}$) **then**
      $E \leftarrow E \bigcup \{v_{i,j-1}v_{i,j}, v_{i,j}v_{i,j+1}\}$
    if $\neg$RejectEdge($v_{i-1,j}, v_{i,j}, v_{i+1,j}$) **then**
      $E \leftarrow E \bigcup \{v_{i-1,j}v_{i,j}, v_{i,j}v_{i+1,j}\}$
  return G

function RejectNode(v)
  if v contains missing data point **then return true**
  else if any point $\mathbf{p}_{i,j} \in v$ is depth-discontinuous with any of its 4 neighbor points **then return true**
  else if MSE(v) > $\mathsf{T}_{\mathsf{MSE}}$ **then return true**
  else return false

function RejectEdge(v_a, v_b, v_c)
  if $v_a = \varnothing \vee v_b = \varnothing \vee v_c = \varnothing$ **then return true**
  else if included angle of plane fitting normal of v_a and v_c is greater than $\mathsf{T}_{\mathsf{ANG}}$ **then return true**
  else return false

function MSE(v)
  if $v = \varnothing$ **then return** $+\infty$
  return the plane fitting MSE for all $\mathbf{p}_{i,j} \in v$

The functions *RejectNode* and *RejectEdge* in Algorithm 2–2 are designed to reduce the influence of these four types of bad initial nodes. The *RejectNode* function removes the first three types of

bad nodes (and thus the points inside) from the graph, while the *RejectEdge* function is for mitigating influence of the fourth type of bad nodes.

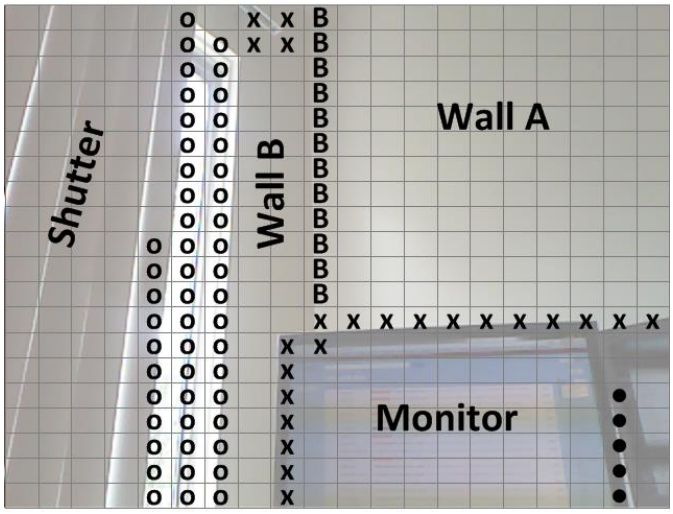


Figure 2-4: Examples of bad initial nodes.

It is interesting to note that the gain in this non-overlapping "disadvantage" is the avoidance of per-point normal estimation. This initialization step can be seen as treating all points inside a node as if they have a common plane normal. This is an important reason for the speed improvement of this method compared to other state-of-the-art methods which often spend a large portion of time in the normal estimation for each point.

2.4.2 Agglomerative Hierarchical Clustering

As shown in Algorithm 2–3, the AHC in this PEAC algorithm is almost the same as that in the line regression, except that it is operated on a graph instead of a double linked list. First a min-heap data structure is built for efficiently finding the node with the minimum plane fitting MSE. It then repeats finding a node v that currently has the minimum plane fitting MSE among all nodes in the graph and merging it with one of its neighbor nodes u_{best} that results in the minimum merging MSE (recall that each node in the graph is a set of points; so the merging

MSE is the plane fitting MSE of the union of the two sets u_{merge}). If this minimum merging MSE exceeds some predefined threshold T_{MSE} (not necessarily a fixed parameter as explained later in section 2.4.3), then a plane segment v is found and extracted from the graph; otherwise the merged node u_{merge} is added back to the graph by edge contraction between v and u_{best}.

Algorithm 2–3: Agglomerative Hierarchical Clustering.

```
1:  function AHCluster(G = (V, E))
2:      Q ← BuildMinMSEHeap(V)
3:      B ← ∅, Π ← ∅
4:      while Q ≠ ∅ do
5:          v ← PopMin(Q)
6:          if v ∉ V then continue                          ▷ v was merged previously
7:          u_best ← ∅, u_merge ← ∅
8:          for each u ∈ N(v) ≡ {u | uv ∈ E} do             ▷ for all neighbor nodes of v
9:              u_test ← u ⋃ v                              ▷ merge attempt
10:             if MSE(u_test) < MSE(u_merge) then
11:                 u_best ← u, u_merge ← u_test
12:         if MSE(u_merge) ⩾ T_MSE then                    ▷ merge fail
13:             if |v| ⩾ T_NUM then                         ▷ extract node v
14:                 B ← B ⋃ {v}, Π ← Π ⋃ Plane(v)
15:             E ← E ⧹ E(v) ≡ {uv | u ∈ N(v)}              ▷ remove all edges to v
16:             V ← V ⧹ {v}                                 ▷ reject small node
17:         else                                            ▷ merge success
18:             Insert(Q, u_merge)
19:             E ← E ⋃ {u_merge w | w ∈ N(v) ⋃ N(u_best) ⧹ {v, u_best}}   ▷ edge contraction
        ⧹ E(u_best) ⧹ E(v)
20:             V ← V ⋃ {u_merge} ⧹ {v, u_best}             ▷ update nodes
21:     return (B, Π)

22: function Plane(v)
23:     return plane equation fitted from points in v by PCA
```

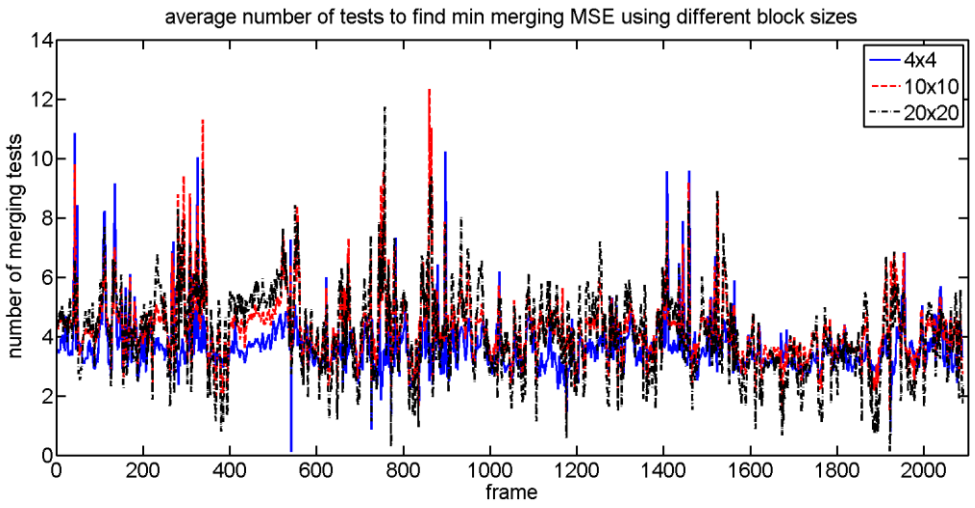


Figure 2-5: Average number of merging tests per frame.

As mentioned in section 2.3.2, PEAC requires a larger number of merging attempts than the line regression. However, it turns out to be still quite efficient and the clustering process can be done in $O(n \log n)$ time in practice. Figure 2-5 experimentally shows the average number of merging attempts during AHC per frame (during 2102 frames of 640 by 480 Kinect point clouds). As can be seen, irrespective of the initial node size (and thus the initial number of nodes), this number stays small. This may be explained by the fact that the graph constructed from Algorithm 2–3 is a planar graph. From graph theory one knows that the average node degree of a planar graph is strictly less than 6. Since the initial graph is planar and merging nodes by edge contraction does not change its planarity, during the whole process of AHC the average node degree is always less than 6. Also, the plane fitting MSE of a large segment is larger than that of a smaller segment, if errors are drawn from the same Gaussian distribution. Thus the AHC process tends to balance the size of all the segments, because it always tries to grow the size of the node with the minimum plane fitting MSE and then switches to other smaller nodes. Therefore, it will not stick to growing a large node (which implies large node degree since it has large boundary), otherwise the average number of merging tests will be much larger. Based on this observation, lines 6 to 20

in Algorithm 2–3 can be done in a constant time irrespective of the initial number of nodes. The $O(n \log n)$ complexity only arises from maintaining the min-heap structure.

2.4.3 Implementation Details

There are several implementation details to improve the speed and accuracy for this fast coarse segmentation:

1. A disjoint set data structure is used for tracking the point membership of each initial node $v_{i,j}$ during the node merging in AHC.
2. As in the line regression, all nodes maintain the first and second order statistics of all the belonging points, i.e., $\sum x_{i,j}, \sum y_{i,j}, \sum z_{i,j}, \sum x_{i,j}^2, \sum y_{i,j}^2, \sum z_{i,j}^2, \sum x_{i,j} y_{i,j}, \sum y_{i,j} z_{i,j}, \sum z_{i,j} x_{i,j}$, such that merging two nodes and calculating its plane equation and MSE through PCA is a constant time operation.
3. The function for determining the depth discontinuity in *RejectNode* of Algorithm 2–2 depends on sensor noise characteristics. For Kinect sensors, the following function is used as suggested in (Holzer et al. 2012) and Point Cloud Library (PCL)[3]:

$$f(\mathbf{p}_a, \mathbf{p}_b) = \begin{cases} 1 & |z_a - z_b| > 2\alpha(|z_a| + 0.5) \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

   The unit of z here (and throughout this chapter) is millimeter and the parameter α was used between 0.01 and 0.02.
4. The threshold $\mathsf{T}_{\mathrm{MSE}}$ for extracting segments is also sensor dependent. For Kinect, the following equation adapted from (Khoshelham and Elberink 2012) is used:

[3] http://www.pointclouds.org

$$T_{MSE} = (\sigma z^2 + \epsilon)^2 \tag{2.2}$$

where $\sigma = 1.6 \times 10^{-6}$ and ϵ between 3 and 8 is used. Similarly, T_{ANG} can also be changed depending on depth.

5. The initial node should be close to a square shape in the image space, i.e., $W \approx H$. If a strip-like shape is used, either $W \gg H$ (e.g., $W = 20, H = 2$) or $H \gg W$, the PCA on the initial node will result in wrong plane normal direction which is usually almost perpendicular to the line-of-sight direction. Consequently the following AHC will fail to segment planes correctly.

2.5 Segmentation Refinement

For many applications, the coarse plane segmentation obtained in the previous section might not be enough, especially if the applications use the boundaries of planes (Pathak et al. 2010; Fernandez-Moral et al. 2013) or require higher accuracy of the estimated plane equations. Thus refinement on the coarse segmentation $\mathcal{B}$ is performed.

Three types of artifacts are expected in the coarse segmentation, as shown in Figure 2-6 (where the bottom row shows the corresponding refined segmentations):

1. **Sawtooth:** Usually at the boundary between two connected planes (e.g., purple and yellow segments of the top-left part).
2. **Unused Data Points:** Usually at the boundary of occlusion or missing data node (e.g., between lamp and wall of the top-right part).
3. **Over-Segmentation:** Usually between two object's occlusion boundary (e.g., purple and red segments of the top-right part).

Algorithm 2–4: Segmentation Refinement.

1: **function** Refine($\mathcal{B}, \Pi$)
2: $Q \leftarrow \varnothing$ ▷ initialize queue of boundary points
3: $\mathcal{R} \leftarrow \varnothing$ ▷ points to be refined
4: $G' \leftarrow (V' \leftarrow \varnothing, E' \leftarrow \varnothing)$ ▷ graph for final merge
5: **for each** $\mathcal{B}_k \in \mathcal{B}$ **do** ▷ 1. erode each segment
6: $\mathcal{R}_k \leftarrow \varnothing, \mathcal{R} \leftarrow \mathcal{R} \bigcup \mathcal{R}_k$
7: **for each** initial node $v_{i,j} \subset \mathcal{B}_k$ **do**
8: **if** $v_{i-1,j} \bigcup v_{i+1,j} \bigcup v_{i,j-1} \bigcup v_{i,j+1} \not\subset \mathcal{B}_k$ **then**
9: $\mathcal{B}_k \leftarrow \mathcal{B}_k \setminus v_{i,j}$ ▷ erode border node
10: **for each** point $\mathbf{p}_{s,t}$ on the boundary of $\mathcal{B}_k$ **do**
11: Enqueue($Q, (\mathbf{p}_{s,t}, k)$)
12: **if** $\mathcal{B}_k \neq \varnothing$ **then** $V' \leftarrow V' \bigcup \{\mathcal{B}_k\}$
13: **while** $Q \neq \varnothing$ **do** ▷ 2. region grow from boundary
14: $(\mathbf{p}_{s,t}, k) \leftarrow$ Dequeue(Q)
15: **for** $\mathbf{p}_{i,j} \in \{\mathbf{p}_{s-1,t}, \mathbf{p}_{s+1,t}, \mathbf{p}_{s,t-1}, \mathbf{p}_{s,t+1}\}$ **do**
16: **if** $\mathbf{p}_{i,j} \in (\mathcal{B}_k \bigcup \mathcal{R}_k) \vee \text{Dist}(\mathbf{p}_{i,j}, \Pi_k)^2 \geqslant 9\text{MSE}(\mathcal{B}_k)$ **then**
17: **continue**
18: **if** $\exists l, \mathbf{p}_{i,j} \in \mathcal{R}_l$ **then**
19: $E' \leftarrow E' \bigcup \{\mathcal{B}_k \mathcal{B}_l\}$ ▷ connect nodes
20: **if** $\text{Dist}(\mathbf{p}_{i,j}, \Pi_k) < \text{Dist}(\mathbf{p}_{i,j}, \Pi_l)$ **then**
21: $\mathcal{R}_l \leftarrow \mathcal{R}_l \setminus \{\mathbf{p}_{i,j}\}, \mathcal{R}_k \leftarrow \mathcal{R}_k \bigcup \{\mathbf{p}_{i,j}\}$
22: Enqueue($Q, (\mathbf{p}_{i,j}, k)$)
23: **else**
24: $\mathcal{R}_k \leftarrow \mathcal{R}_k \bigcup \{\mathbf{p}_{i,j}\}$
25: Enqueue($Q, (\mathbf{p}_{i,j}, k)$)
26: **for each** $\mathcal{R}_k \in \mathcal{R}$ **do**
27: $\mathcal{B}_k \leftarrow \mathcal{B}_k \bigcup \mathcal{R}_k$ ▷ update each coarse segment
28: $(\mathcal{C}, \Pi') \leftarrow$ AHCluster(G') ▷ 3. final merge
29: **return** $(\mathcal{C}, \Pi')$

Sawtooth artifacts cause small amount of outliers to be included in estimation, whereas unused data points and over-segmentation cause less inliers to be used. All of the artifacts produce inaccurate plane boundaries and slightly decrease the accuracy of the estimated plane equation.

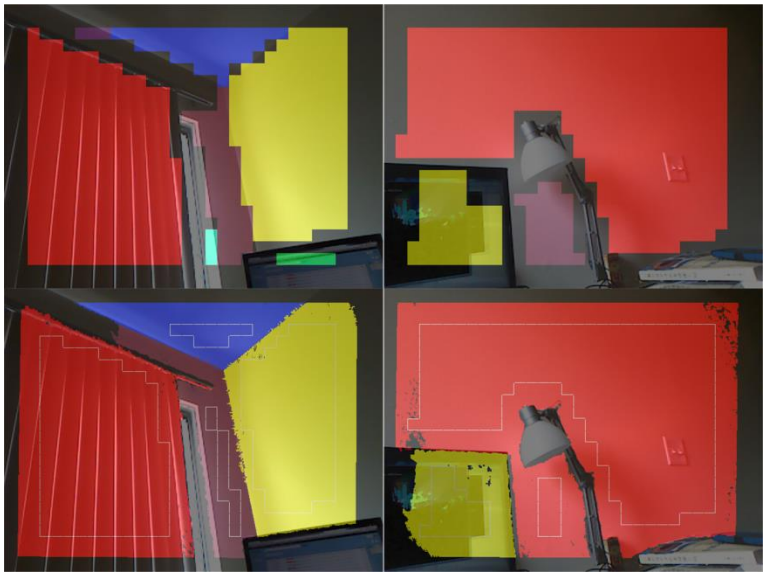


Figure 2-6: Artifacts of coarse segmentations and corresponding refinement.

The solution to them is described in Algorithm 2–4. Since sawtooth artifacts are almost always observed at the boundary regions of $\mathcal{B}$, erosion of boundary regions of each segment can effectively eliminate them (lines 5 to 12). Then pixel-wise region growing is started from all new boundary points to assign all unused data points to its closest plane that is extracted previously (lines 13 to 27). During the region growing the 4-connected neighborhoods are discovered for each segment $\mathcal{B}_k$, which form a new graph G'. Finally applying AHC again on this very small graph (usually less than 30 nodes) fixes the over-segmentation artifact (line 28).

2.6 Experiments and Discussion

To comprehensively evaluate PEAC's performance in terms of robustness, time, and accuracy, three sets of experiments was conducted, as described in the following subsections. This

algorithm was implemented in C/C++. For PCA, the efficient 3 by 3 matrix eigenvalue decomposition routine described in (Kopp 2008)[4] was used. All experiments were conducted on an ordinary laptop with Intel Core i7-2760QM CPU of 2.4GHz and RAM of 8GB. No multi-threading or any other parallelism such as OpenMP or GPU was used in this implementation.

2.6.1 Simulated Data

Similar to the influence of noise simulation in (Georgiev et al. 2011), PEAC's robustness was tested on a simulated depth map with 20 different levels of uniformly distributed noise of magnitude $E = 10l, l = 0,\ldots,20$ (noise unit: mm; ground truth depth ranges from 1396mm to 3704mm). After the noise was added to the depth map, it was converted to an organized point cloud and fed into the algorithm ($\mathsf{W} = \mathsf{H} = 20, \mathsf{T}_{\mathrm{MSE}} = 50^2$). As shown in Figure 2-7, PEAC can reliably detect all of the 4 planes for $l = 0,\ldots,14$, and starts to over-segment after that. Yet even when E=200mm PEAC was able to detect major planes in the scene.

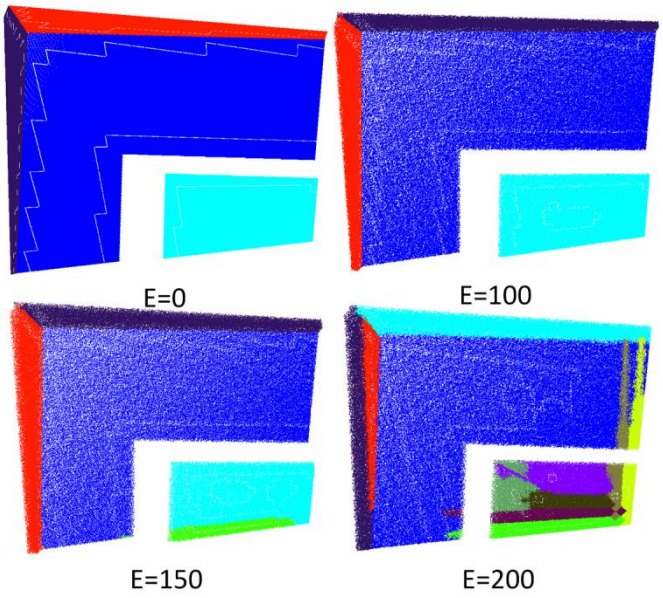


Figure 2-7: Plane extraction results on simulated data.

[4] Implementation available for download at http://www.mpi-hd.mpg.de/personalhomes/globes/3x3/

2.6.2 Real-World Kinect Data

To measure the processing speed of PEAC, 2102 frames of 640 by 480 pixel real-world Kinect data were collected in an indoor scene, partly shown in Figure 2-1 and Figure 2-6. Then they were processed with PEAC using 12 different initial node sizes ($T_{NUM} = 800$, $\alpha = 0.02, \epsilon = 8$ mm, T_{ANG} increases linearly from $15°$ at z=500mm to $90°$ at z=4000mm). As shown in Figure 2-8, with initial node size of 10 by 10, even with refinement, PEAC took only **27.3 ± 6.9 ms** in average to process a frame of 640 by 480 pixel Kinect data, achieving more than **35Hz** frame rate. To the best of my knowledge, this is much faster than existing state-of-the-art algorithms.

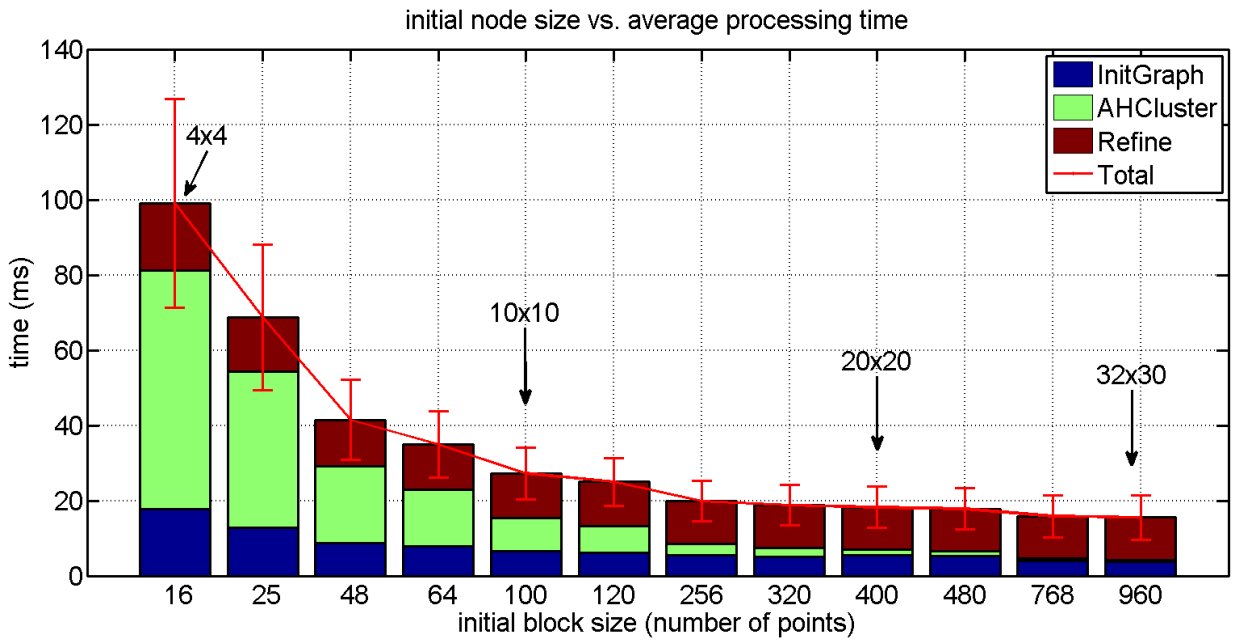


Figure 2-8: Average processing time of the proposed PEAC algorithm.

2.6.3 SegComp Datasets

The accuracy of PEAC was evaluated using the SegComp datasets (Hoover et al. 1996). Both the ABW ($W = H = 4, T_{MSE} = 1, T_{ANG} = 60^{\circ}, T_{NUM} = 160, \alpha = 0.1$) and PERCEPTRON ($W = H = 8, T_{MSE} = 2.1, T_{ANG} = 45^{\circ}, T_{NUM} = 240, \alpha = 0.03$) datasets of planar scenes were

experimented. Typical segmentation results of ABW and PERCEPTRON test datasets are shown in Figure 2-9. The estimated plane normal deviated from the ground truth was $(1.7 \pm 0.1)^{\circ}$ for ABW-TEST (top row) and $(2.4 \pm 1.2)^{\circ}$ for PERCEPTRON-TEST (bottom row). Again, white dash lines are the segmentation boundary before the region-grow-based refinement. The detailed benchmark results using the evaluation tool provided by SegComp are shown in Table 2–1. The results other than the one presented here were obtained from (Gotardo et al. 2003; Oehler et al. 2011; Holz and Behnke 2012). As can be seen, PEAC's performance is comparable to the state-of-the-art in terms of segmentation accuracy as well as plane orientation estimation (higher accuracy can be achieved when the ABW and PERCEPTRON sensors' noise characteristics are known), especially considering the fact that PEAC's frame rate is much higher (Note that the SegComp datasets were not used for speed evaluation since many methods in Table 2–1 were evaluated a decade ago with much lower computational power).

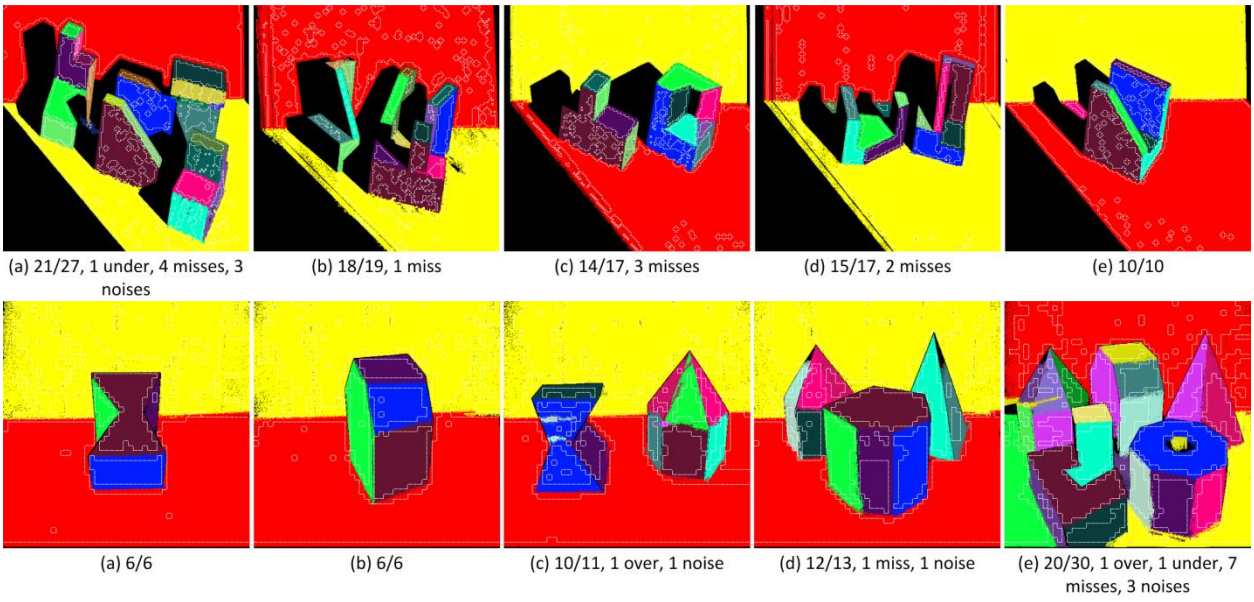


Figure 2-9: Plane extraction on SegComp datasets.

2.7 Conclusions

A novel fast plane extraction algorithm for organized point clouds was presented, achieving more than 35Hz frame rate on 640 by 480 point clouds while providing accurate segmentation. In the future the algorithm is to be extended to non-organized point clouds as well so as to fast extraction of other primitives such as spheres and cylinders.

Table 2–1: Benchmarking results on the SegComp datasets.

Approach	Regions in ground truth	Correctly detected	Orientation deviation (°)	Over-segmented	Under-segmented	Missed (not detected)	Noise (non-existent)
SegComp ABW data set (30 test images) by Hoover et al. (1996), assuming 80% pixel overlap as in (Gotardo et al. 2003)							
USF	15.2	12.7 (83.5%)	1.6	0.2	0.1	2.1	1.2
WSU	15.2	9.7 (63.8%)	1.6	0.5	0.2	4.5	2.2
UB	15.2	12.8 (84.2%)	1.3	0.5	0.1	1.7	2.1
UE	15.2	13.4 (88.1%)	1.6	0.4	0.2	1.1	0.8
OU	15.2	9.8 (64.4%)	–	0.2	0.4	4.4	3.2
PPU	15.2	6.8 (44.7%)	–	0.1	2.1	3.4	2.0
UA	15.2	4.9 (32.2%)	–	0.3	2.2	3.6	3.2
UFPR	15.2	13.0 (85.5%)	1.5	0.5	0.1	1.6	1.4
Oehler et al.	15.2	11.1 (73.0%)	1.4	0.2	0.7	2.2	0.8
Holz et al.	15.2	12.2 (80.1%)	1.9	0.8	0.1	0.9	1.3
Enjarini et al.	15.2	13.2 (86.8%)	–	0.3	0.2	1.1	1.8
PEAC	**15.2**	**12.8 (84.2%)**	**1.7**	**0.1**	**0.0**	**2.4**	**0.7**
SegComp PERCEPTRON data set (30 test images) by Hoover et al. (1996), assuming 80% pixel overlap as in (Gotardo et al. 2003)							
USF	14.6	8.9 (60.9%)	2.7	0.4	0.0	5.3	3.6
WSU	14.6	5.9 (40.4%)	3.3	0.5	0.6	6.7	4.8
UB	14.6	9.6 (65.7%)	3.1	0.6	0.1	4.2	2.8
UE	14.6	10.0 (68.4%)	2.6	0.2	0.3	3.8	2.1
UFPR	14.6	11.0 (75.3%)	2.5	0.3	0.1	3.0	2.5
Oehler et al.	14.6	7.4 (50.1%)	5.2	0.3	0.4	6.2	3.9
Holz et al.	14.6	11.0 (75.3%)	2.6	0.4	0.2	2.7	0.3
Enjarini et al.	14.6	10.7 (73.3%)	–	0.4	0.1	3.6	4.4
PEAC	**14.6**	**8.9 (60.9%)**	**2.4**	**0.2**	**0.2**	**5.1**	**2.1**

Chapter 3
Plane Registration Leveraged by Global Constraints

"We all need people who will give us feedback. That's how we improve."—Bill Gates

3.1 Introduction

The ability to recover a user's pose (i.e., position and orientation within a certain coordinate system) is critical in many engineering domains such as AR, robotics, context-aware computing, and computer vision. In AR, for example, this task is termed as the "registration" problem (Azuma 1997). In robotics, this task is closely related to "simultaneous localization and mapping" (SLAM) (Klein and Murray 2007; Thrun 2008). In computer vision, "structure from motion" (SFM) algorithms are designed to solve this problem with little or no prior knowledge about the environment (Bao and Savarese 2011). Context-aware engineering applications also face a similar problem where the positioning part is more relevant (Akula et al. 2011). In cinematography, the problem is called "move matching". For simplicity, the author will refer to all of these as the registration problem in this research.

Despite the rapid development of sensor technology—such as the global positioning system (GPS) and inertial measurement units (IMU), as well as angle sensors like the digital magnetic sensor, gyroscope, and accelerometer—this problem remains a challenge. A GPS signal is hardly available indoors, IMU suffers from the drifting effect after a while (Akula et al. 2011), and a

magnetic sensor can be hugely affected by the changing environment, especially in challenging environments such as construction sites with all kinds of machines moving around, needless to mention the sensor's annoying jitter effect.

To overcome these technical insufficiencies, especially for indoor environments, infrastructure-based technology has been studied, such as RFID-based indoor tracking (Bekkali et al. 2007) and wireless local area network (WLAN)-based indoor positioning (Khoury and Kamat 2009; Li et al. 2006). Yet these technologies are costly, inefficient, or sensitive to the environment, and lots of work has to be put into the system calibration stage. Further, these technologies' general inability to recover a user's orientation is troublesome for 3D visualization.

However, beyond all of those technologies, it is very interesting to note that a human being can figure out where s/he is to a certain degree of accuracy, given that s/he is familiar enough with that specific region of environment, mostly with the help of visual clues. This intuition inspires us to look into the developments in the computer vision community. Meaningful new tools that confront this registration problem could be developed if the insight of this human-possessed ability can be somehow captured and automated.

Following this motivation, this chapter presents a new visual registration method called KEG planar object tracker, which essentially recovers the pose of the user, i.e. camera, in real-time from a set of planar markers whose own poses are known, capturing the idea that this tracker is familiar enough with the environment so as to perform an estimation of position and orientation, just as humans do. Section 3.2 will briefly introduce the state-of-the-art methods in visual registration. Section 3.3 will explain in detail how the first computation step of this tracker works. Then section 3.4 will explain another class of planar marker-based algorithms that

eventually serve as the second step of this tracker. Section 3.5 will explain the main contribution in this chapter—an efficient improvement based on the previous two classes of algorithm frameworks. After that, section 3.6 will describe several experiments that demonstrate the superiority, under different objective quality measures, of this algorithm framework. Also, section 3.7 will describe two novel AEC applications that deploy this tracking algorithm. Finally, conclusions are drawn in section 3.8.

3.2 Previous Work

The visual registration problem is actively studied in the computer vision community, and several algorithms have been proposed to address it. Based on their different assumptions on the environment (i.e. the surrounding world where visual registration is going to be performed), these algorithms can be classified into two groups (Lepetit and Fua 2005): known environment vs. unknown environment (See Figure 3-1. The proposed method in this chapter can be classified under natural marker-based methods). The first group of algorithms is easier to design since the only unknown is the user's pose. In the second group, an estimation of the environment has to be done along with the registration, which is naturally related to the visual SLAM problem that is being explored in the robotics community (Davison et al. 2007).

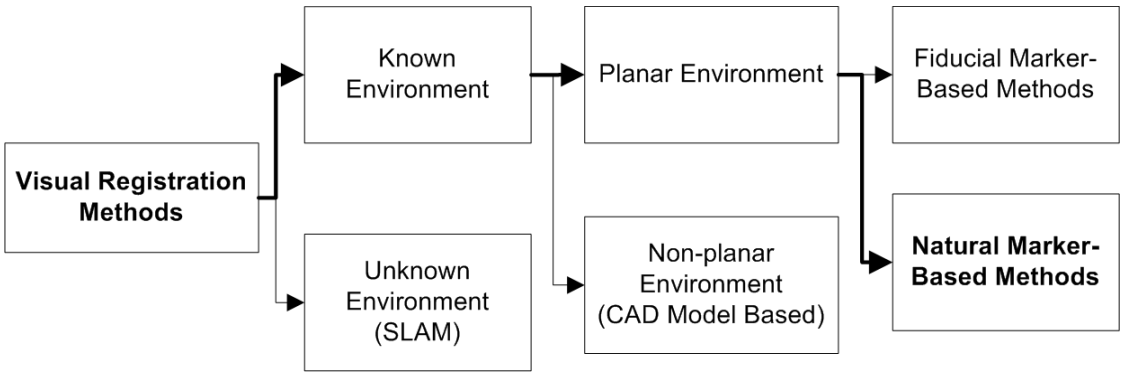


Figure 3-1: A brief taxonomy of visual registration methods.

Generally, SLAM-based methods are more desirable since very few assumptions are made on the environment, which inherently makes those algorithms infrastructure-independent and thus enlarges their sphere of application. Non-visual SLAM-based methods typically rely on measurements from laser or light detection and ranging (lidar), where the cost and even the weight of those devices could be potential disadvantages of applying such technologies. Especially for context-aware computing in civil infrastructure applications (e.g. facility management, bridge inspection), weight is always an important consideration since human inspectors will not be able to comfortably use overly heavy devices. Although those developing visual SLAM algorithms can avoid these drawbacks by using a standard and lightweight web-cam as the main sensor, currently many limitations exist in such an approach (Klein and Murray 2007), including the requirement of high computational power and small range restriction.

Different from the visual SLAM methods, the known environment methods have been well studied, and many powerful algorithms have been proposed in the last two decades. This makes it more realistic to apply them for solving real-world engineering issues. Within this class of methods, they can be further categorized into two groups: planar environment vs. non-planar environment. The first group is again easier to design because of the simple assumption made regarding the environment—a planar structure with known visual features. While the second group typically assumes known 3D structures with known visual features (usually CAD models), they are more often applied in a controlled environment with limited space, such as a small manufacturing workspace (Drummond and Cipolla 2002).

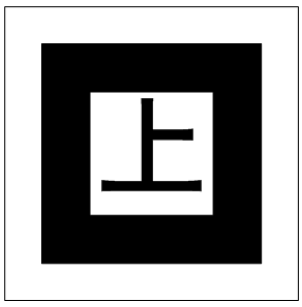
The authors choose to take advantage of plane-based methods since planar structures are abundant in buildings, construction sites, and other human-made environments where engineering operations are conducted, which makes this type of method very convenient to

apply. In addition, a planar structure can simply be an image printed out on a piece of paper and attached to a wall/floor of a corridor, with nearly zero cost. All of those advantages make this method ideal for application, and merit its investigation.

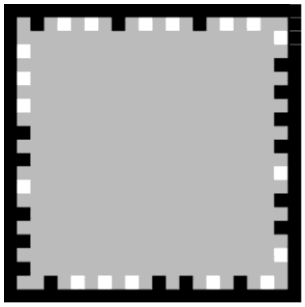
Plane-based methods can be further classified based on different visual features they adopt: fiducial marker vs. natural marker. In the following two subsections, the two types of markers will be introduced.

3.2.1 Fiducial Marker

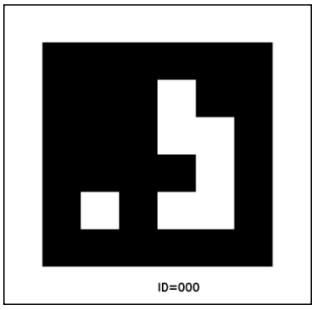
A fiducial marker is composed of a set of visual features that are "easy to extract" and "provide reliable, easy to exploit measurements for the pose estimation" (Lepetit and Fua 2005). Usually those features are a set of black and white patterns forming simple geometry by circles, straight lines, or sharp corners and edges.



(a) ARToolKit (Kato and Billinghurst 1999)



(b) Frame marker (Wagner et al. 2008)



(c) AptilTag (Olson 2011)

Figure 3-2: Examples of fiducial markers.

Fiducial markers have been widely used in AR community for their simplicity and minimal computational requirement. ARToolKit (Kato and Billinghurst 1999) is one of the earliest and most widely used fiducial marker-based AR systems. Since every ARToolKit marker is bounded by a wide black bounding box, the detection of the marker and registration can be solved by

simply taking a threshold of the current input image and then discovering the four outer edges and corners so as to further estimate the camera's pose. A similar method is also adopted in the "frame marker" proposed by (Wagner et al. 2008), as shown in Figure 3-2 (a) and (b).

Different from the simple image-processing algorithms in ARToolKit for detecting marker and estimating pose, AprilTag (Olson 2011) clusters every pixel of the current input image based on its gradient and location, and then extracts multiple line segments as the boundary of each cluster. Finally, different quadrangles are discovered and used to decode a potential tag/marker ID, as shown in Figure 3-2 (c). This newly proposed method is proved through experiments to be superior to ARToolKit and many other state-of-the-art fiducial marker-based methods (Olson 2011).

3.2.2 Natural Marker

Distinct from a fiducial marker, a natural marker does not require special predefined visual features. Instead, it treats any visual features in the same way. In this sense, any common image, ranging from a natural view to a company logo, can immediately be used as a natural marker, of course except for some images with fixed spatial frequency (i.e. images with the same color everywhere or repeated patterns). This major difference makes it much easier to set up a natural marker than a fiducial one. Users do not need to separately design special markers; they can simply take advantage of any meaningful pictures related to the problem of interest.

In addition, one major downside to a fiducial marker lies in the fact that it usually depends on the four corner points or edges of the marker's quadrangle to do further registration estimation, which is the reason that fiducial marker-based methods will fail even if one corner is not within

view. This disadvantage does not exist in natural marker-based methods; in fact natural markers do not even require a marker image to be rectangular.

A natural marker uses more than four points to perform registration estimation. By detecting an appropriate number of feature points in both marker image and current image, a natural marker-based algorithm will then try to establish correspondences between these two sets of feature points. Once correspondences are established, further estimation can easily be made. Thus lots of algorithms have been proposed to tackle the main issue and most computational intensive part here, i.e. how to find as many correct correspondences as quickly as possible. Again, by the fundamental difference in the way they treat input images, these algorithms form two groups: one group treats each input image independently, which is referred to as a detection-based method, such as (Lowe 2004; Ozuysal et al. 2007); the other group needs two or more consecutive images as input, which is referred to as a tracking-based method, such as (Shi and Tomasi 1994; Lucas and Kanade 1981). Since the proposed method in this chapter evolves from both these algorithm groups, the following sections (3.3 and 3.4) will explain in detail the process framework of each type of algorithms, as well as how it inspires and is jointly adapted to the proposed algorithm framework.

3.3 Homography From Detection

In either fiducial marker-based or natural marker-based algorithms, the fundamental task is to find the transformation between the marker image plane and the current camera plane which contains that current image frame. Such a transformation, which is called homography, maps points on the marker image to their corresponding points on the current image frame with the following equation:

$$s[x', y', 1]^{\mathbf{T}} = \mathbf{H}[x, y, 1]^{\mathbf{T}} \tag{3.1}$$

where **H** is a 3 by 3 matrix representing the homography, (x, y) and (x', y') are the corresponding points on the two images, and s is an unknown scaling parameter.

In fact, the general idea behind a plane-based registration algorithm is the fact that homography between two planes encodes the orientation and position information of one plane relative to another. From this perspective, registration is equivalent to finding the homography between the marker plane and the current camera plane. From projective geometry, one knows that with at least four point correspondences between two planes, their homography can be uniquely determined by solving a set of linear equations (Hartley and Zisserman 2000).

More complicated than fiducial marker-based algorithms, which take advantage of simple patterns to find correspondences and then estimate homography, natural marker-based algorithms require a lot more effort to solve a correspondence problem.

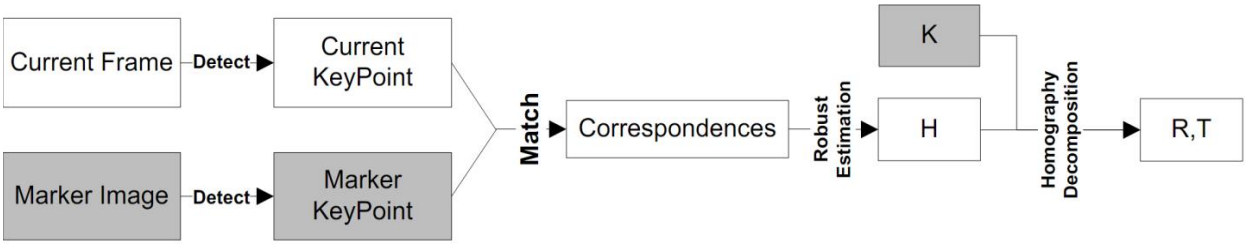


Figure 3-3: Homography-from-detection algorithm framework.

Figure 3-3 shows the generic algorithm framework of the homography-from-detection type of methods. The gray components need be loaded or calculated once during a computation, while the white components need to be updated for each new frame. **H** is the homography between the current frame and the marker image. **K** is the camera calibration matrix storing the focal length and some other intrinsic parameters, which can be calibrated in advance. **R** is the rotation matrix

representing camera orientation, and **T** is the translation vector representing the position of camera center. For each incoming image frame, the first step is to detect a set of keypoints. Also, at the very beginning, a fixed set of keypoints has to be detected on the marker image. Interest point detection algorithms are usually applied in this step, such as Harris corner detector (Harris and Stephens 1988), FAST (Rosten and Drummond 2006), Difference of Gaussian (DoG) (Lowe 2004), and Laplacian of Gaussian (LoG) (Lindeberg 1998).

The second step involves a matching problem, i.e. finding corresponding points between two sets of keypoints based on their local appearance. Among the state-of-the-art algorithms, the Scale Invariant Feature Transform (SIFT) algorithm by (Lowe 2004) is perhaps the most famous and widely used nowadays. Although the SIFT algorithm works very well under large variation of visual conditions—illumination, scale, and rotation change among others—it is computationally time-consuming, which makes it impractical to be applied directly in real-time applications, such as a registration problem, even after applying lot of approximation to SIFT, as proposed in the SURF algorithm (Bay et al. 2008). Ozuysal et al. approach the matching problem from a very different perspective—matching as classification (Ozuysal et al. 2007). Their method, FERNs, differs from SIFT/SURF by the requirement of an offline training stage. Only after a long period of training using the marker image can FERNs recognize different keypoints on that particular marker under different visual conditions. Although the offline training requirement appears to be a disadvantage, FERNs enjoy the high-speed advantage. Recently, other faster methods (Rublee et al. 2011; Taylor et al. 2009) have also been proposed.

As mentioned, since most of these matching algorithms exploit a local feature descriptor, i.e. using image intensity information to describe a keypoint within only a limited neighboring region centered at that keypoint, mismatch is inevitable. In order to avoid most of these false

matches, a robust estimation algorithm, such as the famous RANdom Sample And Consensus (RANSAC) (Fischler and Bolles 1981) or its variants (Chum and Matas 2005; Torr and Zisserman 2000), is usually employed to estimate the homography.

3.3.1 Homography Decomposition

Once homography is found—through matrix decomposition techniques (Benhimane et al. 2005; Malis and Vargas 2007) the camera position, the translation vector **T**, and orientation—the rotation matrix **R**, can be calculated. A simple yet effective method was proposed by (Simon et al. 2000).

If a point's 3D coordinate is (*X, Y, Z*) and its image on the camera plane has a 2D coordinate of (*x, y*), and if it is assumed that the camera is already calibrated with known focal length and principal point position that is stored in the **K** matrix, the camera projection model is (Hartley and Zisserman 2000):

$$\begin{aligned} [x, y, 1]^{\mathbf{T}} &\sim \mathbf{P}[X, Y, Z, 1]^{\mathbf{T}} \\ &\sim \mathbf{K}[\mathbf{R}, \mathbf{T}][X, Y, Z, 1]^{\mathbf{T}}, \end{aligned} \tag{3.2}$$

where "~" means the two vectors are equal up to a scale parameter, i.e. equal in the sense of projective geometry. Since the marker image is a 2D plane, it can be set to be on the X–Y plane without loss of generality. Thus the above projection equation can be rewritten as ($\mathbf{r}_i$ is the i-th column of **R**):

$$\begin{aligned} [x, y, 1]^{\mathbf{T}} &\sim \mathbf{K}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{T}][X, Y, 0, 1]^{\mathbf{T}} \\ &\sim \mathbf{K}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{T}][X, Y, 1]^{\mathbf{T}} \\ &\sim \mathbf{H}[X, Y, 1]^{\mathbf{T}}. \end{aligned} \tag{3.3}$$

From the equation above, one can determine the following equations, which decompose the homography **H** between the current frame and the marker image into **R** and **T**, to be:

$$\begin{aligned} \mathbf{R} &= [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_1 \times \mathbf{a}_2], \\ \mathbf{T} &= \mathbf{a}_3, \end{aligned} \tag{3.4}$$

where $\mathbf{a}_i$ is the i-th column of matrix $\mathbf{K}^{-1}\mathbf{H} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$ and "$\times$" means the cross product. Note again that the actual matrix to be decomposed is $\mathbf{K}^{-1}\mathbf{H}$ rather than **H**, which means the camera needs to be calibrated in advance to get the **K** matrix.

3.4 Homography From Tracking

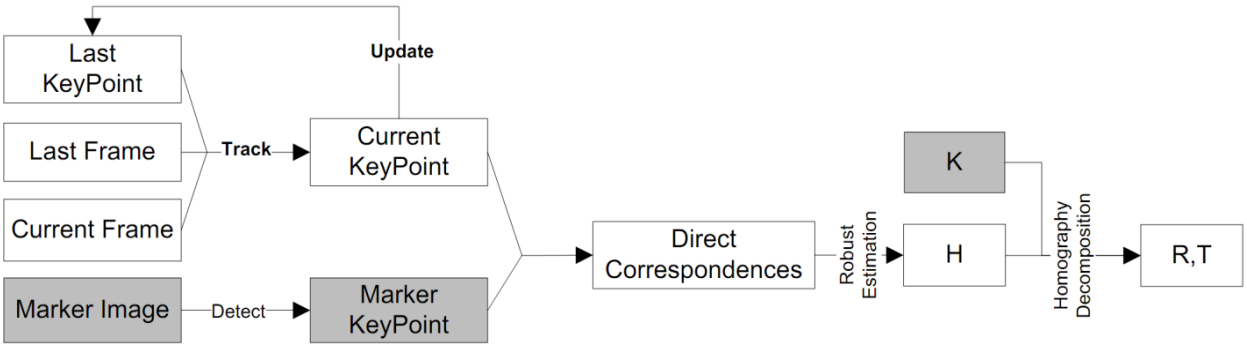


Figure 3-4: Homography-from-tracking algorithm framework.

As shown in Figure 3-4 homography-from-tracking type of methods explore relations between consecutive frames. Since images of two such frames usually look very similar, correspondences needed for homography estimation can easily be maintained by tracking each keypoint around its local neighborhood. Thus this type of methods can circumvent the hardest matching problem, since in this framework, matching between the marker keypoint and the current keypoint to get correspondences is only needed at the very beginning; after that, keypoint correspondences are maintained by a tracking algorithm. Next sections will briefly introduce two tracking algorithms

that play a crucial role in the proposed method: the Kanade-Lucas-Tomasi (KLT) feature tracker and Efficient Second-order Minimization (ESM) algorithm.

3.4.1 Kanade-Lucas-Tomasi Feature Tracker

Proposed by (Lucas and Kanade 1981), the KLT tracker's ultimate goal is to find the feature point displacement within two consecutive frames. It assumes that during these two frames, the local appearance of the feature point **x** does not change, and that the displacement is small. Thus in order to find the optimal displacement, these two assumptions of the invariance in local appearance can be mathematically represented as:

$$\begin{aligned} & \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in neighbor(\mathbf{z})} \left[I_1(\mathbf{x}+\mathbf{p}) - I_0(\mathbf{x}) \right]^2 \\ \Leftrightarrow \quad & \arg\min_{\mathbf{\Delta p}} \sum_{\mathbf{x}\in neighbor(\mathbf{z})} \left[I_1(\mathbf{x}+\mathbf{p}+\mathbf{\Delta p}) - I_0(\mathbf{x}) \right]^2, \end{aligned} \tag{3.5}$$

In equation (3.5), I_0 is the last frame, I_1 is the current frame, and **z** is the location of the point to be tracked in the image frame I_0. Note that **x**, **z**, **p**, **Δp** are all 2D column vectors. Here, an image is a function that takes a 2D point as input and outputs a real intensity value, i.e. $I : \Re^2 \to \Re$. In order to solve this minimization problem, the KLT tracker applies the first-order Taylor expansion on the image function I_1(**.**) around **x+p** in equation (3.5), leading to:

$$\arg\min_{\mathbf{\Delta p}} \sum_{\mathbf{x}\in neighbor(\mathbf{z})} \left[\nabla I_1(\mathbf{x}+\mathbf{p})\mathbf{\Delta p} + I_1(\mathbf{x}+\mathbf{p}) - I_0(\mathbf{x}) \right]^2, \tag{3.6}$$

where $\nabla I_1(\mathbf{x}+\mathbf{p})$ is the 1 by 2 image gradient of the current frame at location **x+p**.

This becomes a standard least-square problem with close-form solution. Since the KLT tracker uses a first-order approximation of the image function during the derivation, an iterative process

is adopted here (see Algorithm 3–1). Note that this is a local tracking algorithm since in equation (3.5) and (3.6) only the neighborhood of the tracked point is **z** considered.

Algorithm 3–1: KLT algorithm.

Input: a single keypoint **z** to be tracked from the last frame I_0 to the current frame I_1.

Initiate $\mathbf{p}=[0,0]^{\mathbf{T}}$.

Iterate:

1. Compute the error image $I_0(\mathbf{x}) - I_1(\mathbf{x}+\mathbf{p})$ around the neighbor of **z**;
2. Compute the image gradient of the current image ∇I_1 around the neighbor of **z**;
3. Compute $\Delta\mathbf{p}^*$ by solving equation (3.6);
4. Update $\mathbf{p} = \mathbf{p} + \Delta\mathbf{p}^*$

Until $\left\|\Delta\mathbf{p}^*\right\| \leqslant \epsilon$.

3.4.2 Efficient Second-order Minimization

As discussed previously, the KLT tracker actually formulates the point tracking problem as a minimization problem. Its motion model is very simple, a 2D displacement. Similarly, if using the 2D homography as the motion model, and using second-order approximation of the image function, one will get a global refinement algorithm with a faster convergence rate, i.e. the ESM algorithm proposed by (Benhimane and Malis 2004; Benhimane et al. 2005).

Firstly, the optimization equation is:

$$\begin{aligned} & \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \left[I(H(\mathbf{p})\cdot\mathbf{x}) - T(\mathbf{x}) \right]^2 \\ \Leftrightarrow \quad & \arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[I(H(\mathbf{p})H(\Delta\mathbf{p})\cdot\mathbf{x}) - T(\mathbf{x}) \right]^2, \end{aligned} \tag{3.7}$$

where $H(\mathbf{p})$ is a homography function that takes an 8 by 1 parameter vector $\mathbf{p}$ as input and outputs a 3 by 3 homography matrix, satisfying that $H(\mathbf{0})$ is an identity matrix, and $\mathbf{\Delta p}$ is the updating parameter of the initial guess on optimal parameter $\mathbf{p}$; also, $H(\mathbf{p})\cdot\mathbf{x}$ means mapping the 2D point $\mathbf{x}$ using the homography $H(\mathbf{p})$ by equation (3.1). Notice that in this equation $\mathbf{x}$ is not limited to any local region. Instead, $\mathbf{x}$ can be any pixel location in the template image T, i.e. the marker image in this framework, which is why this is dubbed as a global refinement. Then a second order Taylor expansion of the image function $I(H(\mathbf{p})H(\cdot)\cdot\mathbf{x})$ around updating parameter $\mathbf{\Delta p=0}$ is performed:

$$\arg\min_{\mathbf{\Delta p}}\sum_{\mathbf{x}}\left[I(H(\mathbf{p})\cdot\mathbf{x})-T(\mathbf{x})+J(\mathbf{0})\mathbf{\Delta p}+\frac{1}{2}\mathbf{\Delta p}^{\mathbf{T}}M(\mathbf{0})\mathbf{\Delta p}\right]^2, \tag{3.8}$$

where $J(\mathbf{0})$ is the 1 by 8 Jacobian matrix of the function $I(H(\mathbf{p})H(\cdot)\cdot\mathbf{x})$, and $M(\mathbf{0})$ its 8 by 8 Hessian matrix, all evaluated at $\mathbf{\Delta p=0}$ and location $\mathbf{x}$.

Next, let's consider the novelty of ESM. One can again apply a first-order approximation of the Jacobian matrix $J(\mathbf{\Delta p})=J(\mathbf{0})+\mathbf{\Delta p}^{\mathbf{T}}M(\mathbf{0})$ and substitute it into equation (3.8) to get:

$$\arg\min_{\mathbf{\Delta p}}\sum_{\mathbf{x}}\left[I(H(\mathbf{p})\cdot\mathbf{x})-T(\mathbf{x})+\frac{1}{2}\big(J(\mathbf{0})+J(\mathbf{\Delta p})\big)\mathbf{\Delta p}\right]^2. \tag{3.9}$$

Thus one gets a second-order approximation without calculating the Hessian matrix, whose computation is time-consuming. Let $J_{esm}(\mathbf{\Delta p})=\frac{1}{2}\big(J(\mathbf{0})+J(\mathbf{\Delta p})\big)$, then the optimization becomes:

$$\arg\min_{\mathbf{\Delta p}}\sum_{\mathbf{x}}\big[J_{esm}(\mathbf{\Delta p})\mathbf{\Delta p}+I(H(\mathbf{p})\cdot\mathbf{x})-T(\mathbf{x})\big]^2. \tag{3.10}$$

If one knows how to compute the $J_{esm}(\mathbf{\Delta p})$ without knowing **Δp**, the problem is again a standard least-square problem similar to equation (3.6).

So the only problem now is the calculation of $J_{esm}(\mathbf{\Delta p})$. Let's consider the first half of it. Applying the chain rule, one has:

$$\begin{aligned} J(\mathbf{0}) &= \nabla I(H(\mathbf{p})H(\mathbf{\Delta p})\cdot\mathbf{x})\,|_{\mathbf{\Delta p}=\mathbf{0}} \\ &= \nabla I(H(\mathbf{p}))\nabla(H\cdot\mathbf{x})\,|_{H=H(\mathbf{0})=I}\ \nabla H(\mathbf{p})\,|_{\mathbf{p}=\mathbf{\Delta p}=\mathbf{0}} \\ &= J_I J_W J_H, \end{aligned} \tag{3.11}$$

where J_I is the image gradient of the warped image $I(H(\mathbf{p}))$ at location **x**, J_W is the Jacobian of the homographic mapping function $W(\mathbf{x};\mathbf{p}) = H(\mathbf{p})\cdot\mathbf{x}$, and J_H depends on the choice of the parameterization of the homography. J_W and J_H can be pre-computed once and for all.

In order to prevent direct calculation of $J(\mathbf{\Delta p})$, since **Δp** is the variable to be optimized, Benhimane et al. chose to use a Lie Algebra method for parameterizing the homography (Benhimane and Malis 2004). Thanks to this method, the calculation of $J(\mathbf{\Delta p})$ can be circumvented by $J(\mathbf{\Delta p})=J_T\cdot J_W\cdot J_H$, where J_T is the image gradient of the template image T. This means:

$$J_{esm}(\mathbf{\Delta p}) = \frac{1}{2}(J_I + J_T)J_W J_H. \tag{3.12}$$

Since, in the above equations, the computation of J_I, J_T, J_W and J_H only requires the location **x** and the **p** as the current guess of homography parameters, without the need of knowing **Δp**, the previous problem is solved. In summary of the above derivation, one can get the ESM global refinement algorithm (see Algorithm 3–2).

Algorithm 3–2: ESM global refinement algorithm.

Initiate $\mathbf{p}=\mathbf{0}$.

Pre-compute $J_W J_H$ as in equation (3.11), and also image gradient of template image T.

Iterate:

1. Warp current image I with $H(\mathbf{p})$, result in the warped image $I(H(\mathbf{p}))$;
2. Compute image gradient of the warped image $I(H(\mathbf{p}))$;
3. Compute J_{esm} as in equation (3.12);
4. Compute error image $I(H(\mathbf{p})\cdot\mathbf{x})$-$T(\mathbf{x})$;
5. Solve the least-square problem in equation (3.10);
6. Update the homographic warp $H(\mathbf{p}) \leftarrow H(\mathbf{p})H(\Delta\mathbf{p}^*)$

Until $\left\|\Delta\mathbf{p}^*\right\| \leqslant \epsilon$.

3.5 Global Geometric and Appearance Constraints

The advantage of homography-from-detection methods lies in the fact that since they treat each image frame separately, as shown in Figure 3-3, estimation can be totally wrong at one particular frame, and the following frames won't be affected at all. However, the problem with methods such as SURF and FERNs is that, in order to speed up the time-consuming matching step in detection, lots of approximations are adapted. This makes the homography estimation very unstable, resulting in a very annoying jitter effect if adopted in AR applications, i.e. the augmented object appears to be shaking in the scene (Kamat and El-Tawil 2007). In the author's experiments, even if the camera is fixed, the estimated camera position and orientation could have very large variance.

In a different approach, homography-from-tracking methods, such as ESM and KLT, compare the current frame with the previous one in order to track the change. Although they benefit from

the relatively higher tracking speed that improves their frame rate, one critical problem of this group of methods is that every tracker suffers from the drifting effect, i.e. the updated position of the tracked point actually differs to some extent from its true new position, and will thus eventually fail. The drifting effect will lead to large error in homography estimation, since these drifting errors usually do not follow Gaussian distribution and shall be seen as systematic errors that are changing dynamically. Also, the greater the number of points failing to be tracked, the larger the variance that the estimated camera pose could have. Thus the augmented objects could be in a wrong position and shaking at the same time.

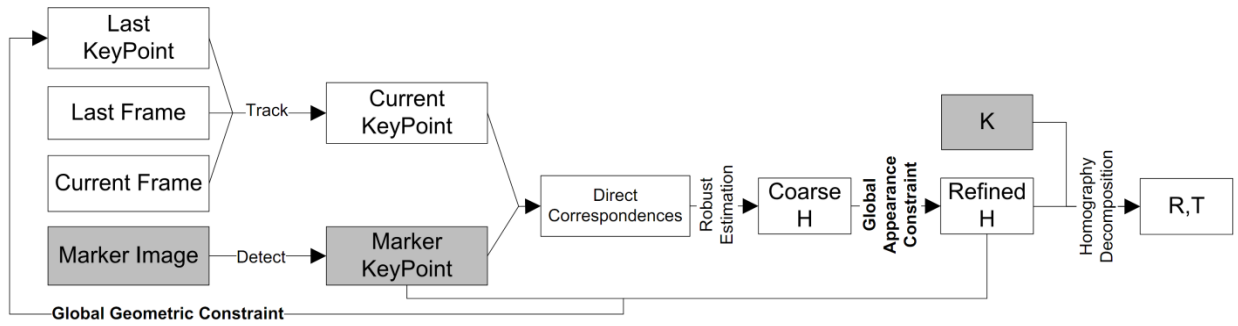


Figure 3-5: KEG algorithm framework.

The new framework (Figure 3-5) proposed in this chapter integrates the homography-from-detection and homography-from-tracking frameworks, utilizing their strong points and circumventing their short-comings. In general, the framework starts by the original homography-from-detection framework. Once the marker image is detected along with a rough estimation of the homography, it immediately moves into a coarse-to-fine framework. Only when the track is somehow lost will this procedure be repeated. Within the new framework, and firstly via the original tracking algorithm (KLT), a coarse homography could be found. Then, it would be refined by a global optimization algorithm (ESM). Finally the refined homography would be

used to correct the positions of the set of points to be tracked in the next frame, which is inspired by the following analysis of the cause of the drifting effect.

3.5.1 Drifting Effect Analysis

After analyzing the drifting effect in homography-from-tracking methods, it is actually found to be an error accumulation issue. During the tracking between every two consecutive frames, the error introduced by any tracking algorithm is accumulated. After a while, this accumulation could directly lead to the tracking of a wrong local target or even to the failure of the tracker. After realizing this, one pertinent question to ask is: is there any way to correct the error before the next tracking is actually performed? It was found that this is possible, which led to the design of this proposed framework. To gain more understanding of the cause of the drifting effect, the detailed error analysis is shown, as follows.

Firstly, the error source of any tracking algorithm, such as KLT, can be seen as composed by the following terms:

$$\hat{\mathbf{x}}_{new} = \mathbf{x}_{old} + \Delta\mathbf{x} + \varepsilon_d + \varepsilon_g \tag{3.13}$$

where $\hat{\mathbf{x}}_{new}$ is the updated position estimated by KLT, $\mathbf{x}_{old}$ is the true original position, $\Delta\mathbf{x} = \mathbf{x}_{new} - \mathbf{x}_{old}$ is the true displacement, ε_d is the systematic drifting error, and ε_g is the rest of the error, which is assumed to follow some Gaussian distribution. Here, $\Delta\mathbf{x}$ is caused by physical movement between the camera and the scene, ε_g is mainly caused by camera CCD sensor noise, and ε_d is usually caused by the tracking algorithm and other complicated reasons, such as the fact that KLT will be affected a lot when the camera is moving too fast, which leads to motion blur and thus violates KLT's underlying assumptions.

The second step of homography-from-tracking methods applies robust estimation algorithms (e.g. RANSAC) to estimate $\mathbf{H}_{coarse}$ from the array of tracked points $\{\hat{\mathbf{x}}_{new}\}$ and their corresponding points on marker image. However, even though RANSAC can eliminate a lot of outliers if the absolute value of error $\left|\varepsilon_d + \varepsilon_g\right|$ exceeds some threshold, and further, can eliminate the Gaussian error ε_g by a final least-square estimation on the outlier-free subset of correspondent points, there still remains a part of systematic drifting error ε_d not handled and thus propagated into $\mathbf{H}_{coarse}$. In the homography-from-tracking framework, neither ε_d nor ε_g are corrected during the update step, so these errors are accumulated, which can cause a large drift even after a few frames of tracking.

3.5.2 Error Correction by Global Constraints

One natural way to reduce the effect of ε_d is to apply the **global appearance constraint**, as shown in equation (3.14),

$$\begin{aligned} \Delta\mathbf{p}^* &= \arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[I(\mathbf{H}_{coarse} H(\Delta\mathbf{p}) \cdot \mathbf{x}) - T(\mathbf{x}) \right]^2, \\ \mathbf{H}_{refined} &= \mathbf{H}_{coarse} H(\Delta\mathbf{p}^*), \end{aligned} \tag{3.14}$$

which essentially means that the original marker image will look the same as the image rectified from the current frame by estimated homography **H**. Before this step, all of the information used by the KLT tracker is local, while ε_d is systematic, therefore a global optimization based on the whole marker image, i.e. the global appearance constraint represented by equation (3.14), will theoretically eliminate all the systematic error and $\mathbf{H}_{coarse}$ can serve as a good optimization starting point.

After the drifting error is eliminated when estimating the refined homography $\mathbf{H}_{refined}$, one can easily correct tracking errors and update keypoint positions to be filled into the next tracking iteration by the homography mapping:

$$\bar{\mathbf{x}}_{new} = \mathbf{H}_{refined}\mathbf{x}_{ref} \tag{3.15}$$

where $\mathbf{x}_{ref}$ are keypoint positions on the original marker image; this is named as applying the **global geometric constraint** (for it relies on the prior knowledge that all keypoints lie in the same plane). Since the estimated $\mathbf{H}_{refined}$ is already theoretically error-free, updating using the above equation (3.15) instead of equation (3.13) prevents tracking error from propagating into the tracking of the next frame, and thus increases the tracking stability.

Besides the improvement in accuracy, this algorithm also enjoys an increase in tracking speed. Because of the global refinement step, the local tracking algorithm such as KLT needs not be very accurate by reducing the number of iterations of KLT algorithms that result in larger error $\left|\varepsilon_d + \varepsilon_g\right|$. Since the direct result of KLT is just a coarse homography serving as an ESM optimization starting point, a certain amount of error can be tolerated and will be theoretically eliminated after global refinement (ESM). Similarly, since the time complexity of a local tracking algorithm such as KLT is usually positively correlated to the number of points to be tracked, the number of keypoints to be tracked can be decreased.

This method is denoted as the KEG (KLT Enhanced by Global constraints) tracker, and the complete algorithm framework is described in Algorithm 3–3. It's worth noting that KLT, ESM, and RANSAC, as well as the initial detection method (AprilTag/SURF), are replaceable components in this approach. This makes the method very flexible and easy to be extended by

new algorithms (as long as they serve the same purpose). Detailed comparisons in section 3.6 show that, even though the new framework involves more steps, its performance in accuracy, stability, and speed is increased as compared to the state-of-the-art algorithms.

Algorithm 3–3: KEG algorithm.

---

1. Detect N keypoints $\{\mathbf{x}_{ref}\}$ on marker image T;
2. Apply Fiducial Marker method (AprilTag) or Homography-from-detection algorithm (SURF), try to find the marker image and its corresponding homography $\mathbf{H}_{refined}$. If found, go to step 6; otherwise, re-do step 2;
3. Take a new incoming frame I_{new}, the last frame I_{old}, and the old keypoint positions $\{\mathbf{x}_{old}\}$, perform local tracking (KLT) and output new keypoint position $\{\mathbf{x}_{new}\}$;
4. Perform robust estimation (RANSAC) on correspondent keypoint array $\{\mathbf{x}_{ref}\}$ and $\{\mathbf{x}_{new}\}$ and output $\mathbf{H}_{coarse}$;
5. Apply global appearance constraint by equation (3.14) and output $\mathbf{H}_{refined}$;
6. Validate $\mathbf{H}_{refined}$ by similarity (zero-mean normalized cross-correlation between T and rectified I_{new} by $\mathbf{H}_{refined}$) threshold. If valid, $\mathbf{H}_{refined}$ can be output for homography decomposition by equation (3.4); otherwise, meaning loss-of-track, go to step 2;
7. Update keypoints position $\{\mathbf{x}_{new}\}$ using global geometric constraint by equation (3.15);
8. Replace I_{old} with I_{new}. Replace $\{\mathbf{x}_{old}\}$ with $\{\mathbf{x}_{new}\}$. Go to step 3 until no new incoming frames.

3.6 Experimental Results

In order to validate this method and compare it to state-of-the-art algorithms, several experiments on both real-world and synthesized video sequences (in which the ground-truth of the camera pose is known) were conducted. Experiments were all conducted on a desktop computer with an eight-core 2.8 GHz Intel Core i7 CPU with 6 GB memory. Also, all of the video sequences have a frame size of 640 by 480 pixels, which is the commonly adopted size of commercial webcams.

In all of the test cases to be shown in the following, for the purpose of showing the necessity of the three core components in KEG—local tracker (K-step), global refinement (E-step), and error correction (G-step) —and proving its superiority to state-of-the-art methods, 7 different algorithms were tested over those cases:

1. KEG with AprilTag as initialization method (referred to as A+KEG).
2. No global appearance constraints applied; others are the same as 1 (A+K G).
3. No global geometric constraints applied; others are the same as 1 (A+KE).
4. No global constraints applied, representing homography-from-tracking method (A+K).
5. AprilTag, representing fiducial marker-based method (A).
6. AprilTag with global appearance constraints applied (A+ E).
7. FERNs, representing homography-from-detection method (FERNs).

For the homography-from-detection component, the C++ implementation of AprilTag (https://github.com/simbaforrest/cv2cg) was used, which originates from the java implementation by April Lab (http://april.eecs.umich.edu/wiki/index.php/AprilTags) at the University of Michigan (Olson 2011). The KEG algorithm is open-source, and the C++ code can be found at https://github.com/simbaforrest/cv2cg. For comparison with state-of-the-art homography-from-detection methods, the well-known and widely used Open Source Computer Vision (OpenCV) library (http://opencv.org/) implementation of the FERNs method was adopted, which also provides the implementation of KLT. For the ESM method, the binary library provided by INRIA Sophia-Antipolis at http://esm.gforge.inria.fr/ESMdownloads.html was used.

Different performance metrics are proposed so as to have a comprehensive understanding of the performance of these algorithms:

- **Duration:** The time to process each frame, reflecting the speed of the algorithm. This metric is crucial for real-time applications.
- **NCC:** The zero-mean normalized cross-correlation between the original marker image I_1 and the rectified image I_2 by $\mathbf{H}_{refined}$, which can be calculated by:

$$NCC(I_1, I_2) = \frac{1}{n} \sum_{\mathbf{x}} \frac{[I_1(\mathbf{x}) - m_1][I_2(\mathbf{x}) - m_2]}{\sigma_1 \sigma_2} \tag{3.16}$$

  where n is the total number of pixels of image I_1 or I_2, and m_i and σ_i are the mean value and standard deviation of intensity of image I_i. Obviously, if I_1 and I_2 are exactly the same, their NCC index should be one, and the larger their difference, the smaller the NCC index. This means NCC is a good similarity index (Ladikos et al. 2007). This index is also used in KEG to determine whether it loses track or not by a simple threshold of 0.5; if at any frame the NCC index is smaller than 0.5, it is regarded as a loss-of-track frame.
- **LOT:** The total number of loss-of-track frames. This metric represents a registration algorithm's stability to some extent.
- **T-RMS/R-RMS:** The root mean square error between ground truth and estimated camera position/orientation. This metric represents the absolute accuracy of a registration algorithm, and is calculated by:

$$T-RMS=\sqrt{\frac{1}{k}\sum_{i=1}^{k}\left\|\mathbf{T}_i-\hat{\mathbf{T}}_i\right\|^2},$$
$$R-RMS=\sqrt{\frac{1}{k}\sum_{i=1}^{k}\left\|e_i-\hat{e}_i\right\|^2}, \tag{3.17}$$

where k is the total number of frames of a test case, $\mathbf{T}_i$ and $\hat{\mathbf{T}}_i$ are the i-th frame's ground truth and estimated position vector of dimension 3 by 1, and e_i and $\hat{e}_i$ are the i-th frame's ground truth and estimated Euler angle vector of dimension 3 by 1, respectively. Since these two indices require ground truth data, they are only examined for synthesized test cases.

- **UOT:** This new index is also proposed for estimating the extent of jitter effect of a registration algorithm, i.e. the unsmoothness-of-tracking, taking advantage of the NCC index by

$$d_i=NCC_{i+1}-NCC_i,\ \forall i=1,2,\cdots,k-1,$$
$$UOT=\sqrt{\frac{1}{k-1}\sum_{i=1}^{k-1}(d_i-\mu)^2},\ \mu=\frac{1}{k-1}\sum_{i=1}^{k-1}d_i, \tag{3.18}$$

where NCC_i is the NCC index of the i-th frame, which essentially means that UOT index is the standard deviation of the difference between consecutive NCC indices. In MATLAB, this can be simply calculated by "*std(diff(ncc))*." A stable registration algorithm should give a UOT index value as small as possible.

(a) UM Logo. (b) The Marker Image. (c) Background Image of Synthesized Test Case

Figure 3-6: Marker image composition.

In all of the test cases, the marker image is composed of a 16 bits AprilTag of ID equal to zero (Figure 3-2c) and a natural image, the logo of University of Michigan (Figure 3-6a), that is rich in features (Figure 6b). Note that using AprilTag here does not mean that this method is fiducial marker based method. As explained in step 2 of Algorithm 3–3, applying either fiducial marker based method or Homography-from-detection based method (SURF) could serve as a starting point of this method in the first video frame. However, even though AprilTag is not a necessary component here, it is added to support multiple markers, which is very useful in many applications to be explained in section 3.7.

3.6.1 Synthesized Test Cases

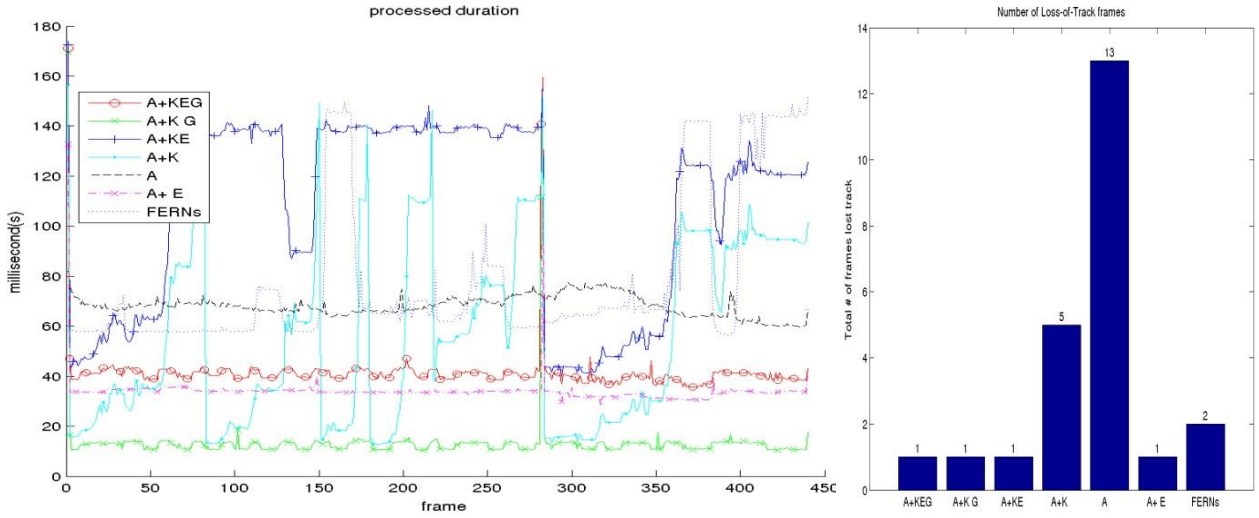
Three test cases were synthesized in OpenGL, using the marker image and a static real-world image as background (Figure 3-2c). The first test case, S-1, simulates a random camera movement of both position and orientation change with 440 frames. The second test case, S-2, simulates 421 frames of the situation in which the camera moves around the marker image with a fixed distance. The last test case, S-3, simulates 304 frames of the situation in which the camera first moves close to the marker image and then far away, and the camera plane is parallel to the

marker image plane. The different performance measures of different algorithms are shown in Figure 3-7, Figure 3-8 and Figure 3-9.

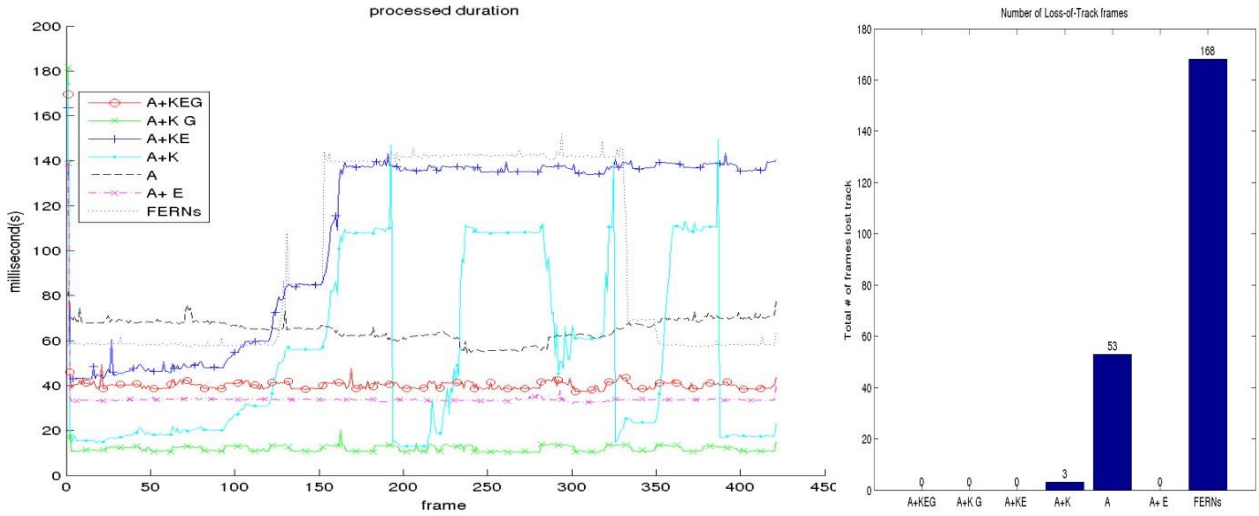
From the left column of Figure 3-7 one can see that the average processing time of A+KEG is about 40 milliseconds, which is even faster than AprilTag. While other algorithms have very unstable processing time and are mostly slower than A+KEG, the only exception is A+KG, which is as expected since it has no global refinement step. These curves prove that the KEG method does fit for real-time applications and that it can process images at 20 frames-per-second or faster.

From the left column of Figure 3-8 one can find that, even though sometimes AprilTag might have a slightly higher NCC value, in most cases, the NCC curve of A+KEG is the upper bound for the other algorithms, especially performing better than the state-of-the-art algorithm, FERNs.

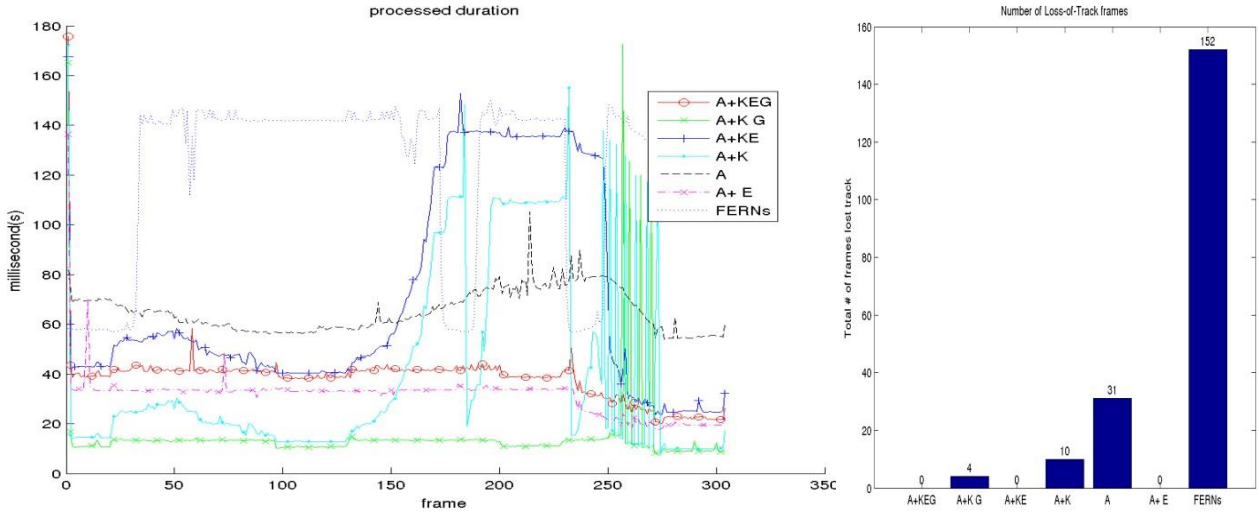
From the right column of Figure 3-7 and Figure 3-8, one can figure out that the A+KEG method has fewer loss-of-track frames, showing its ability to track longer, and lower UOT index, showing its smoothness in tracking—an important feature if applied in AR. Also A+KEG is more accurate, by giving less T-RMS/R-RMS errors in Figure 3-9. Notice that here it is assumed the radius of the synthesized marker is 20 cm (which was the real size when it was printed out on an A4 sheet of paper in the real-world test cases). In this configuration, the KEG tracker's maximum working distance can be as far as 3 meters, and its maximum working Euler angle can be about 85 degree offset from the marker image's normal direction. If an even larger working distance is desirable, a higher resolution camera and bigger marker can be adopted.



(a) Test case S-1.

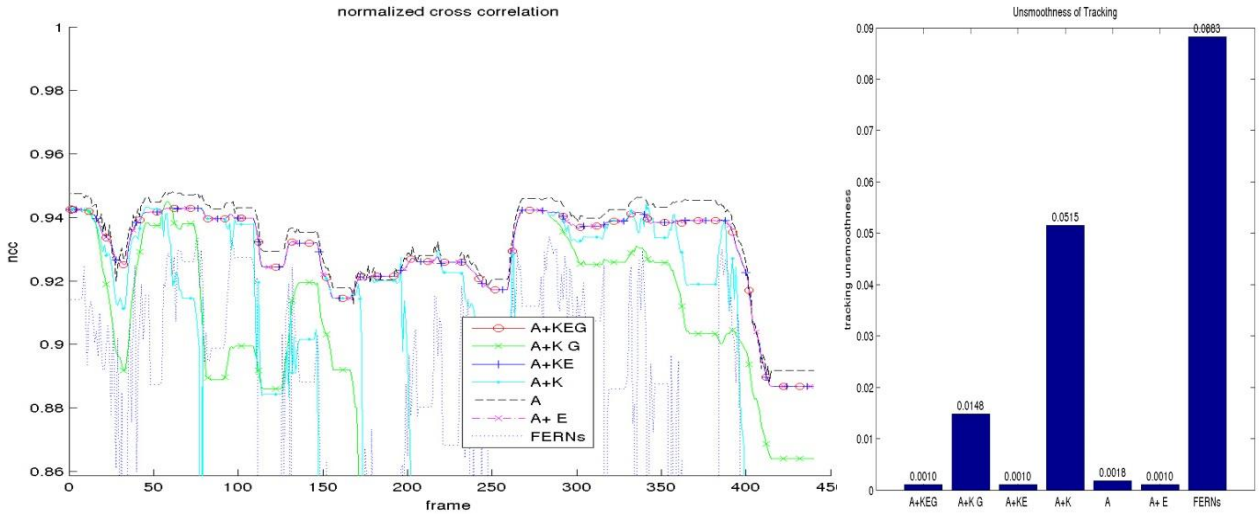


(b) Test case S-2.

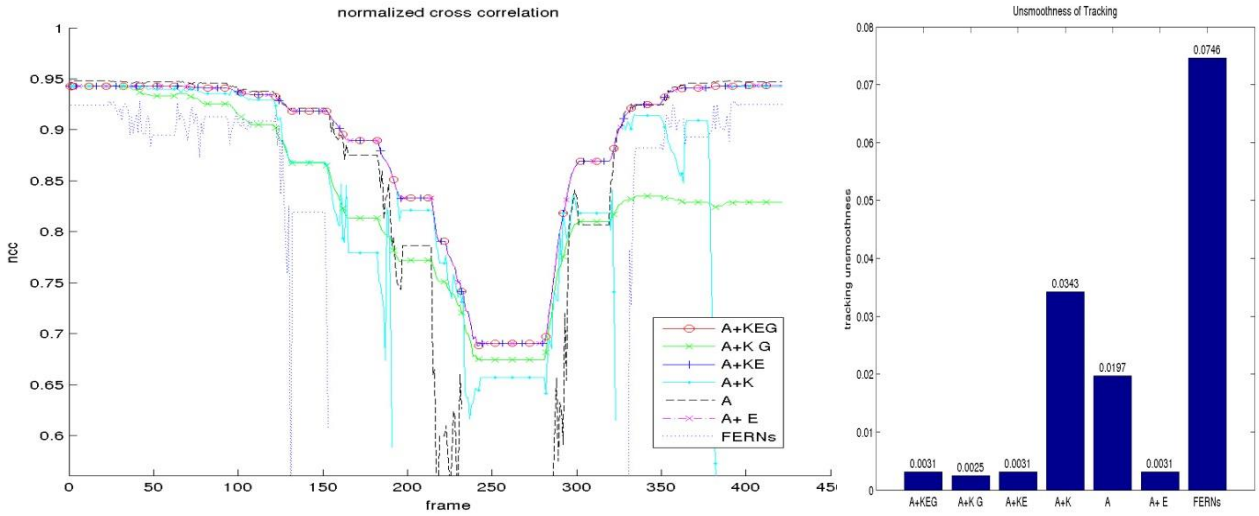


(c) Test case S-3.

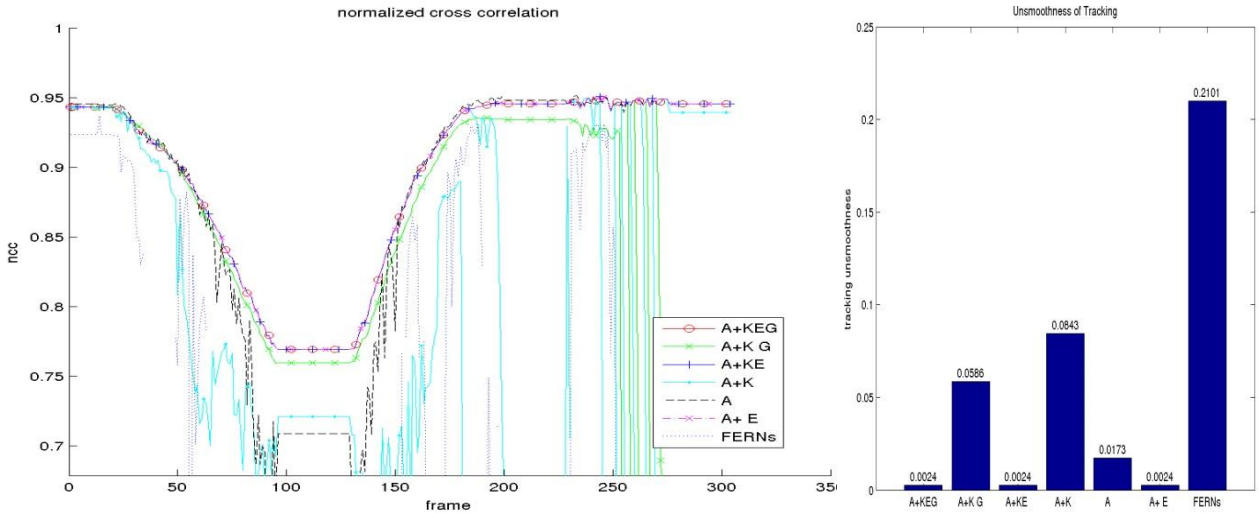
Figure 3-7: Duration curves (left) and LOT bars (right) for synthesized test cases.



(a) Test case S-1.

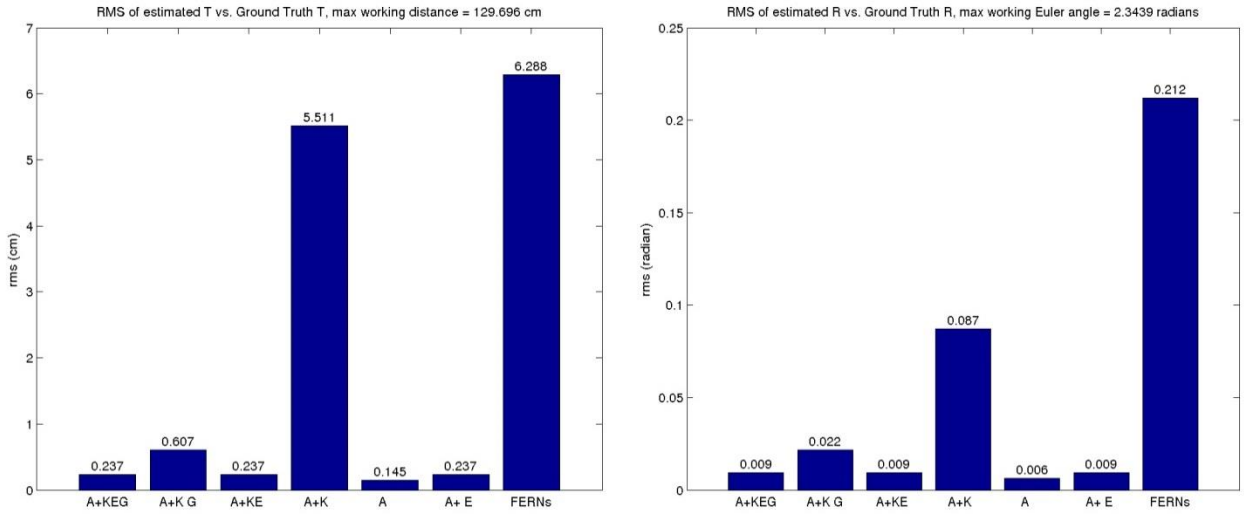


(b) Test case S-2.

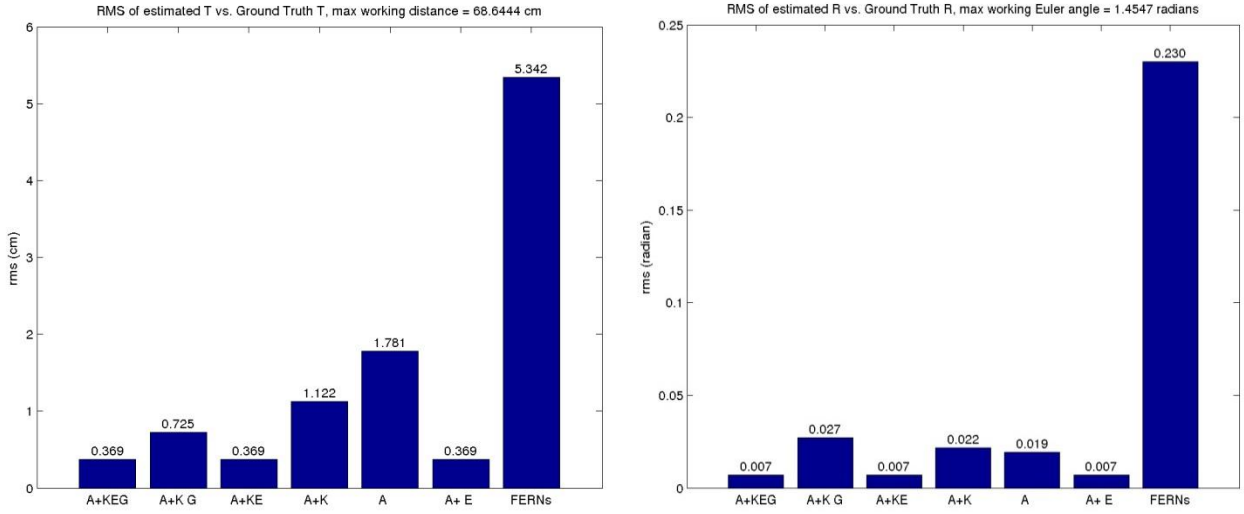


(c) Test case S-3.

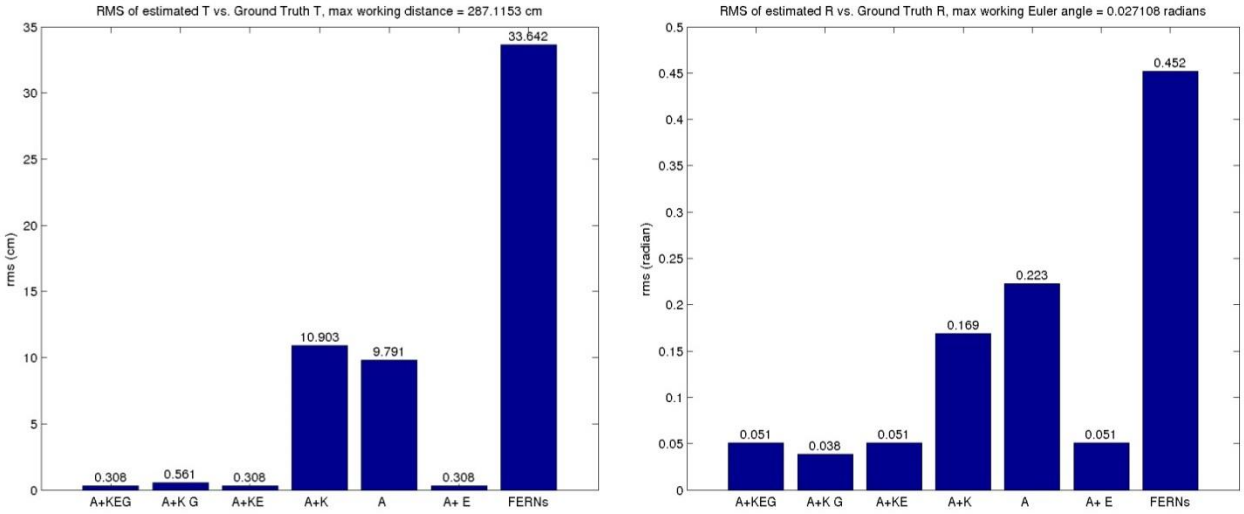
Figure 3-8: NCC (left) curves and UOT (right) bars for synthesized test cases.



(a) Test case S-1.

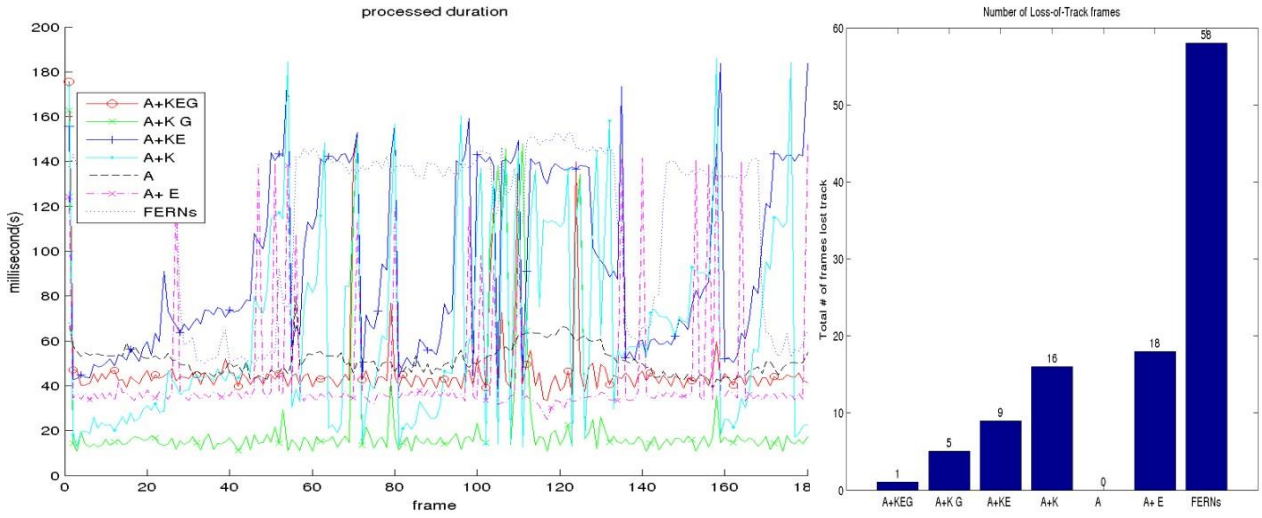


(b) Test case S-2.

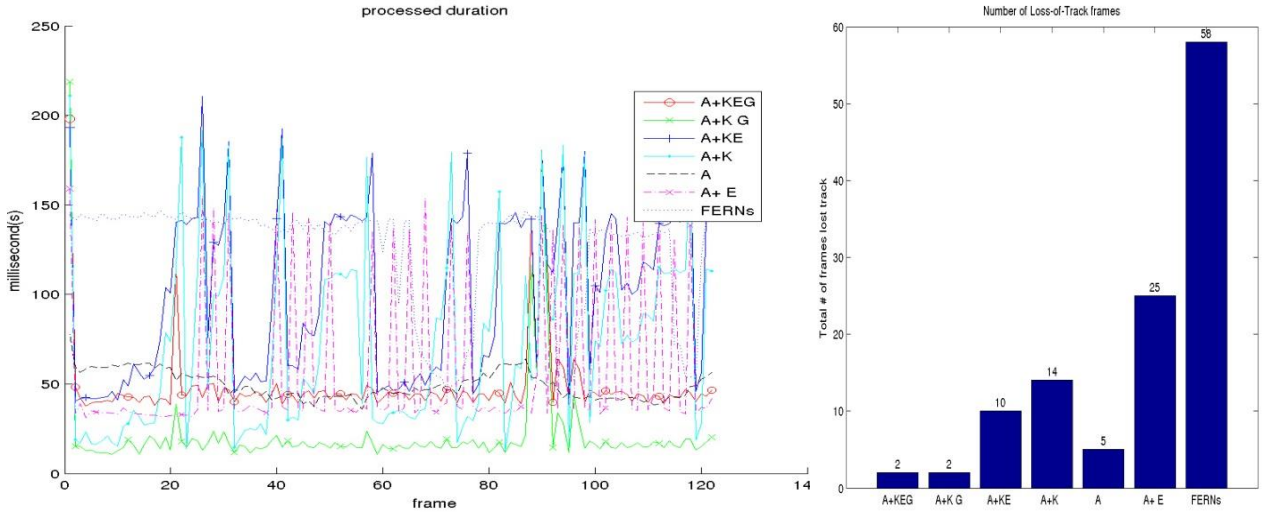


(c) Test case S-3.

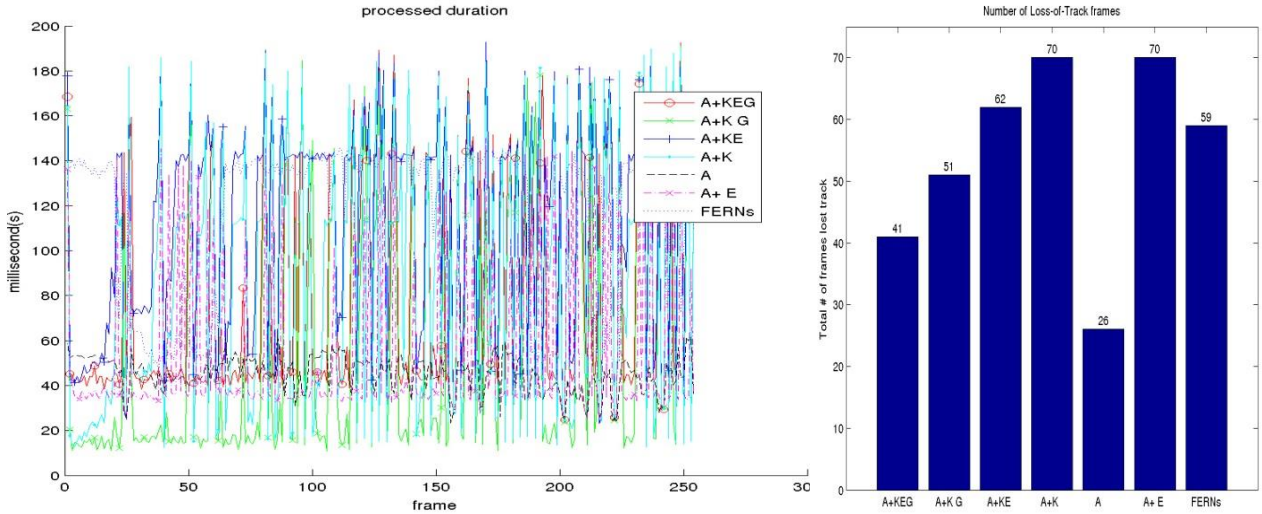
Figure 3-9: T-RMS (left) and R-RMS (right) bars for synthesized test cases.



(a) Test case R-1.

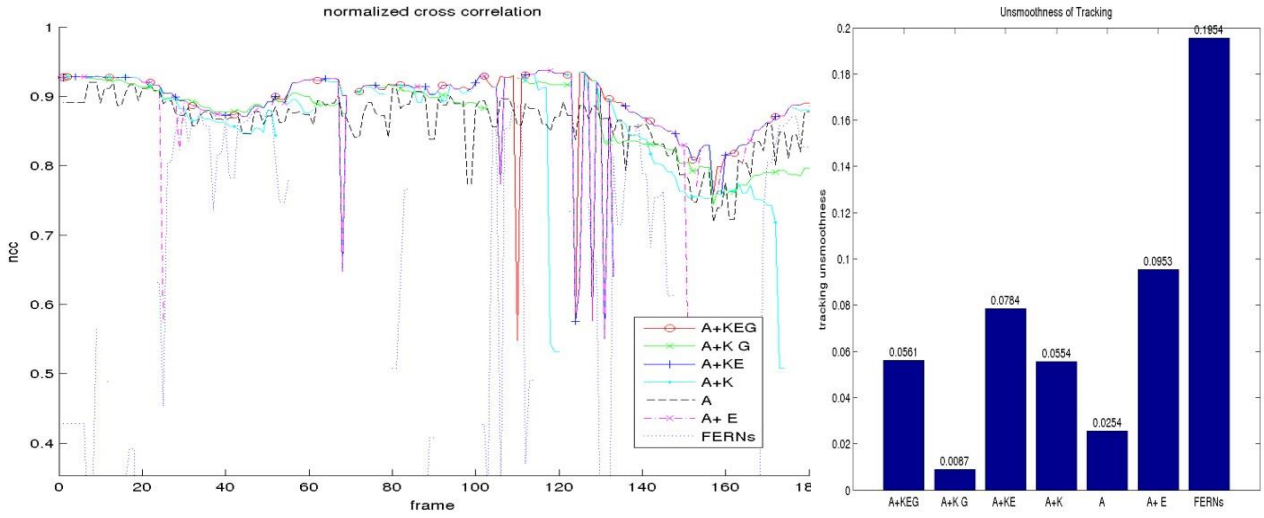


(b) Test case R-2.

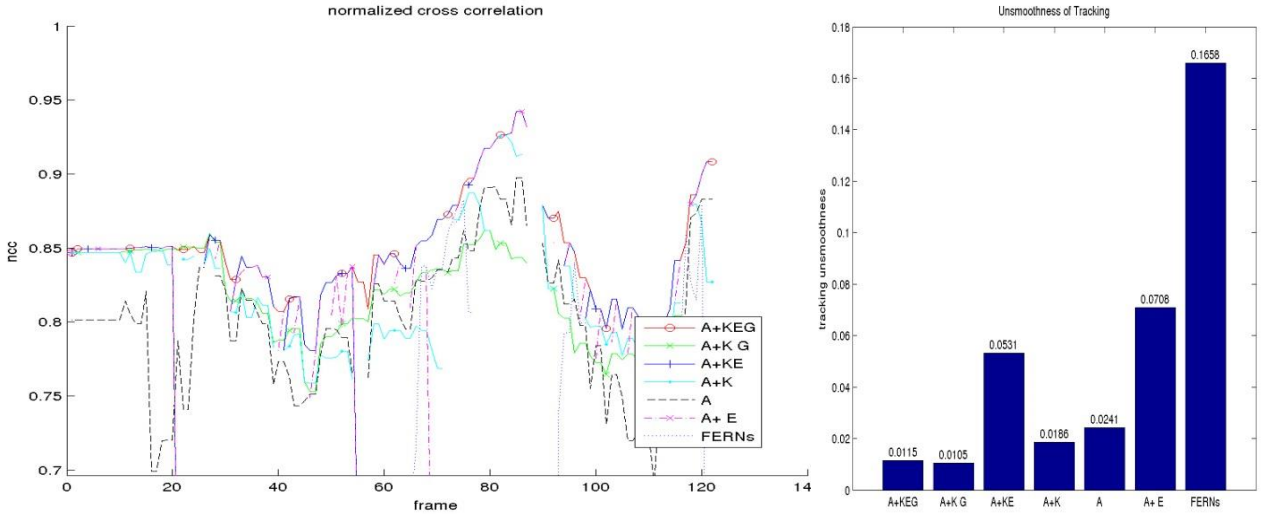


(c) Test case R-3.

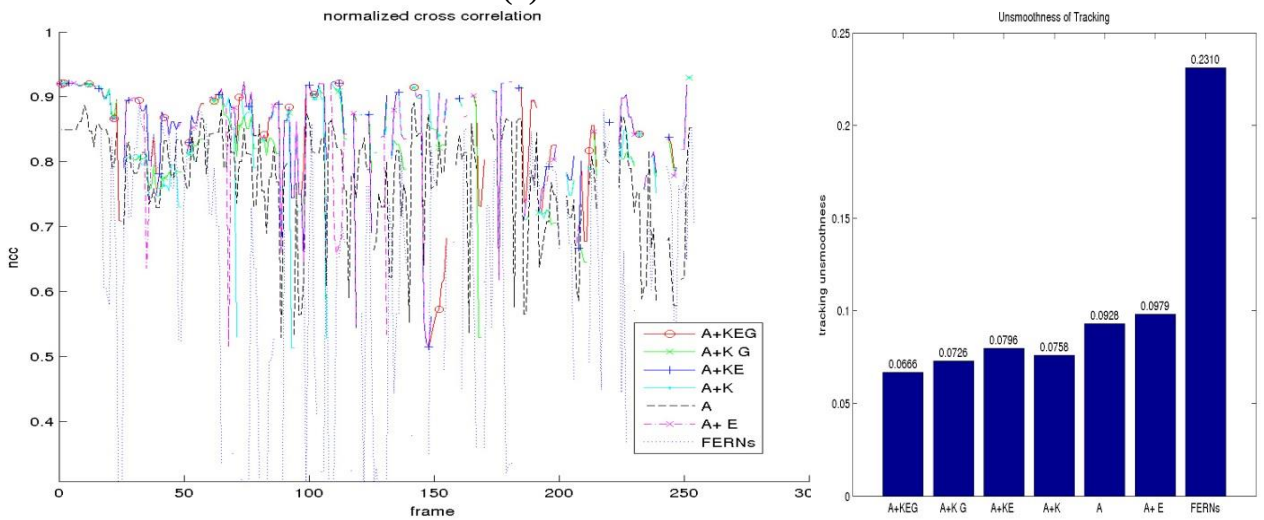
Figure 3-10: Duration curves (left) and LOT bars (right) for real-world test cases.



(a) Test case R-1.



(b) Test case R-2.



(c) Test case R-3.

Figure 3-11: NCC (left) curves and UOT (right) bars for real-world test cases.

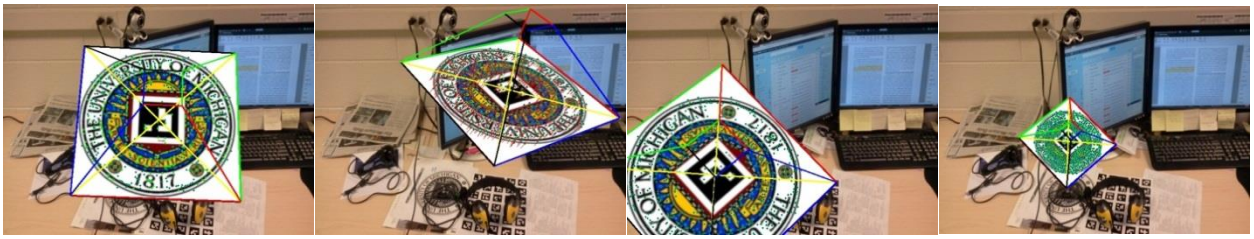
3.6.2 Real-world Test Cases

Three real-world video sequences were also recorded for a test. The first sequences (R-1) have 180 frames, the second (R-2) 122 frames, and the third (R-3) 254 frames. The R-3 test purposefully has a lot of shaking in the camera movements; this is to test tracking performance in a very challenging situation. Note that in real-world test cases, the ground truth of the camera pose is unknown, so T-RMS and R-RMS are not examined here. Another difference between real-world and synthesized test cases is that noises introduced by the webcam sensor in real-world test cases will affect the accuracy of tracking.

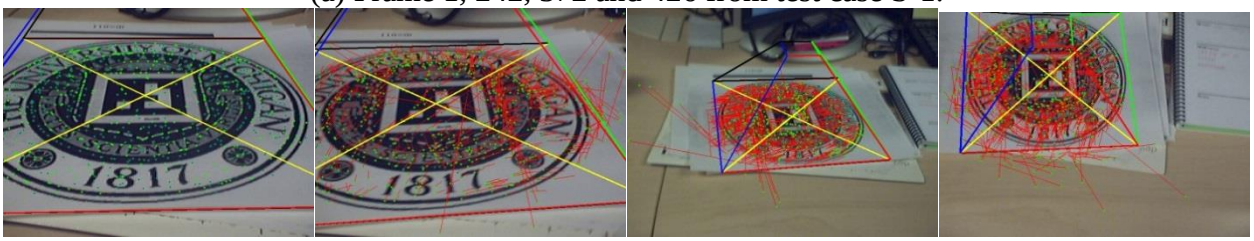
From Figure 3-11 one can see a similar performance analysis as in the synthesized test cases, which firmly proves the claim that the KEG method outperforms the state-of-the-art methods FERNs and AprilTag in speed, accuracy and stability. It's also worth noting that by comparing the algorithm configuration between A+KEG, A+KG, A+KE, and A+K, one can conclude that the three core steps of KEG are all crucial, and that the KEG method cannot achieve the same performance while missing any one of the three components. Another interesting observation is that the FERNs algorithm in real-world test cases is affected a lot by the noisy measurements. Comparing the left column of Figure 3-7 and Figure 3-10, it can be seen that the processing time per frame increased from 60 to 150 milliseconds on average, which drops its frame rate to only about 7 frames-per-second.

Figure 3-12 shows 4 representative frames under different challenging visual conditions such as large rotation or scale change (picked from test case S-1), or partial occlusion (picked from test case R-2), using the A+KEG method. The color pyramid represents the registration result, **R** and **T** of the camera relative to the marker image (since it is a 3D pyramid, without correct **R** and **T**,

it is impossible to be rendered correctly). The red trails show the position update from $\mathbf{x}_{old}$ to $\bar{\mathbf{x}}_{new}$.



(a) Frame 1, 242, 372 and 426 from test case S-1.



(b) Frame 1, 20, 38 and 92 from test case R-2.

Figure 3-12: Visualization of A+KEG registration results of some representative frames.

3.7 Applications

As noted in the Introduction, this algorithm has several potential applications in many different areas. One example application implemented here is context-aware computing. Indoor context-aware computing has been studied in AEC for its ability to speed up information delivery in many aspects, including construction site inspection/monitoring and facility management (Aziz et al. 2005; Behzadan and Kamat 2009; Khoury and Kamat 2009; May et al. 2005). Prior approaches for indoor ubiquitous tracking utilize an inertial measurement device, which suffers from its drifting effect. By (Akula et al. 2011), a context-aware computing system integrated with both GPS and inertial measurement device is developed, requiring human intelligence to recognize certain predefined locations to manually correct the drifting error caused by the inertial measurement device.

(a) A+KEG in context-aware computing.

(b) A+KEG in desktop AR.

Figure 3-13: Two example applications of the KEG tracker.

In this application, manual error correction was naturally replaced by automated correction using the A+KEG method, as shown in Figure 3-13a[5]. The green text shows that the algorithm successfully recognizes different locations by composing a natural photo (UM logo in this case) with different AprilTags. Once the inspector is within the effective range of the KEG marker image, the marker is automatically detected and then the inspector's pose relative to the marker is continuously estimated. Similar to (Akula et al. 2011), both the location and orientation of these predefined markers in the global coordinate system are known and stored in a database. Therefore the inspector's pose in the global coordinate system can be determined, as well. Benefiting from the KEG tracker, this application can provide automatic regional drifting error correction instead of manual point correction. This application can thus further facilitate information delivery on construction sites or in indoor building environments.

Another interesting application is to apply this algorithm in tabletop augmented reality. Figure 3-13b[6] shows a desktop environment AR showcase of a 3D building design. Since the KEG

[5] See video in http://www.youtube.com/watch?v=Cnvr3l104wM
[6] See video in http://www.youtube.com/watch?v=8Y8Mlh7jhsY

tracker has the ability to quickly detect and accurately maintain the tracking of a marker image without requiring that the marker image be fully in sight (as required by ARToolkit), it can easily be adapted into tabletop collaborative AR applications (Dong and Kamat, 2011) to support better interactive design demonstration or visual simulation for construction planning.

3.8 Conclusions

After studying the two different types of natural marker-based registration algorithms and analyzing the cause of the drifting and jitter effects in both homography-from-tracking and homography-from-detection methods, a new natural marker-based registration algorithm framework, KEG tracker, is proposed, combining the advantages of those two, and circumventing their shortcomings. In theoretical analysis, it was found out that the drifting effect is an error that occurs because of an accumulation problem. This problem was solved by applying two global constraints: a geometric and appearance constraint.

The experiments on both synthesized and real-world test cases prove that the KEG method is fast enough for real-time applications (about 40 milliseconds for processing one 640 by 480 image), and also more accurate and stable than state-of-the-art algorithms such as FERNs and AprilTag. When the radius of the KEG marker is 20 cm printed out on a sheet of A4 paper, and the webcam provides an image resolution of 640 by 480 pixels, the KEG tracker's maximum working distance can be as far as 3 meters, and its maximum working Euler angle can be about 85 degrees offset from marker image's normal direction. If a larger working distance is desirable, a higher resolution camera and bigger marker can be deployed.

Potential applications of the new tracker was also explored, such as context-aware computing, for replacing manually drifting error correction, and augmented reality for tabletop 3D visual simulation.

In the future, one direction for further research is applying more object recognition techniques so that, without the need of composing a fiducial marker (AprilTag), the method could more naturally support multiple marker recognition and tracking. Extending this method to a 3D environment, such that no planar structure assumption is needed, could also be a very interesting research direction. In addition, specific AEC applications of the tracker can be explored, such as pose estimation for construction equipment.

Chapter 4
Camera Marker Network

"In nature we never see anything isolated, but everything in connection with something else which is before it, beside it, under it and over it."—Johann Wolfgang von Goethe

4.1 Introduction and Previous Work

In Chapter 2 and Chapter 3, two algorithms about scene understanding and pose estimation using a single camera (either a depth camera or an ordinary RGB camera) were described in detail. However they share a common drawback, which is the relatively small working range of a single camera, due to the narrow field of view (FOV) of normal cameras with non-wide-angle lenses. For example, the popular RGBD camera, Microsoft Kinect, has a vertical FOV of 43° and horizontal FOV of 57° (Microsoft 2015). An ordinary webcam usually has a horizontal FOV between 50° and 90°. Additionally, due to the limitation in camera spatial resolution, objects or features beyond a certain distance from a camera become less significant in terms of depth and pose estimation. For instance, Microsoft Kinect's depth sensor range is minimum 800mm and maximum 4000mm (Microsoft 2012).

Such a drawback limits the sphere of application of those algorithms. For example, the KEG tracking algorithm and other marker-based pose estimation algorithms can only be employed in desktop range applications such as table-top AR (Dong et al. 2013). To increase the working range of those algorithms, as illustrated in Figure 4-1, there are three types of solutions addressing the problem from either spatial or temporal perspective:

1. Using more than one camera;
2. Moving the single camera while maintaining estimations of that camera's poses;
3. Combining the two options above, i.e., using more than one camera and moving some/all of the cameras while tracking the cameras' poses.

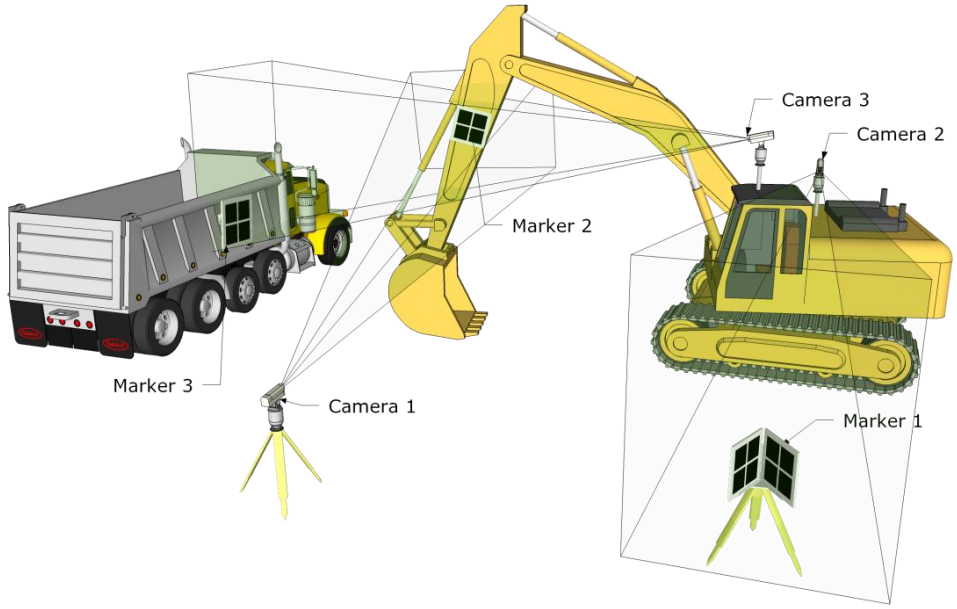


Figure 4-1: Illustration of camera marker networks applied on construction sites.

4.1.1 Multiple Cameras Solution

The solution 1, i.e., multiple cameras solution, approaches the problem from a spatial perspective, i.e., using multiple cameras' FOV to cover the desired working range. There are a few possible implementations. First is to simply form a camera cluster by rigidly mounting two or more cameras at nearby positions but pointing outwards to different directions. In this way, multiple ordinary cameras' narrow FOV is composed into the camera cluster's wider FOV. Different from directly using cameras with wide-angle lens (e.g., fish-eye or catadioptric panoramic cameras), this implementation preserves the same projection model as ordinary cameras and many marker-based pose estimation algorithms including the KEG tracking

algorithm can be directly applied without any modification. Point Grey's Ladybug series cameras (Point Grey 2015) are commercialized examples of this implementation.

Another possible implementation is more de-centralized, i.e., to form a camera network by rigidly mounting multiple cameras at different positions and pointing towards different directions, as long as the composite wider FOV of this camera network covers the desired range. Surveillance cameras on construction sites are examples of this implementation.

An important requirement of this type of solutions is the synchronization between all cameras. This means that for either pose estimation or scene understanding, the images from all cameras should be taken at the same time in order to make sound analysis and accurate estimation. This is outside the scope of this research, but can be achieved in a relatively straightforward manner.

Another requirement is that all cameras' poses should be calibrated in a same coordinate reference frame in advance of system operation. Only after this can poses that are independently estimated or 3D scene point clouds that are partially reconstructed in different views be transformed into a consistent reference frame for further analysis. Since a camera coordinate frame often cannot be directly surveyed physically, with a set of static markers, this requirement can be satisfied using the camera marker networks technique proposed in this research. This camera marker networks technique is similar to previously proposed spatial relationship patterns (Pustka et al. 2006) in the sense that both use graphs to abstract spatial relationships between objects. However, as explained below, there are several differences between the two techniques.

4.1.2 Multiple Views Solution

A potential disadvantage of the multiple cameras solution is that the increased number of cameras could increase the cost, the energy consumption and the vulnerability of the solution.

Another drawback is the relative non-flexibility due to the fixed camera setup. To avoid such weakness, the multiple views solution approaches the FOV problem from a temporal perspective by allowing the single camera to be dynamically moving and capturing multiple views for pose estimation and scene understanding. The previously introduced SfM (Snavely et al. 2006) and SLAM (Klein and Murray 2007) technologies all belong to this type of solutions.

Yet the dynamically moving camera introduces a different important requirement for this type of solutions, which is to estimate this moving camera's pose at any time when a new view is recorded. This is done in similar ways of the marker-based pose estimation methods explained in Chapter 3. Through identification of same physical elements' corresponding images (e.g., points, lines, planes, objects, etc.) across different views, the poses of those views can be estimated relative to each other. However the assumption for this correspondence identification process to be possible and reliable is that the target scene is rich in locally distinguishable features that can be captured by the camera. As mentioned in Chapter 1, many industrial environments, such as indoor construction sites before finishing, often do not satisfy such assumption. They (e.g., walls, floors, ceilings) are frequently featureless or texture-less, or with repeated features.

To enable a reliable and rapidly reconfigurable multiple views solution in industrial environments is another objective of the camera marker networks technique proposed in this research. With markers attached in these featureless target environments, the feature correspondence in computer vision and SfM, or data association in SLAM, can be more reliable. In addition, the cost to manufacture and install multiple markers in such environments is much less than that to install cameras and other active or passive sensors, especially considering that markers consume no energy.

4.1.3 Multiple Cameras and Views Solution

The third solution is a hybrid one of the previous two. For example in Figure 4-1, camera 2 and 3 forms a moving camera cluster, while camera 1 is a static surveillance camera onsite. Naturally this solution also inherits requirements of the previous two. For a dynamically moving camera cluster with relatively static cameras, markers will be needed to estimate the motion of the whole cluster. Markers are also necessary to calibrate the relative static poses of cameras in such clusters, or the poses of static cameras observing target environments.

In fact, both multiple cameras, multiple views and the hybrid solutions, can be abstracted in a unified framework, which is termed as the camera marker networks, explained in details in the following sections. Section 4.2 explains its methodology, including its definition, abstraction and mathematical solution. Section 4.3 performs the uncertainty analysis to such networks and lists a series of observations. Section 4.4 demonstrates several important experiments proving the networks effectiveness in real world. Finally conclusions are drawn.

4.2 Methodology

A **camera marker network** is defined as *an observation system containing multiple cameras or markers for estimating poses of objects embedded in this system*. The example in Figure 4-1 shows a camera marker network with three cameras and three markers. The objects of interests in this case are the excavator's base, its stick, and a haul truck.

4.2.1 Graph Abstraction

A camera marker network as defined above can be abstractly represented as a graph with three types of nodes and two types of edges. A *node* denotes the pose of an object (i.e. the local coordinate frame of that object) in the world coordinate frame, which can be a camera (or a

view), a marker, or the world coordinate frame. An *edge* denotes the relative relation of poses (i.e. transformation) between two objects connected by this edge, which can be either a set of *observations* (e.g., image point coordinates corresponding to an observed marker's corners) for pose estimation from the previously mentioned marker-based algorithms, or a set of *constraints* (e.g., known transformation through calibration, known geometrical relationship such as coplanarity and perpendicularity). Figure 4-2 demonstrates such a graph corresponding to the camera marker network in Figure 4-1.

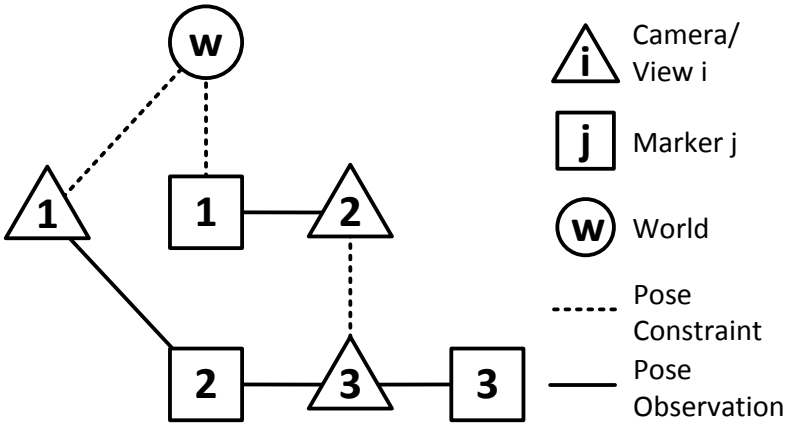


Figure 4-2: Graph representation of a camera marker network.

Therefore, if at least one path exists between any two nodes in such a graph, the relative pose between them can be estimated using algorithms described below. In addition, any loop in the graph means a hidden constraint of poses, which can be used to improve the pose estimation. With more cameras and markers in the network as shown in Figure 4-1, there are more opportunities of creating loops and thus improving pose estimation of the whole network.

Note that this graph abstraction of camera marker networks provides a unified theoretical framework to both systematically manage all observations and relationships in different realizations of such networks, and efficiently find their solutions, in real world applications.

4.2.2 Network Calibration

Two types of calibration are necessary for a camera marker network before its operation. The first type is intrinsic calibration which determines internal parameters (e.g., focal length) of all cameras in the system. This is done either using a planar calibration rig (Zhang Z. 2000) or a 3D calibration field by the direct linear transformation (DLT) algorithm (Abdel-Aziz 1971).

The second type is extrinsic calibration which determines relative poses (e.g. dotted edges in the graph) designed to be calibrated before system operation. There are two kinds of such poses:

1. poses of static markers in the world coordinate frame, and
2. poses between rigidly connected cameras, or camera and markers.

The first kind of poses can be calibrated by traditional surveying methods using a total station. The second kind of poses, however, can hardly be physically surveyed directly since a camera frame's origin and principal directions usually cannot be found or marked tangibly on that camera.

Thus to calibrate a set of *m* rigidly connected cameras, a camera marker graph needs to be constructed as denoted in Figure 4-3. A set of *n* markers' poses need to be surveyed in the world frame. Then when the *m* cameras observe these *n* calibration markers, the graph is formed to estimate each camera's pose in the world frame and thus their relative poses between each other (i.e., edges with question mark) are calibrated. It is suggested to ensure that multiple loops exist in this graph to improve the accuracy of the poses to be calibrated. Such loop exists as long as at least two markers are observed by a same camera simultaneously. It is also worth noting that with enough many calibration markers, each camera's intrinsic parameters can be further optimized together with their extrinsic parameters.

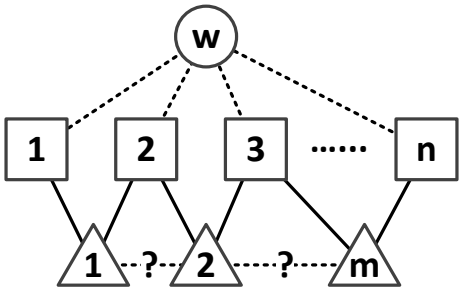


Figure 4-3: A camera marker graph for extrinsic calibration.

4.2.3 Mathematical Solution

For a unified mathematical solution and further uncertainty analysis, a formal definition of a camera marker graph as an abstraction of the corresponding network is defined as follows. This graph **G** contains two parts, a set of nodes **X** and a set of edges **E**, i.e., **G** = (**X**, **E**). Both marker nodes, camera/view nodes, and the world node, are all mathematically represented as parameter vectors encoding the poses of nodes in the world reference frame, i.e., $\mathbf{X} = \{\mathbf{x}_i \mid i = 1, \cdots, N\}$. Note that this **X** is interchangeably termed as the state or configuration of this camera marker network. Usually, each node $\mathbf{x}_i = \left[\mathbf{e}_i^{\mathrm{T}}, \mathbf{t}_i^{\mathrm{T}}\right]^{\mathrm{T}}$ is a 6D column vector, with the first three elements $\mathbf{e}_i$ encoding the orientation of the node in the world reference frame, and the last three elements $\mathbf{t}_i$ encoding the position of the node in the world reference frame. Various parameterizations can be applied for $\mathbf{e}_i$, including Euler angles and axis-angle representation. In this research, the axis-angle parameterization is adopted. The orientation parameters and corresponding rotation matrix can be converted back and forth using the Rodrigues' rotation formula. Both the axis-angle parameterization and the conversion equations are explained in detail in the Appendix.

On the other hand, all edges are divided into two sets, i.e., **E** = (**O**, **C**), where the first set $\mathbf{O} = \{\mathbf{o}_j \mid j = 1, \cdots, M\}$ corresponds to the set of observations (i.e., edges denoted by solid lines),

and the second set $\mathbf{C}=\{\mathbf{c}_k \mid k=1,\cdots,L\}$ corresponds to the set of constraints (i.e., edges denoted by dotted lines). Each observation edge $\mathbf{o}_j=(s_j,e_j,\mathbf{z}_j)$ contains a camera/view node's index s_j and a marker node's index e_j, and also the observation column vector $\mathbf{z}_j$. Usually, in a network with only RGB cameras and markers, $\mathbf{z}_j$ is constructed by stacking all observed 2-dimentional coordinates of image points of feature points on the e_j-th marker. Similarly, each constraint edge $\mathbf{c}_k=(s_k,e_k,\mathrm{g}_k)$ contains indices s_k and e_k of the two nodes involved, and a constraint function $\mathrm{g}_k(\mathbf{x}_{s_k},\mathbf{x}_{e_k})=\mathbf{0}$. Note that edges in this graph encode observations and constraints, which is different from the aforementioned spatial relationship patterns (Pustka et al. 2006) where edges encode transformations between nodes.

4.2.3.1 Initial Solution

Solving such a network is essentially an estimation problem. The final solution is the optimal poses of nodes, $\hat{\mathbf{X}}$, which is most consistent with all the observations and constraints. Since in most cases there are no close form solutions, iterative optimizations are indispensable for obtaining the final solution. Before optimization, an initial solution $\tilde{\mathbf{X}}$ must be provided as a starting point.

Different methods can be used to initialize those poses. If a node is connected to another initialized node by an edge constraining their relative pose to calibrated values, then this node's pose can be initialized by composing the calibrated relative pose and the initialized node's pose. For example, poses of camera 1 and marker 1 in Figure 4-2 can be initialized in this way.

On the other hand, if a node is connected to other nodes only through observation edges, marker based pose estimation can be applied to solve the relative pose between the camera node and the

marker node. Marker 2 and 3 in Figure 4-2 are examples for this initialization method. As mentioned in the previous chapter, it firstly finds a set of 2D geometry features (e.g., points or lines) on an image captured by a calibrated camera, then establishes correspondences between another set of 2D or 3D geometry features on a marker whose pose has been initialized with respect to a certain coordinate frame of interest, and finally estimates the pose of the camera in that coordinate system. If 2D-2D correspondences are used, the pose is typically estimated by homography decomposition (Hartley and Zisserman 2000). If 2D-3D, the pose is typically estimated by solving the perspective-n-point (PnP) problem (Abdel-Aziz 1971; Fischler and Bolles 1981; Lepetit et al. 2009). Two typical marker-based pose estimation algorithms are AprilTag (Olson 2011) and the KEG tracker (Feng and Kamat 2013) developed in Chapter 3.

4.2.3.2 Optimization

Thanks to the graph representation, once all nodes are initialized, a systematic optimization can be applied to the whole graph to find the optimal solution $\hat{\mathbf{X}}$. This optimization is achieved by adjusting all nodes' initial poses $\tilde{\mathbf{X}}$ to minimize all observation residuals or constraint residuals on each edge of the graph.

The observation residuals of an observation edge $\mathbf{o}_j = (s_j, e_j, \hat{\mathbf{z}}_j)$ in such a graph is in fact the difference between the actual measured and stored observation column vector $\hat{\mathbf{z}}_j$ and the predicted observation column vector $\mathbf{z}_j$ calculated using the current pose estimation of the two connected nodes $\mathbf{x}_{s_j}$ and $\mathbf{x}_{e_j}$. In a camera marker network with only RGB cameras, this prediction model is based on the following equation:

$$\mathbf{z} = \boldsymbol{\pi}\left(\mathbf{T}(\mathbf{x}_s)^{-1}\mathbf{T}(\mathbf{x}_e)\begin{bmatrix}\mathbf{y}\\ 1\end{bmatrix}\right), \tag{4.1}$$

where $\mathbf{y}$ is the 3D coordinate of a point on the marker e_j connected to this edge $\mathbf{o}_j$; z is the predicted 2D coordinate of this point's image in camera/view s_j; $\mathbf{T}:\mathbb{R}^6 \to \mathbf{SE}(3)$ is the 6D pose conversion equation explained in the Appendix; and the function $\boldsymbol{\pi}:\mathbb{R}^4 \to \mathbb{R}^2$ is the well-known *central perspective projection* function[7]:

$$\mathbf{z} = \begin{bmatrix} u \\ v \end{bmatrix} = \boldsymbol{\pi}\left(\begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}\right) = \boldsymbol{\pi}\left(\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}\right) = \begin{bmatrix} f_x\left(\frac{1+\sum_{i=1}^{3} r^{2i}k_i}{1+\sum_{i=1}^{3} r^{2i}k_{i+3}}\frac{x}{z} + 2p_1\frac{xy}{z^2} + p_2(r^2+2\frac{x^2}{z^2})\right) + c_x \\ f_y\left(\frac{1+\sum_{i=1}^{3} r^{2i}k_i}{1+\sum_{i=1}^{3} r^{2i}k_{i+3}}\frac{y}{z} + p_1(r^2+2\frac{y^2}{z^2}) + 2p_2\frac{xy}{z^2}\right) + c_y \end{bmatrix} \tag{4.2}$$

in which $r^2 = (x^2+y^2)/z^2$, and $\mathbf{p} = [x, y, z]^{\mathbf{T}}$ is the above point $\mathbf{y}$'s coordinate transformed into the local coordinate frame of camera/view s_j using $\mathbf{x}_s$ and $\mathbf{x}_e$. Notice that this projection function is parameterized by the camera's intrinsic parameters including the linear part f_x, f_y, c_x, c_y and the distortion part k_i $(i = 1,\cdots,6)$, p_j $(j = 1,2)$. Also note that by stacking the predicted 2D image coordinates of each point on this marker e_j results in the predicted observation column vector $\mathbf{z}_j$.

Now, the process of repeating this prediction for all observation edges in $\mathbf{O}$ by equation (4.1) and (4.2), and stacking the results into a single column vector $\mathbf{Z} = \left[\mathbf{z}_1^{\mathbf{T}}, \cdots, \mathbf{z}_M^{\mathbf{T}}\right]^{\mathbf{T}}$, can be abstracted as the following observation function of the system:

$$\mathbf{Z} = \mathbf{F}(\mathbf{X}; \mathbf{Y}, \mathbf{K}), \tag{4.3}$$

[7] There are different variations of central perspective projection function, due to different camera distortion models. In this research, the popular OpenCV model is adopted.

which is parameterized by the known parameters **Y** (usually contains all marker points' local coordinates **y,** measured precisely), and the calibrated parameters **K** (usually encodes all cameras' intrinsic parameters). Thus the observation residual of the system is $\hat{\mathbf{Z}} - \mathbf{Z}$, where $\hat{\mathbf{Z}} = \left[\hat{\mathbf{z}}_1^{\mathbf{T}}, \cdots, \hat{\mathbf{z}}_M^{\mathbf{T}} \right]^{\mathbf{T}}$ is the correspondingly stacked all actual observation vectors.

Similarly, the residuals of all constraint edges **C** can be abstracted as the following function:

$$\mathbf{G}(\mathbf{X}) = \left[\mathrm{g}_1(\mathbf{x}_{s_1}, \mathbf{x}_{e_1})^{\mathbf{T}}, \cdots, \mathrm{g}_L(\mathbf{x}_{s_L}, \mathbf{x}_{e_L})^{\mathbf{T}} \right]^{\mathbf{T}}. \tag{4.4}$$

Finally, as previously mentioned, the maximum likelihood estimation of this camera marker network is obtained as:

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \left\| \hat{\mathbf{Z}} - \mathbf{F}(\mathbf{X}; \mathbf{Y}, \mathbf{K}) \right\|_{\mathbf{P}_{\hat{\mathbf{Z}}}}^2 + \left\| \mathbf{G}(\mathbf{X}) \right\|_{\mathbf{P}_{\mathbf{C}}}^2, \tag{4.5}$$

where $\mathbf{P}_{\hat{\mathbf{Z}}}$ is the a priori covariance matrix of the actual observations $\hat{\mathbf{Z}}$ (typically assumed as $\sigma_u^2 \mathbf{I}$ when image coordinates are measured with a standard deviation of σ_u^2), and $\mathbf{P}_{\mathbf{C}}^{-1/2}$ is the weighting matrix of all the constraints (typically large weight to ensure satisfying of constraints). Note that $\left\| \cdot \right\|_{\mathbf{P}}$ here denotes the Mahalanobis norm with covariance matrix **P**.

Solving this highly non-linear optimization problem is in fact termed as bundle adjustment (Triggs et al. 2000), originated from photogrammetry and rediscovered in computer vision. The well-known Levenberg-Marquardt algorithm is commonly applied to solve this bundle adjustment problem iteratively. When implementing this optimization, programming libraries such as the Ceres (Agarwal and Mierle 2012) or the g2o (Kummerle et al. 2011) solver can be adopted for efficient implementation.

4.3 Uncertainty Analysis

It is not sufficient to only estimate all the poses in a camera marker network. The uncertainty of an estimated pose is critical for the following reasons. Firstly the uncertainty provides a measure of the confidence level of the estimated pose, which is necessary for many downstream applications (e.g., deciding buffer size for collision avoidance between two objects of interests). Secondly it serves as a tool for evaluating the stability of this pose estimation system under different configurations, and thus further guiding to avoid critical configurations that will lead to unstable pose estimation.

4.3.1 Uncertainty Propagation

No matter how complex a camera marker network is and what method is used to get an initial estimate $\tilde{\mathbf{X}}$ (either PnP or homograph decomposition), the uncertainty of the optimized poses $\hat{\mathbf{X}}$ from equation (4.5) can be backward propagated from the observation uncertainty $\mathbf{P}_{\hat{\mathbf{Z}}}$ with linearization of $\mathbf{F}$ around the solution $\hat{\mathbf{X}}$. Since the errors are assumed to come from only the observations (the uncertainties in calibrated parameters $\mathbf{K}$ will be included in future work, but are assumed to be negligible in this chapter as long as all the cameras are carefully calibrated; similarly the uncertainties of all constraints, $\mathbf{C}$, are neglected), one can directly apply the results in (Hartley and Zisserman 2000) to calculate the uncertainty of the optimized states:

$$\mathbf{P}_{\hat{\mathbf{X}}} = (\mathbf{J}^{\mathbf{T}}\mathbf{P}_{\hat{\mathbf{Z}}}^{-1}\mathbf{J})^{-1} = \sigma_u{}^2(\mathbf{J}^{\mathbf{T}}\mathbf{J})^{-1}, \tag{4.6}$$

where $\mathbf{J} = \left.\frac{\partial \mathbf{F}}{\partial \mathbf{X}}\right|_{\hat{\mathbf{X}}}$ is the Jacobian matrix of $\mathbf{F}$ evaluated at $\hat{\mathbf{X}}$, and the diagonal elements of this covariance matrix, $\text{diag}(\mathbf{P}_{\hat{\mathbf{X}}})$, are the marginal variations of each random variables in the estimated pose vector $\hat{\mathbf{X}}$.

4.3.2 Uncertainty and Configuration

Equation (4.6) provides not only a means of evaluating uncertainty of the optimized pose estimation of a camera marker network, but also a tool to predict the system stability at any given configuration **X** before even making any measurements. This is done by evaluating the Jacobian matrix **J** of **F** at that **X**, then applying equation (4.6) to predict **X**'s covariance matrix:

$$\mathbf{P}_{\mathbf{X}}(\mathbf{X}) = \left(\mathbf{J}(\mathbf{X})^{\mathbf{T}}\mathbf{P}_{\hat{\mathbf{Z}}}^{-1}\mathbf{J}(\mathbf{X})\right)^{-1}. \tag{4.7}$$

It is based on the fact that the aforementioned backward propagation of observation uncertainty does not directly rely on specific observations. In fact it directly relies on the configuration **X** around which the linearization of **F** is performed. Thus, when evaluating Jacobian matrix **J** at a configuration **X**, equation (4.7) yields the theoretically best/smallest pose estimation uncertainty that one can expect at that **X**, which denotes the system stability at that configuration.

Furthermore, noticing the fact that equation (4.7) is a function of the system configuration **X**, it provides a method to a directly optimize any camera marker network designs so as to reduce the theoretical uncertainty. This can be done by performing the following nonlinear optimization:

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \mathrm{Cost}\left(\mathbf{P}_{\mathbf{X}}(\mathbf{X})\right), \tag{4.8}$$

subject to constraints such as: 1) all predicted image points by **X** should stay within the photo's nominal size, e.g. 640 by 480 pixels for common webcam image; 2) direct pose constraints on **X** so as to ensure markers' detectability in practice, e.g., camera marker distance not too long. Note that there are various options for the cost function on the covariance matrix, $\mathrm{Cost}: \mathbf{S}_{++}^{n} \to \mathbb{R}$ [8],

[8] $\mathbf{S}_{++}^{n}$ denotes any n by n symmetric positive definite matrix, in this case, any n-dimensional covariance matrix.

according to different network design criteria. Some possible choices of this cost function are the trace of the matrix, $\mathrm{tr}(\cdot)$, or the maximum diagonal element of the matrix, $\max(\mathrm{diag}(\cdot))$. To solve this constrained nonlinear optimization, classic algorithms like trust region reflective (Coleman and Li 1996), active set (Powell 1978), or interior point (Byrd et al. 2000) can be used. For program implementation of equation (4.8), one may take advantage of MATLAB's *fmincon* function in its optimization toolbox.

It is also worth noting that without equation (4.7), the conventional method for such uncertainty analysis is Monte Carlo simulation (Luhmann 2009). However its drawback is that a large amount of independent trials are necessary for credible results. For a camera marker network with many nodes and edges, the resulting time for estimating equation (4.7) for a single $\mathbf{X}$ becomes very long. Thus, it is intractable to evaluate equation (4.8) by Monte Carlo simulation.

4.4 Experimental Results

4.4.1 Single Camera Single Marker

Using the above method, some important empirical conclusions on the basic *single camera single marker system* are found about relationships between system stability and configuration, based on various numerical experiments, which are useful for more complex network designs and are listed as follows.

1. *The marker's origin/position in the camera frame, ${}^{c}\mathbf{t}_{m}$, has the largest uncertainty along a direction nearly parallel to the camera's line of sight to the marker, i.e., ${}^{c}\mathbf{t}_{m}$ itself.*

Figure 4-4 exemplifies this conclusion at two randomly generated poses (and thus the true ones) between a camera and a marker. For left side of this figure, firstly the expected position

uncertainty $\mathbf{P_t}$ was calculated using one of the randomly generated pose $\mathbf{X} = [\mathbf{e}^\mathbf{T}, \mathbf{t}^\mathbf{T}]^\mathbf{T}$ via methods explained in section 4.3.2, shown as the $\chi^2 = 9$ ellipsoid wireframe. Then a Monte Carlo simulation of 100 times was conducted to simulate the camera marker network estimation of the pose $\mathbf{X}$ with image point measurement noise variance $\sigma_u^2 = 0.04$ pixels, and the position components are shown as blue dots in the figure. One can verify that almost all blue dots are within that position uncertainty ellipsoid. Then the eigenvalue decomposition, equivalently singular value decomposition (SVD) or principal component analysis (PCA), was performed on this 3 by 3 position covariance matrix, i.e., $\mathbf{P_t} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]\text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2)[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]^\mathbf{T}$, where $3\sigma_3\mathbf{v}_3$ is the largest direction of position uncertainty as shown in red solid line. One can then verify that this direction is almost parallel with the line of sight direction (the black dashed line). Similar results also present on another randomly generated pose (the right side of the figure).

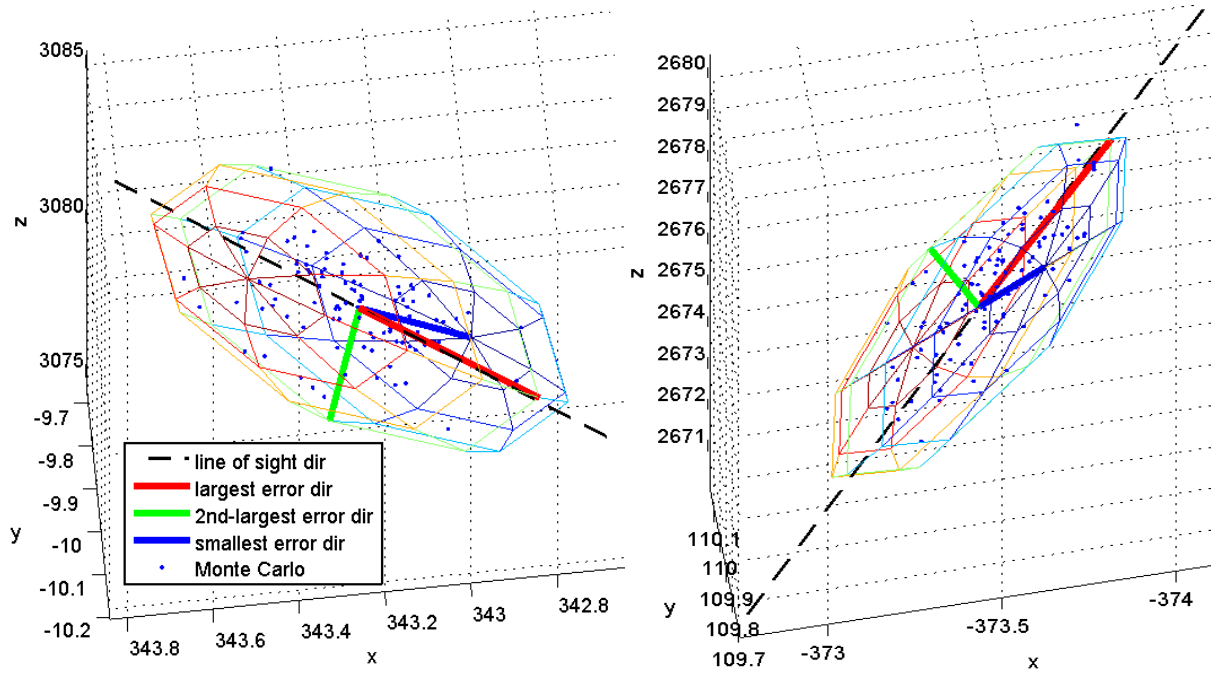


Figure 4-4: Two examples of the largest position error direction.

2. *The largest uncertainty of marker's position in the camera frame increases approximately quadratic to the marker's distance to the camera; compared to which the two smallest uncertainty's increases are almost negligible.*

Figure 4-5 shows an example of this conclusion. After randomly generating a camera pose $({}^{c}\mathbf{R}_m, {}^{c}\mathbf{t}_m)$, and scaling ${}^{c}\mathbf{t}_m$ by a factor s varying from 0.5 to 1.5 without change the orientation ${}^{c}\mathbf{R}_m$, using above methods one can calculate the position uncertainty corresponding to each pose $({}^{c}\mathbf{R}_m, s{}^{c}\mathbf{t}_m)$. Then PCA was performed on each of such position covariance matrix, to extract the smallest ($3\sigma_1$, i.e., PCA error 1 in the figure) and largest ($3\sigma_3$, i.e., PCA error 3 in the figure) uncertainty. The x/y/z error in the figure corresponds to $3\sigma_x$ / $3\sigma_y$ / $3\sigma_z$ respectively where $(\sigma_x^2, \sigma_y^2, \sigma_z^2) = \text{diag}(\mathbf{P_t})$. By comparing $3\sigma_3$ and $3\sigma_x$, one can verify this observation.

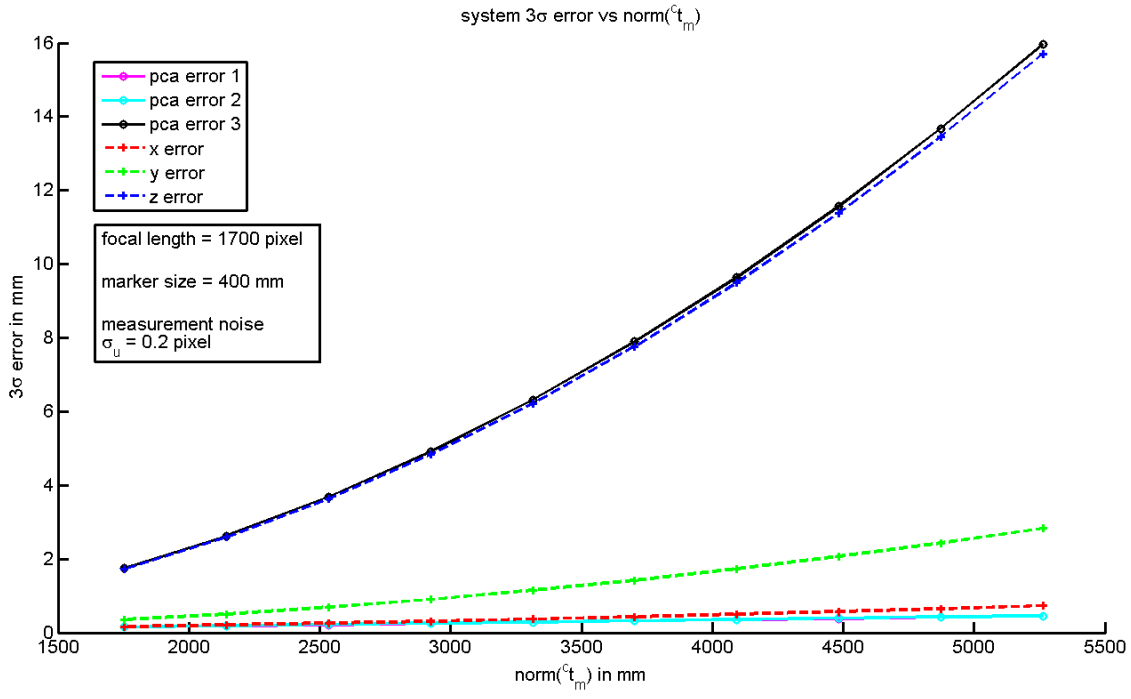


Figure 4-5: Position error vs marker distance.

3. *The largest uncertainty of marker's position in the camera frame increases approximately linear to the camera focal length; compared to which the two smallest uncertainty's increases are almost negligible.*

Figure 4-6 shows an example of this conclusion using similar numerical experiment setup, except that this time the randomly generated camera pose was fixed and the camera focal length f_x and f_y was varied.

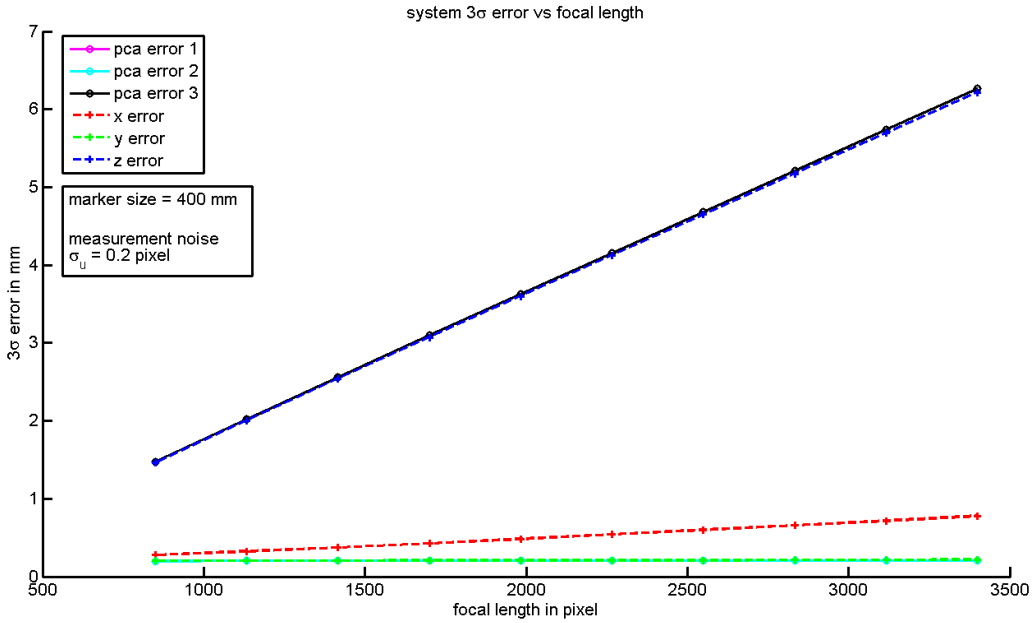


Figure 4-6: Position error vs focal length.

These empirical conclusions are useful for designing and implementing a camera marker network. For example, the third conclusion indicates that a camera with relatively shorter focal length will lead to smaller position estimation uncertainty, which can guide the selection of cameras. Similarly, the second conclusion indicates that although the position estimation error will increase when camera marker distance increases, such increases are slow and small along the directions that are parallel to the camera/view's image plane.

4.4.2 Camera Calibration

Another group of important empirical conclusions are made about the intrinsic camera calibration using either planar (Zhang Z. 2000) or 3D calibration rigs (Abdel-Aziz 1971). As stated in equation (4.3), all of cameras' intrinsic parameters are assumed to be calibrated in advance with negligible uncertainty for equation (4.9) and (4.7) to be valid. Thus it is important to understand how to achieve stable and accurate intrinsic camera calibration.

In fact, the camera intrinsic calibration can be similarly modeled as a camera marker network of one camera node and *n* marker nodes with *n* edges between the camera and each marker (or equivalently, one marker and *n* views). Then the observation prediction equation changes from equation (4.3) to $\mathbf{Z} = \mathbf{F}_c(\mathbf{K}, \mathbf{X}; \mathbf{Y})$ where the camera intrinsic parameters vector $\mathbf{K}$ becomes a part of the state/configuration of this new system, instead of the original parameters for the function $\mathbf{F}$. Thus following the same arguments as in section 4.3, the expected estimation uncertainty of $\mathbf{K}$ and $\mathbf{X}$ becomes:

$$\begin{bmatrix} \mathbf{P}_{\mathbf{K}}(\mathbf{K},\mathbf{X}) & \mathbf{P}_{\mathbf{KX}}(\mathbf{K},\mathbf{X}) \\ \mathbf{P}_{\mathbf{KX}}(\mathbf{K},\mathbf{X})^{\mathbf{T}} & \mathbf{P}_{\mathbf{X}}(\mathbf{K},\mathbf{X}) \end{bmatrix} = \left(\mathbf{J}_c(\mathbf{K},\mathbf{X})^{\mathbf{T}} \mathbf{P}_{\hat{\mathbf{Z}}}^{-1} \mathbf{J}_c(\mathbf{K},\mathbf{X}) \right)^{-1} \tag{4.10}$$

where $\mathbf{J}_c = \left[\frac{\partial \mathbf{F}_c}{\partial \mathbf{K}}, \frac{\partial \mathbf{F}_c}{\partial \mathbf{X}} \right]\Bigg|_{\mathbf{K},\mathbf{X}}$ is the Jacobian matrix for the function $\mathbf{F}_c$ evaluated at $\mathbf{K}$ and $\mathbf{X}$; $\mathbf{P}_{\mathbf{K}}$ is the expected covariance matrix of $\mathbf{K}$; $\mathbf{P}_{\mathbf{X}}$ is that of $\mathbf{X}$; and $\mathbf{P}_{\mathbf{KX}}$ is the expected cross-covariance matrix of $\mathbf{K}$ and $\mathbf{X}$. Similar to equation (4.8), optimizing a calibration network design, for reducing calibration uncertainty, can be done by performing $\min_{\mathbf{X}} \text{Cost}\left(\mathbf{P}_{\mathbf{K}}(\bar{\mathbf{K}}, \mathbf{X})\right)$ on a given camera intrinsic parameters vector $\bar{\mathbf{K}}$. As guidance for improving camera intrinsic calibration, several empirical conclusions based on these analyses are listed as follows.

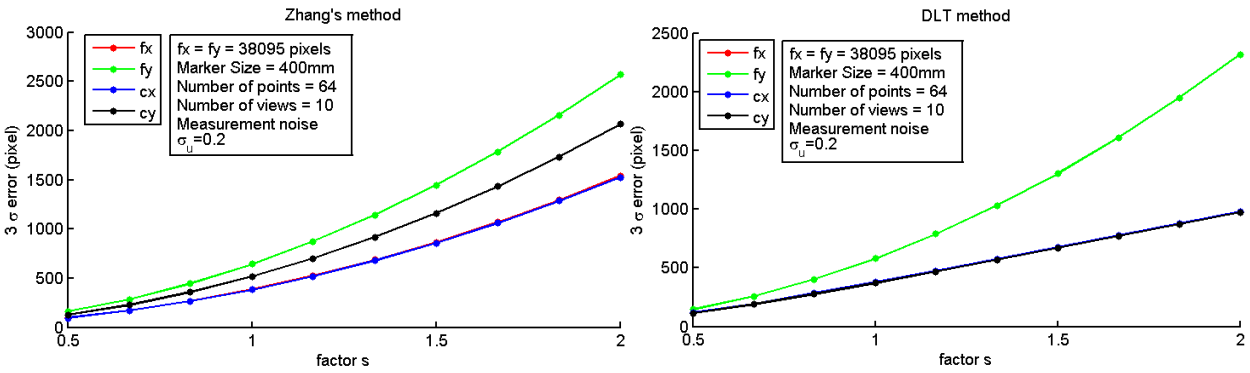


Figure 4-7: Calibration error vs camera marker distance.

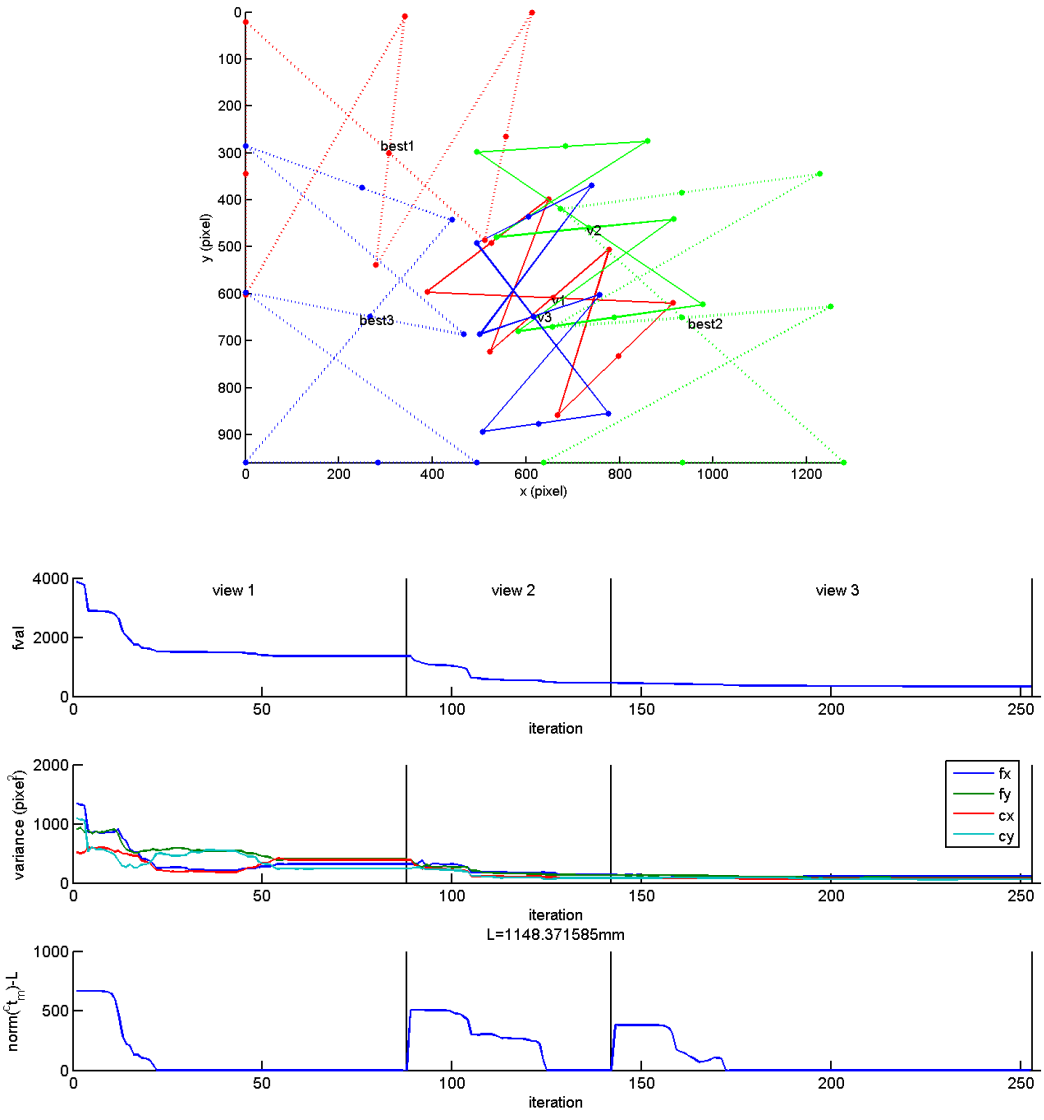


Figure 4-8: Pose optimization for camera calibration.

1. *Shortening camera marker distance during calibration reduces calibration uncertainty.*

Figure 4-7 shows two sets of calibration examples that verify this conclusion. Firstly a random camera calibration configuration of 10 markers (or equivalently 10 views) was generated. Then for each of these 10 markers, their relative orientations in the camera frame, ${}^{c}\mathbf{R}_{m}$, were fixed, while their positions, ${}^{c}\mathbf{t}_{m}$, were scaled by a varying factor s from 0.5 to 2. Then $\mathbf{P}_{\mathbf{K}}$ can be estimated using equation (4.10) for each scaled pose. Finally the diagonal elements of $\mathbf{P}_{\mathbf{K}}$ can be plotted versus this scaling factor s, for both planar (left) and 3D (right) calibration methods.

2. *Calibration has lower uncertainty if images of markers are distributed on regions that are closer to the image margins, especially corners.*

This empirical conclusion comes from the aforementioned optimization $\min_{\mathbf{X}} \text{Cost}\left(\mathbf{P}_{\mathbf{K}}(\bar{\mathbf{K}}, \mathbf{X})\right)$, as shown in Figure 4-8. Firstly a random camera calibration configuration of 3 markers (whose images are shown as solid red, blue, and green lines in the upper figure) was generated, and each marker has 9 feature points. Then the above optimization was performed to find out the "best" calibration configuration $\mathbf{X}$ (and the corresponding markers' images are shown as dashed lines in the upper figure). As can be verified, all markers' images were moved to corners regions. The lower figure shows the optimization details with respect to iterations. In this example the cost function $\text{Cost}(\cdot)$ is selected to be the sum of the first four diagonal elements of $\mathbf{P}_{\mathbf{K}}$ (shown as the *fval* in the figure), i.e. the sum of the variances of the camera's linear intrinsic parameters f_x, f_y, c_x, c_y. It's worth noting that as $\mathbf{X}$ becomes larger, this becomes a higher dimensional optimization which could be more and more inefficient. To make it tractable, this optimization of $\mathbf{P}_{\mathbf{K}}$ can be approximated by iteratively adjusting each marker pose $\mathbf{x}_i$ in this calibration

network, instead of the original batch adjustment of all markers' poses at a same time, so that it is always a 6D optimization. Moreover, this calibration configuration optimization can also be applied in camera calibration methods with user guidance (Richardson et al. 2013) for efficiently selecting the next best marker pose to reduce overall the calibration uncertainty.

4.5 Conclusions

In conclusion, this chapter summarized different solutions addressing limitations of a single camera's narrow FOV into a unified theoretical framework, which is termed as camera marker networks. This framework can be used to model both pose estimation and also camera calibration (both intrinsic and extrinsic calibration) processes. A graph abstraction of different realizations of such networks was developed to both systematically manage all observations and relationships, and efficiently find their solutions. A mathematical notation and solution based on such abstraction was then established. Moreover, a systematic uncertainty analysis was developed based on the uncertainty backward propagation in estimation theory. Such analysis avoids traditionally time-consuming Monte Carlo simulation and enables optimization of network configurations in order to reduce estimation uncertainty. Finally, based on various numerical experiments, several important empirical conclusions were drawn to guide better camera calibration and pose estimation. The real world experiments proving this proposed method's effectiveness will be demonstrated in the following chapters that describe applications.

Several future directions to improve this proposed method are identified. This includes 1) performing similar uncertainty analysis on several basic camera marker network configurations, such as single camera multiple markers or multiple cameras and markers; 2) incorporating calibration uncertainty into the analysis: and 3) including hard constraints (directly encoding constraints into parameters, i.e. nodes) in addition to current soft constraints $\mathbf{C}$.

Part II: Applications in Robotic Construction Machinery

"I hear and I forget; I see and I remember; I do and I understand."—Xunzi[9]

This part includes two robotic construction machinery applications of the algorithms developed in part I, to validate their effectiveness and to test their potential in real world industrial scenarios.

The architects nowadays prefer to design many buildings with non-planar and curved elements. Using robotic manipulators to assemble and construct such elements is appealing because of the precision, efficiency, and repeatability of their movements. However different from traditional manufacturing, robotic manipulators need to work onsite. Thus the in-situ pose estimation is important. Chapter 5 describes how marker based pose estimation can be applied in such cases.

Another application is described in Chapter 6 about excavation monitoring and guidance. To improve the excavation safety, the poses of excavators' key components need to be continuously monitored to alert operators for any potential collisions. Similarly excavations according to design profiles can be accelerated if buckets' positions are always known. A prototype was implemented in this chapter to evaluate the solutions accuracy.

[9] The original Chinese text is most likely from 《荀子・儒效篇》：“不闻不若闻之，闻之不若见之，见之不若知之，知之不若行之；学至于行之而止矣”. It is frequently and mistakenly accredited to Confucius in the West.

Chapter 5
Autonomous Onsite Robotic Assembly and As-Built Scanning

EVE: "Name?" WALL-E: "WALL-E." EVE: "WALL-E?"—Andrew Stanton

5.1 Introduction

Several studies have argued that among all industries, construction has seen a significant productivity decrease over the last several decades compared to other industries (Rojas and Aramvareekul 2003). Construction has also been documented to have some of the highest rates of workspace injuries and fatalities (Bureau of Labor Statistics 2013). Automation and robotics in construction (ARC) has the potential to relieve human workers from repetitive and dangerous tasks, and has been extensively promoted in the literature as a means of improving construction productivity and safety (Balaguer 2004).

Compared to the tangible benefits of automation and robotics identified by the manufacturing industry, the construction industry is still exploring feasible and broadly deployable ARC applications (Balaguer 2004). This can be attributed to several commercial and technical challenges. From the commercial perspective, the fragmented and risk-averse nature of the construction industry leads to little investment in ARC research causing construction to lag behind other industries (Saidi et al. 2008). On the other hand, as described next, there are several

technical complexities inherent in construction that have contributed to hindering the successful development and widespread use of field construction robots.

5.1.1 Technical Challenges

5.1.1.1 Unstructured Construction Environments

Automated and robotized manufacturing facilities are typically considered as structured environments, since both the machines and evolving products either stay in their predefined locations or move on predesigned and typically fixed paths. In general, such environments do not change shape or configuration during the performance of manufacturing tasks, making the enforcement of tight tolerances possible (Milberg and Tommelein 2003). In contrast, construction sites can typically be considered unstructured since they are constantly evolving, and dramatically changing shape and form in response to construction tasks. Building components are moved around without fixed paths or laydown/staging areas. Various physical connections are established through improvisation in response to in-situ conditions, making tight tolerances hard to maintain and enforce (Milberg and Tommelein 2005).

5.1.1.2 Mobility of Construction Manipulators

In manufacturing, factory robotics typically involves robotic platforms that are generally stationary (or have limited linear mobility) and partially complete products that arrive at robot workstations and precisely localize themselves in the robots' base reference frames. Precision is achieved by controlling the pose of the moving (and evolving) product, and the robots themselves are programmed to manipulate the products through fixed trajectories. Thus, from a mobility and cognitive perspective, a factory robot has little responsibility and autonomy. Control is achieved by enforcing tight tolerances in moving and securing the product in the

manipulator's vicinity. However, this spatial relationship is reversed in construction. A construction robot has to travel to its next workface (or be manually set up there), perceive its environment, account for the lack of tight tolerances, and then perform manipulation activities in that environment. This places a significant mobility and cognitive burden on a robot intended for construction tasks even if the task itself is repetitive.

This discussion highlights that factory-style automation on construction sites requires development of robots that are significantly more mobile and perceptive when compared to typical industrial robots. Such on-site construction robots have to be able to semantically sense and adjust to their unstructured surroundings and the resulting loose tolerances. This chapter presents a new high-accuracy 3D machine vision metrology for mobile construction robots. The developed method uses fiducial markers to rapidly establish a local high-accuracy control environment for autonomous robot manipulation on construction sites. Using this method, it is possible to rapidly convert a portion of a large unstructured environment into a high-accuracy, controllable reference frame that can allow a robot to operate autonomously.

The rest of the chapter is organized as follows. Related work is reviewed in section 5.1.2. The technical approach is discussed next in detail in section 5.2. The experimental results of both assembly and scanning are shown and discussed in section 5.3. Finally, in section 5.4, the conclusions are drawn and the future work is summarized.

5.1.2 Previous Work

5.1.2.1 Robotic Manipulators in Construction

The construction community has pursued research on robotic manipulators for several decades: for example, Slocum and Schena (1988) proposed the Blockbot for automatic cement block wall

construction; Pritschow et al. (1993) identified the needs and requirements of a brick-laying robot for masonry construction and developed a control system for such robots.

A large portion of the construction robotic manipulator research focused on mechanics and control of specific construction activities. Fukuda et al. (1991) discussed the mechanism and the control method of a robotic manipulator in construction based on human-robot cooperation. Yu et al. (2009) proposed an optimal brick laying pattern and trajectory planning algorithm for a mobile manipulator system, with computer simulation to test its efficiency. Hansson and Servin (2010) developed a semi-autonomous shared control system of a large-scale manipulator in unstructured environments, with a forwarder crane prototype to test its performance. Chung et al. (2010) proposed a new spatial 3 degree-of-freedom (DOF) parallel type master device for glass window panel fitting task. Gambao et al. (2012) developed a modular flexible collaborative robot prototype for material handling, although without any perception sensors for capturing the working environment.

Another important aspect of construction robotic manipulators lies in sensing and perception. Kahane and Rosenfeld (2004a) proposed a "sense-and-act" operation concept enabling an indoor mobile robot to position itself with approximately 10 cm accuracy, using a CCD camera and several laser projectors. Kim and Haas (2000) proposed automatic infrastructure inspection and maintenance using machine vision for crack mapping. Gambao et al. (2000) developed a robot assembly system based on laser telemeter sensors of 6 to 15 mm positional precision. During these research studies, it was generally realized that increasing the level of autonomy for construction robots requires high accuracy localization of the robot: from 3-5 cm indoor positional accuracy for contactless construction tasks such as spray-painting, to 2-3 mm accuracy for more precise tasks demanding direct contact between manipulator and building components

(Shohet and Rosenfeld 1997). This requirement has posed a significant challenge for ARC because even by using current state-of-the-art simultaneous localization and mapping (SLAM) techniques, such accuracy is hard to achieve at large scales (Kümmerle et al. 2009). In order to address this issue, in this study vision-based pose estimation algorithms was chosen since they can achieve high accuracy locally around a visual marker (Feng and Kamat 2013; Olson 2011).

5.1.2.2 3D As-Built Modeling in Construction

3D as-built modeling (e.g., BIM) plays an important role in a wide range of civil engineering applications. This modeling process usually starts with collecting 3D point clouds of sites of interest. Paul et al. (2007) utilized a 6DOF anthropomorphic robotic arm to get the 3D mapping of a complex steel bridge with a laser range scanner. Brilakis et al. (2010) outlined the technical approach for automated as-built modeling based on point clouds generated from hybrid video and laser scanning data. Akula et al. (2013) explored different 3D imaging technologies, e.g., 3D image system, image based 3D reconstruction and Coherent Laser Radar scanner, to map the locations of rebar within a section of a railway bridge deck in order to assist a future drill operator in making real-time decisions with visual feedback. Zhu and Donia (2013) investigated the advantages and drawbacks of RGBD cameras in as-built indoor environments modeling, with evaluation on the accuracy of collected data, the difficulty of automatic scan registration and the recognition of building elements, demonstrating RGBD camera's potential in as-built BIM modeling. In this research, the automatic planning, scanning and registration of point clouds obtained from a 3D camera mounted on the manipulator are achieved with the visual marker-based metrology and the manipulator's internal encoders.

Once 3D point clouds are obtained, CAD-like geometric models can be generated. For example, Son et al. (2014) automatically extracted 3D pipeline models from laser scanning data based on

the curvature and normal of point clouds; Han et al. (2012) proposed an automated and efficient method to extract tunnel cross sections from terrestrial laser scan (TLS) data. While this research focuses more on the automatic registration of different frames of point clouds, the resulting registered point clouds could be input into such algorithms to generate semantically meaningful CAD-like geometric entities for as-built BIM.

5.1.2.3 Robotic Manipulators in Architecture

Recently the architectural design community has also shown an increased interest in industrial robotics, with many academic programs investing in their own robotic work cells[10]. Capitalizing on the machines' inherent flexibility, they have leveraged the industrial robot as a development platform for the exploration and refinement of novel production techniques in which material behavior is intrinsically linked to fabrication and assembly logics. As part of the general ecosystem of industrial robotics, computer vision systems has begun to play an increasingly important role in these research initiatives, with a number of architectural research groups developing interfaces for accessible hardware such as the Microsoft Kinect.

Initially the majority of architectural robotic research utilizing computer vision has revolved around its application at the micro scale, using a vision feedback system to make incremental adjustments to a robotic strategy based upon local variations. Examples of this include Dierichs et al.'s research (2012) into poured aggregate structures at the Institute for Computational Design at University of Stuttgart and Dubor and Diaz's project (2012) Magnetic Architecture from the Institute for Advanced Architecture of Catalonia, two robotics projects which use the Microsoft Kinect to provide information on local variations from the intended design geometry, which is then used to generate incremental adjustments in succeeding operations. While beneficial as a

[10] http://www.robotsinarchitecture.org/map-of-robots-in-architecture

means to adjust for material variation and machine error, these implementations are not robust enough for the in-situ robotics due to the complexities of construction sites.

Acknowledging the limitations of processes developed within the safety of the research laboratory, architects have slowly begun to explore application of computer vision at the macro scale for adaptive path planning. At the fore-front of this work has been the research conducted at the ETH Zurich led by Fabio Gramazio and Matthias Kohler. Heavily invested in the potential of construction site robotics, their work has included the development of hardware/software solutions that allow industrial robotics to dynamically adjust their operations at both the micro and macro levels. This research is best exemplified in their ECHORD project[11], in which an eight-meter-long module wall was assembled by an ABB robot mounted to a mobile track system (Helm et al. 2012). Constructing a stacked module wall along a gestural path captured by the robot's computer vision system, this mobile robot also used the same system to reposition itself on the construction site (expanding its operable reach envelope) and make local adjustments based on topographic variation. Without using an extensive sensor suite like the one used in ECHORD, the robotic platform described in this chapter successfully built similar module walls purely based on perceived information from a single camera; moreover, as-built 3D point clouds of the assembled outcome can be readily obtained if the robotic platform is equipped with a 3D camera on the manipulator, for purposes like as-built BIM generation or construction progress documentation.

[11] http://dfab.arch.ethz.ch/web/e/forschung/198.html

5.2 Technical Approach

5.2.1 System Overview

The developed robotic in-situ assembly and scan system consists of several major components, as shown in Figure 5-1. The assembly workflow of the system consists of an offline design process and an online building process. During the offline process, a designer models the intended structure in 3D, which is then analyzed and validated by the assembly planer, outputting an assembly plan for the online process. The online assembly process uses a fixed camera mounted on the base of the robot for providing images to the pose estimator to detect staged building components and estimate their poses, and more importantly localize the robot itself in the local building reference frame. Having computed this information, the plan achiever then sequentially transforms each step from the automatically generated assembly plan into an executable command, which can be interpreted by the robot controller and subsequently executed by the robot manipulator. In addition, the visualization component also receives the information generated by the pose estimator as well as the robot's real-time pose feedback from the controller, to simultaneously represent the actual on-site assembly process into a 3D virtual environment for improved monitoring.

Similarly for the scan workflow, based on design information, the scan planner can feed a scanning plan to the plan achiever, controlling the robot manipulator to stop and capture 3D images at a series of desired poses using the 3D camera mounted on the manipulator. Since the robot platform is aware of both its base's pose relative to the local building reference frame through the marker-based metrology, and the 3D camera's pose relative to its base through its internal encoders, the scan register can then readily transform each frame of captured 3D images

into the same local building reference frame, resulting a 3D point cloud describing the construction site.

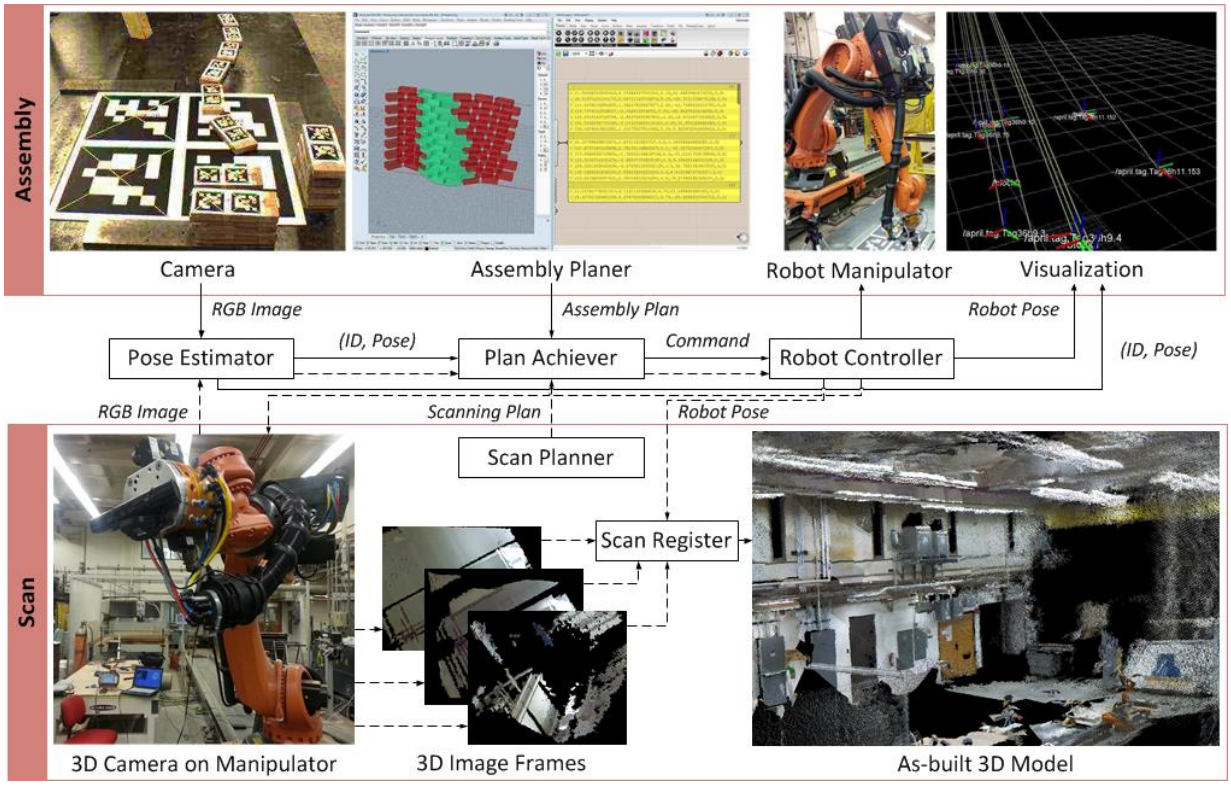


Figure 5-1: Overview of the autonomous assembly and scanning system.

5.2.2 Calibration of Pose Estimator

Before introducing the details of other components of the system, it is important to discuss how the pose estimator is calibrated, since this is crucial to the level of assembly/scan accuracy that the system can achieve. This process includes two steps to be finished before system operation: intrinsic and extrinsic calibration of the camera. Intrinsic calibration involves estimating the camera's focal length, principle point's position on the image plane, and distortion parameters. On the other hand, extrinsic calibration aims to determine: 1) the relative 6-DOF pose of the camera in the robot's base coordinate frame for assembly, and 2) the relative 6-DOF pose of the 3D camera in the robot's tool coordinate frame for scan registration. It must be noted that both

intrinsic and extrinsic calibration are one-time processes, as long as the camera/3D camera is fixed-focus and rigidly mounted in the robot's base/tool coordinate frame (e.g., fixed installation on the robot's base/end-effector).

5.2.2.1 Intrinsic Parameters

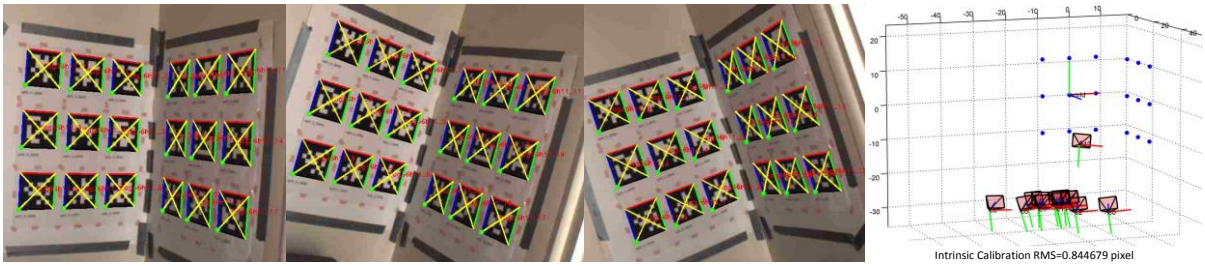


Figure 5-2: Intrinsic calibration of the camera.

Unlike the popular plane-based camera calibration method (Zhang Z. 2000) implemented in OpenCV[12] and Matlab Calibration Toolbox[13], it is chosen in this research to calibrate the camera using a 3D rig, which is similar to the classic calibration in photogrammetry. This 3D rig was made by attaching N Apriltags (Olson 2011) on two intersecting planes forming a 90 degree angle (as shown in the top of Figure 5-2) so that the 3D coordinate **X** of each Apriltag's center could be readily measured.

The process of calibration was then simply taking a sequence of M images of the rig and inputting them into the camera calibration tool[14] developed by me, which takes advantage of the Apriltag detection algorithm to detect the 2D image coordinate **U** of each tag center and establish correspondence with **X**. Then the initial camera intrinsic and extrinsic parameters can be obtained through Direct Linear Transform (DLT) (Hartley and Zisserman 2000) and

[12] http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html
[13] http://www.vision.caltech.edu/bouguetj/calib_doc/
[14] Available at https://code.google.com/p/cv2cg/#apriltag

subsequently optimized by bundle adjustment (Triggs et al. 2000), which minimizes the re-projection error by tuning both intrinsic (**K**) and extrinsic (**R** and **t**) camera parameters as following equation (5.1):

$$\arg\min_{\mathbf{K},\{\mathbf{R}_i,\mathbf{t}_i\}} \sum_{i=1}^{M}\sum_{j=1}^{N} \left\| \mathbf{U}_{i,j} - \pi\left(\mathbf{K}(\mathbf{R}_i\mathbf{X}_j + \mathbf{t}_i)\right) \right\|^2, \tag{5.1}$$

where the perspective division function $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ converts a 2D homogeneous coordinate into a 2D Cartesian coordinate.

Benefitting from the high corner detection accuracy of Apriltag as well as the 3D rig, this intrinsic calibration produces more robust and repeatable results than the alternatives mentioned above.

5.2.2.2 Extrinsic Parameters

Once the camera's intrinsic parameters are calibrated, the camera's relative pose in the robot's base coordinate frame, ${}^{r}\mathbf{T}_c = [{}^{r}\mathbf{R}_c, {}^{r}\mathbf{t}_c; \mathbf{0}, 1]$, can be estimated using an extrinsic calibration marker containing L (L>1) Apriltags with known size and spacing.

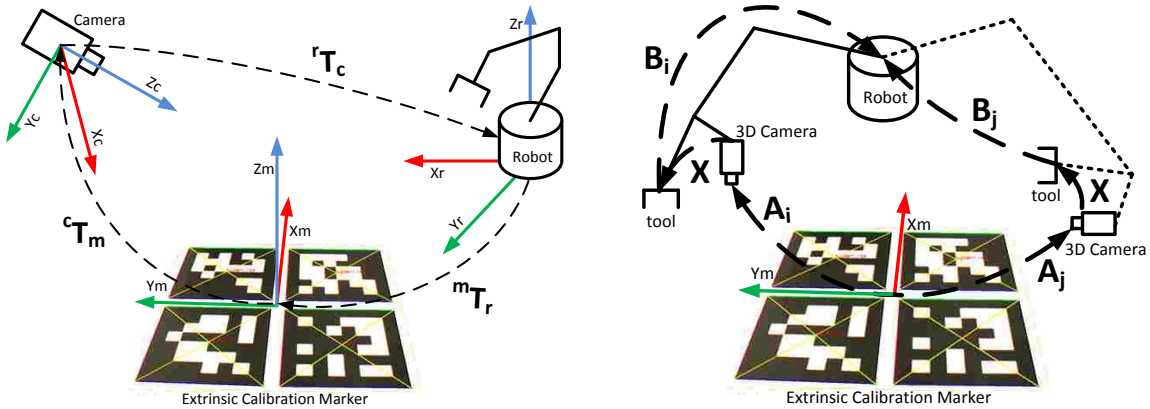


Figure 5-3: Extrinsic calibration between the camera and the robot base.

As shown in Figure 5-3(a), ${}^{r}\mathbf{T}_{c}$ can be composed from the two other poses, ${}^{m}\mathbf{T}_{r}$, the robot's pose in the extrinsic calibration marker's coordinate frame, and ${}^{c}\mathbf{T}_{m}$, the marker's pose in the camera coordinate frame, by ${}^{r}\mathbf{T}_{c} = ({}^{c}\mathbf{T}_{m}\,{}^{m}\mathbf{T}_{r})^{-1}$.

This extrinsic calibration of the camera used for assembly consists of the following steps:

1. Fix the marker in the camera's field of view;
2. Manually control the robot manipulator to pinpoint at least 4 non-collinear points on the marker and record their 3D coordinates ${}^{r}\mathbf{X}$ in the robot's base coordinate frame; also measure their local 3D coordinates ${}^{m}\mathbf{X}$ in the marker's reference frame (by setting all Z coordinates to be zero);
3. Take an image from the camera and detect the L Apriltags' 4L corners' 2D image coordinates **U**.

With this information collected, the ${}^{m}\mathbf{T}_{r}$ can be estimated using the well-known rigid body registration (Besl and McKay 1992) from 3D point set ${}^{r}\mathbf{X}$ to ${}^{m}\mathbf{X}$, while the ${}^{c}\mathbf{T}_{m}$ can be estimated by decomposing the homography between ${}^{m}\mathbf{X}$ and **U** using the previously calibrated camera intrinsic parameter **K** (Zhang Z. 2000; Feng and Kamat 2013).

In order to improve the extrinsic calibration's accuracy, a non-linear optimization of ${}^{c}\mathbf{T}_{m}$ is also performed in addition, since during the homography decomposition, a polar decomposition is performed to get a valid rotation matrix ${}^{c}\mathbf{R}_{m}$, which causes the result to be non-optimal. This optimization, as shown in equation (5.2), can be done by tuning the initial ${}^{c}\mathbf{T}_{m}$ to minimize the re-projection error:

$$\arg\min_{{}^{c}\mathbf{R}_m,\,{}^{c}\mathbf{t}_m} \sum_{j=1}^{4L} \left\| \mathbf{U}_j - \pi\left(\mathbf{K}({}^{c}\mathbf{R}_m\,{}^{m}\mathbf{X}_j + {}^{c}\mathbf{t}_m)\right)\right\|^2 \tag{5.2}$$

While the extrinsic calibration of the 3D camera used for scanning could be done in a similar manner, a faster and more efficient solution is to perform the classic robot hand-eye calibration: by moving the robot end-effector to different poses relative to its base ($\mathbf{B}_i, \mathbf{B}_j$) and correspondingly estimate through Apriltags the extrinsic calibration marker's pose relative to the 3D camera reference frame ($\mathbf{A}_i, \mathbf{A}_j$), equations $\mathbf{A}_i\mathbf{X}\mathbf{B}_i = \mathbf{A}_j\mathbf{X}\mathbf{B}_j$ can be established where $\mathbf{X} = {}^{t}\mathbf{T}_c$, representing the 3D camera's pose relative to the robot's tool coordinate frame, as shown in Figure 5-3(b). After rearranging such equations into the form of **AX=XB** where $\mathbf{A} = \mathbf{A}_j^{-1}\mathbf{A}_i$ and $\mathbf{B} = \mathbf{B}_j\mathbf{B}_i^{-1}$, the calibration can be solved by Tsai and Lenz's methods (Tsai and Lenz 1989). It's worth noting that the previous extrinsic calibration of the camera for assembly could also be solved as a hand-eye calibration problem, even though the camera is not fixed on the manipulator's end-effector, to further simply the calibration process.

5.2.2.3 Calibration Validation

The calibration can be validated by the following procedure:

1. Fix the extrinsic calibration marker to a new pose that is different from the one used in the calibration;
2. Measure, in robot's base frame, the 3D coordinates ${}^{r}\mathbf{X}$ of a set of P corner points on the marker (e.g., the 4L corners used previously);
3. Take an image and find out the corresponding 3D coordinates ${}^{c}\mathbf{X}$ in the camera reference frame using Apriltags;

4. Calculate the residual using equation (5.3):

$$\sqrt{\frac{1}{M}\sum_{j=1}^{P}\left\| {}^{r}\mathbf{X}_j - ({}^{r}\mathbf{R}_c\,{}^{c}\mathbf{X}_j + {}^{r}\mathbf{t}_c)\right\|^2} \tag{5.3}$$

This residual should ideally approach zero, as smaller residual indicates better calibration accuracy.

5.2.3 Automatic Assembly Planer

5.2.3.1 Algorithmic Architectural Design

Starting with the introduction of Ivan Sutherland's Sketchpad at MIT in the 1963, the architectural exploration of computation has focused on the digital environment's ability to represent an object as a system "compromised of and working with a series of interrelated systems" (Ahlquist and Menges 2011), a surprising contrast to the discrete geometric representations found in many 2D and 3D CAD applications. Initially as a domain of specialized research groups embedded in academia or commercial practices, this systems approach to digital design has become increasingly commonplace. In the 1990s many architects, unsatisfied with the capabilities presented by off-the-self software, began to develop their own software solutions through both higher-level scripting languages for CAD packages and ground-up application development. Currently, visual programming interfaces that afford designers quick access to the potential of computation without the effort of coding syntax are available as plug-ins for popular commercial software, such as Dynamo[15] for Autodesk's Revit and Grasshopper[16] for McNeel's Rhinoceros.

[15] http://autodeskvasari.com/dynamo
[16] http://www.grasshopper3d.com

This systems approach (and its respective tools) was implemented to automatically derive the robotic positioning data for each building block in a curved stack-unit wall from a single user-generated non-uniform rational basis spline (NURBS) surface. Working from a predetermined block size, an algorithm developed in the Grasshopper plug-in for Rhinoceros extracts latitudinal section curves from the input surface, generating a running bond pattern of the said blocks. Each block is checked for volumetric collisions with adjacent blocks, color-coding collisions in the Rhinoceros environment. Instead of applying simple heuristics to arbitrarily resolve these collisions, the color-coding can provide real-time feedback, engaging the designer to actively participate in the development of the curved stack-unit wall. Simultaneously the necessary positional information for each block is output for the generation of the assembly plan. Combined into a single algorithmic process, these functions "enable an explicit and bidirectional traversal of the modern division between design and making" (Pigram et al. 2012), reinforcing the implications of William Mitchell's statement that "architects tend to draw what they can build, and build what they can draw" (2001).

5.2.3.2 Assembly Plan Generation and Simulation

Given the final positions and orientations of all the building blocks in the design, the assembly plan is generated and written into a text file stored for the plan achiever to process later during the online building phase.

This assembly plan file contains a list of sequential instructions for the robot manipulator to build the designed structure. Each line in the file corresponds to an instruction. For example, the following plan file segment will instruct the robot manipulator to first grab a building component named "block0" directly from above (line 1-4), then lift it vertically up for 500 mm (line 5) and finally place it at its destination in another reference frame named "building" (line 6-8):

1: *Gripper* 0
2: *Goto block* 0 0 0 500 0 0 0
3: *Goto block* 0 −12 −10 −10 0 0 0
4: *Gripper* 1
5: *Shift* 0 0 500 0 0 0
6: *Goto building* 200.00 −300.00 500.00 −63.92 0.00 0.00
7: *Goto building* 200.00 −300.00 19.05 −63.92 0.00 0.00
8: *Gripper* 0

Currently, 3 types of instructions are implemented in the system:

1. **Gripper 0/1:** Control the manipulator's gripper to open (0) or close (1);
2. **Goto reference_frame x y z a b c:** Control the manipulator to move to a new pose (x, y, z, a, b, c) in the reference frame, in which the (x, y, z) is the new position and (a, b, c) specifies the new orientation as three Euler angles in "ZYX" order;
3. **Shift x y z a b c:** Control the manipulator to incrementally move by (x, y, z, a, b, c).

This assembly plan can also be simulated in Rhinoceros to check if there exists any self-collision between the robot manipulator and the wall during the building process, as shown in Figure 5-4

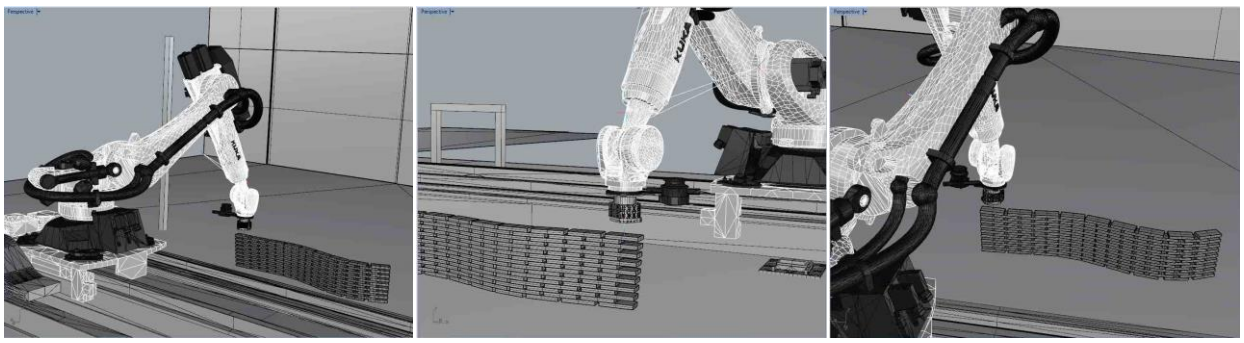


Figure 5-4: Assembly simulation.

5.2.4 Vision-based Plan Achiever

5.2.4.1 Rapid Setup of Building Reference Frame

As previously mentioned, the reversed spatial relationship of product and manipulator on construction sites poses a significant challenge for autonomous mobile robots. This is notably different from typical autonomous manufacturing spatial configurations, where robots' bases are either stationary or have finite mobility, and materials/components can be readily staged at fixed locations within the manipulators' static workspaces. In contract, for mobile robots to autonomously perform building tasks on unstructured construction sites, their bases require significant mobility, and consequently their manipulators' workspaces are not fixed with respect to the construction site. In order to complete building tasks at the correct locations and assemble materials into their intended poses, a robotic system must be able to establish the accurate 6-DOF transformation between the robot's base and the building reference frame at all times. As pointed out in (Shohet and Rosenfeld 1997), this requires the localization accuracy to be at least at centimeter level, which could not be easily achieved using state-of-the-art Visual SLAM style techniques for mobile robots.

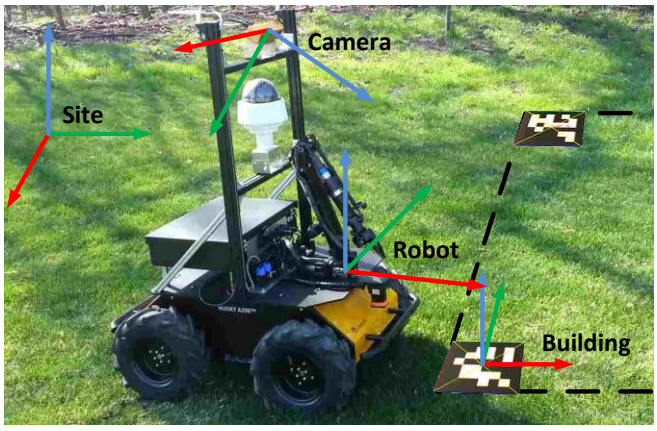


Figure 5-5: Different reference frames

In order to address this challenge, a convenient and accurate solution is proposed using planar marker-based pose estimation (Olson 2011; Feng and Kamat 2013), as shown in Figure 5-5. By 1) attaching fiducial markers at appropriate locations on-site where building tasks are to be performed, 2) surveying their poses ${}^{m}\mathbf{T}_{b}$ in the building reference frame using a total station, and 3) storing these poses inside the system's database, a mobile robot can readily estimate its base's pose ${}^{b}\mathbf{T}_{r}$ inside the building reference frame using equation (5.4) whenever its on-board camera detects such a marker, based on previous calibration results:

$$ {}^{b}\mathbf{T}_{r} = ({}^{r}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{m}\,{}^{m}\mathbf{T}_{b})^{-1} \tag{5.4} $$

5.2.4.2 Conversion from Plans to Commands

With the information input from the pose estimator, the vision-based plan achiever starts to execute the assembly plan generated beforehand, according to the following procedure:

1. Read a single plan step (i.e. one line) from the assembly plan file;
2. Wait until all the poses needed to convert the step into a building command are available;
3. Convert this step into a command that is executable by the robot controller;
4. Send the command to the robot controller;
5. Wait for the controller to complete the command;
6. Repeat this process unless all plan steps are completed, i.e. the plan is achieved.

It must be noted that the core step of this procedure is the conversion from a plan step to an executable command. This is because the poses stored in the previously generated assembly plan are not completely specified in the robot's base reference frame. Recall that every pose in the "*Goto*" step is specified in a "*reference_frame*" relatively. Specifying all steps in the assembly

plan in the robot base reference frame is not possible because: a) during the design phase the designer conceives all component locations in the building reference frame; b) the robot's base is expected to be mobile during the building phase; and c) more importantly, the building components will be arbitrarily transported and staged in the building reference frame in the vicinity of the robot manipulator's workspace. This, in fact, is one of the core differences between on-site construction automation and manufacturing automation.

In this research, this conversion is facilitated by the aforementioned rapid setup of the building reference frame using markers. As long as the pose estimator can detect and report the transformation between the on-board camera and the marker used to specify the building reference frame, the poses in the plan steps can be readily converted to the robot base frame using equations similar to (5.4). Similarly, by attaching markers on the building components, the robot manipulator can detect and clasp them autonomously after the corresponding plan steps are converted.

5.2.5 As-built Point Clouds from 3D Camera

Due to the operation and maintenance requirement, nowadays construction project owners are often more satisfied if given access to not only an as-designed BIM but also an as-built BIM of a project. As reviewed previously, a common practice is to start with various 3D scanners such as TLS to create 3D point clouds of the built environment. These point clouds are then manually or semi-automatically abstracted into CAD-like 3D geometric objects conveying semantic meanings, such as walls, floors, doors, windows and utility pipelines.

Since more often than not limited by the 3D scanner's field of view, a single scan cannot fully cover a construction project's outcome, multiple scans need to be carefully planned, performed

and then registered into a single reference frame. As pointed out by previous work, the automation of such planning and registration becomes important to improve the productivity as well as the quality of the final point clouds.

Facilitated by the proposed visual marker-based metrology and the robot's internal encoders, the process to automatically plan, scan and register different frames of 3D point clouds becomes readily available on the proposed robotic platform with minimal effort of mounting and calibrating a 3D camera on the manipulator, when reusing existing components for autonomous assembly. Whenever a new frame of 3D point clouds ${}^{c}\mathbf{P}_i$ $(\forall i = 1, 2, 3\cdots)$ is captured (and thus expressed naturally in the 3D camera's reference frame), controlled by the scan planner, they are transformed into the building reference by equation (5.5):

$$ {}^{b}\mathbf{P}_i = {}^{b}\mathbf{T}_r \, {}^{r}\mathbf{T}_t \, {}^{t}\mathbf{T}_c \, {}^{c}\mathbf{P}_i, (\forall i = 1, 2, 3, \cdots) \tag{5.5} $$

where ${}^{t}\mathbf{T}_c$ is the 3D camera's pose relative to the robot's tool reference frame, i.e., hand-eye calibration result obtained in section 5.2.2.2, ${}^{r}\mathbf{T}_t$ the robot's tool's pose relative to its base reference frame, maintained by the robot's internal encoders. As noted earlier, the robot's base pose in the building reference frame, ${}^{b}\mathbf{T}_r$, is estimated through the marker-based metrology in equation (5.4).

5.3 Results and Discussion

5.3.1 Assembly Experiments

The designed algorithms were implemented into a robotic system using a 7-axis KUKA KR100 robotic arm with sub-millimeter level accuracy, a Point Grey Firefly MV camera, and a laptop

with an Intel i7 CPU, connected through the Robot Operating System (ROS)[17]. Each component in Figure 5-1 except for the assembly planner and robot manipulator is a process corresponding to a ROS node. The camera node sends images of size 1280 pixels by 960 pixels to the pose estimator that implements the Apriltag detection algorithm in C/C++. The plan achiever was implemented in Python. The robot controller was also developed using Python to send and receive control signals via Ethernet through KUKA's native Robot Sensor Interface (RSI). Inside this controller, a 6-DOF PID control algorithm was employed to drive the robot manipulator (with a two-finger gripper) to its destination pose when executing commands from the plan achiever. The involved inverse kinematics computations are performed inside the KUKA manipulator's controlling middleware.

In the first phase of experiments, the robot was tasked with assembling a section of a curved wall, as designed in section 5.2.3.1. The design was shown in Figure 5-1. The building components used were a set of 170x70x20mm3 medium-density fiberboard (MDF) blocks, each affixed with two different 56x56mm^2 Apriltags. The building reference frame was setup by 4 different 276x276mm^2 Apriltags. The overall goal of the experiment was to test the robot's ability to autonomously build the designed wall.

The system was first calibrated and validated using the methods discussed earlier. The validation residual calculated using equation (5.3) was found to be less than 1mm. With the accurately calibrated intrinsic and extrinsic parameters, the online building process proceeded smoothly. The building blocks were affixed with smaller markers, which decreased the building block localization accuracy to centimeter level (2cm) during the clasping process. The error was however compensated by the tolerance of the gripper.

[17] http://www.ros.org

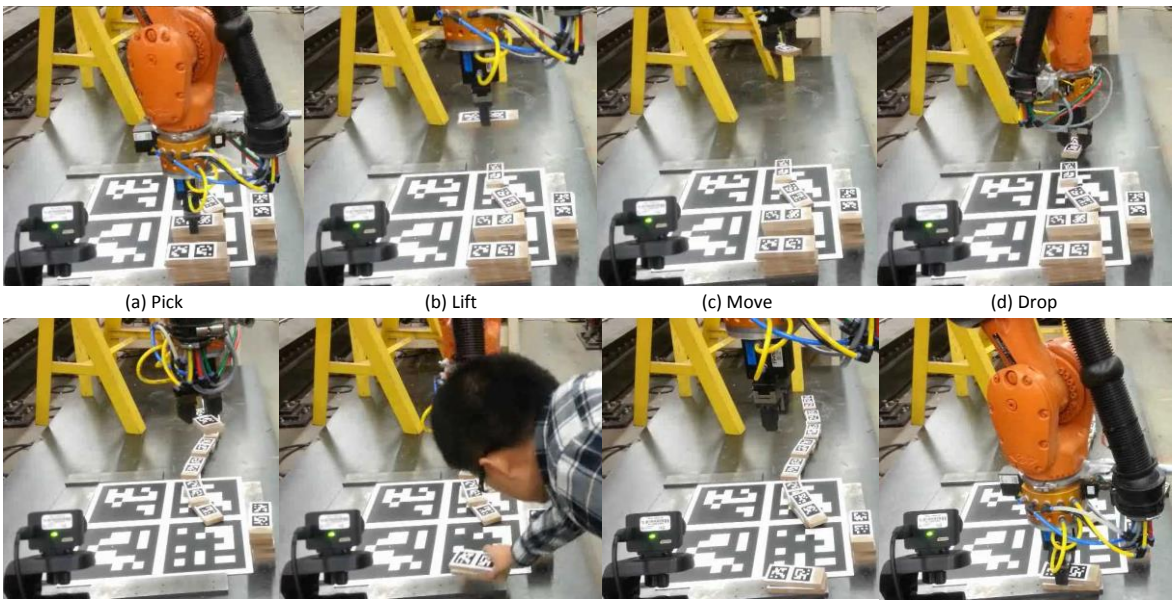


Figure 5-6: Autonomous robotic assembly experiments.

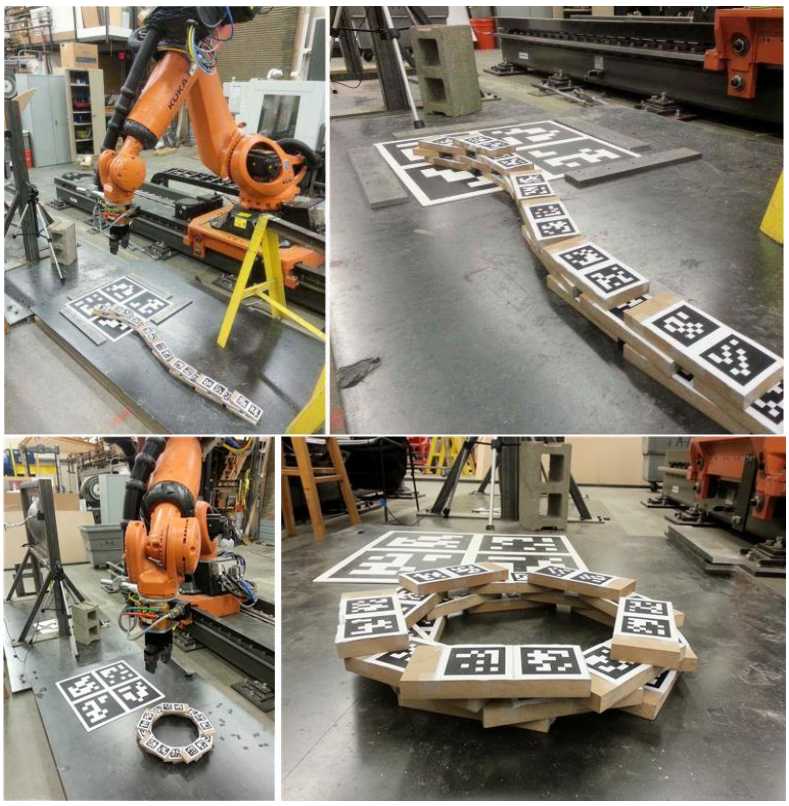


Figure 5-7: Curved and circular walls assembled onsite by the designed system.

A working cycle of the autonomous building process was shown in Figure 5-6(a)-(d). Since the pose estimator was constantly monitoring and updating the poses of each marker, the system was naturally capable of automatically adapting to pose changes on-site. As shown in Figure 5-6(e)-(h), when a building block's pose changed, the robot manipulator was automatically able to pick it up at its newest location. A video recording of the experiment can be found online at the following URL: http://youtu.be/fj7AXRpj97o. A fully assembled three-layer curved wall approximately 1.5m in length, and another three-layer circular wall, are shown in Figure 5-7.

5.3.2 Scanning Experiment

The scan module was implemented on the same KUKA robotic arm with a Microsoft Kinect as the 3D camera, as shown in Figure 5-8, with the scan planner and the scan register implemented in Matlab. The Fabrication Lab (FabLab) in the College of Architecture of the University of Michigan was scanned as an experiment demonstrating the scan module's effectiveness. Using the methods described in section 5.2.5, each frame of newly captured Kinect RGBD image of 640x480 pixels is first converted into a frame of colored point cloud and then transformed into the same building reference frame. The resulting final point cloud is then visualized in Point Cloud Library (PCL)[18], as shown in Figure 5-9.

5.3.3 Limitations and Future Work

There are several limitations in the above implementation of the proposed vision guided robotic assembly and scanning solution that need to be overcome to facilitate its future applications in real world construction sites. These limitations include:

[18] http://pointclouds.org

1. Feasibility and robustness of the marker-based pose estimation under different illumination conditions;
2. Occlusion of markers on construction sites may impede the feasibility or even performance of such systems;
3. Substantial effort may be necessary to setup such systems onsite due to requirements to survey numerous markers and register their poses into the project coordinate frame;
4. Additional effort is needed for attaching markers on each construction material/component such as blocks for the robot to recognize and pick them up;
5. The durability of markers is a critical limitation for applications in rugged environments and difficult weather conditions;
6. Lack of systematic and quantitative comparisons of such robotic system with traditional manual methods.

The first concern on marker-based pose estimation's robustness noted above has been addressed in section 6.4.1 of Chapter 6 with various field experiments proving the feasibility, robustness and accuracy for applying such technique in both indoor and outdoor construction environments.

Even though the second concern about occlusion is a common limitation of all vision based methods, it is not such a critical problem in this proposed method. This is because many occlusions of markers onsite are temporary due to moving equipment or humans. Once the mobile robotic manipulator observes those markers and estimates its own pose in the project reference frame, as long as its base stays static during the occlusion period, the robot can still accurately maintain its pose using its internal encoders.

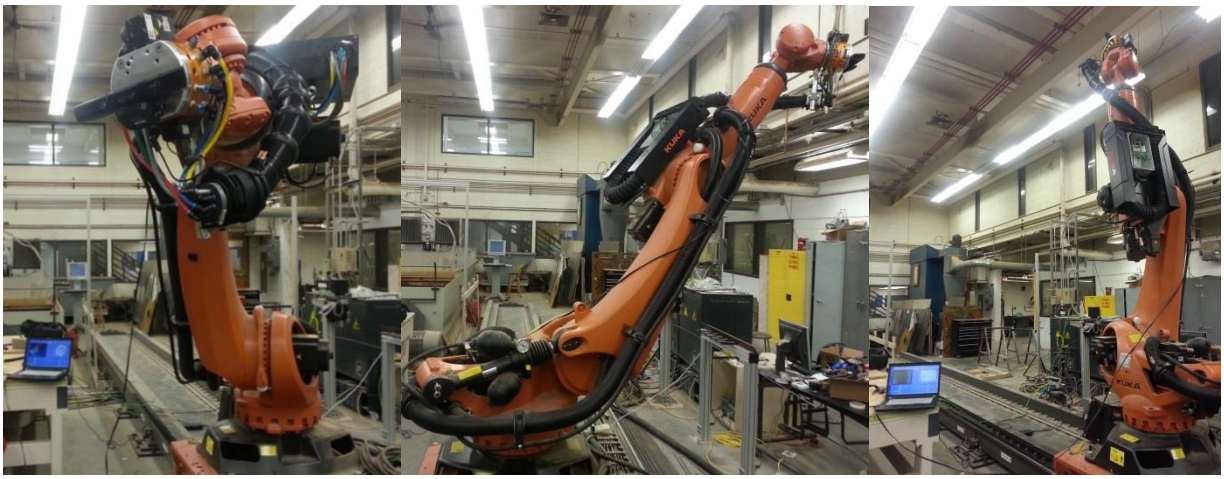


Figure 5-8: Scan module of the designed robotic system.

Figure 5-9: An as-built 3D point cloud of the FabLab at the University of Michigan

The third concern about the workload of installing and surveying markers onsite indeed exists in the current system implementation. Yet, compared to the efforts of setting up other pose estimation and localization methods that require powered hardware infrastructure such as WLAN or UWB, or methods that only work well in open-air outdoor environments such as GPS, this effort could be a worthwhile tradeoff. Moreover, Chapter 8 will describe a novel method that integrates the markers with Structure-from-Motion technique so that there will be no need to survey these markers using specialized surveying equipment such as total station. Instead, by simply taking a set of images of those installed markers, their poses can be automatically estimated.

The fourth and fifth concerns were also recognized, and approached by removing the requirement of attaching markers on raw construction materials. For example, although not closely related to the core contribution of this chapter, the block detection and grasp can be achieved without markers using Kinect sensor and the fast plane extraction algorithm in Chapter 2, and are demonstrated at the following URL: http://youtu.be/CyX4Pr_xly0. For markers to be attached on indoor construction sites, even printing on papers can achieve reasonably good durability, which is very cost-efficient and can be conveniently replaced if destroyed. For outdoor situations, markers can be spray painted on existing planar surfaces (e.g., walls) or manufactured wood, foam, or plastic boards.

Lastly, quantitative comparisons with tradition manual methods are a logical future direction in this research. Such comparisons will be more meaningful when this research advances to the next stage where the mobile manipulator can actually build stable structures with cement mortar, rather than in this research as the proof-of-concept stage.

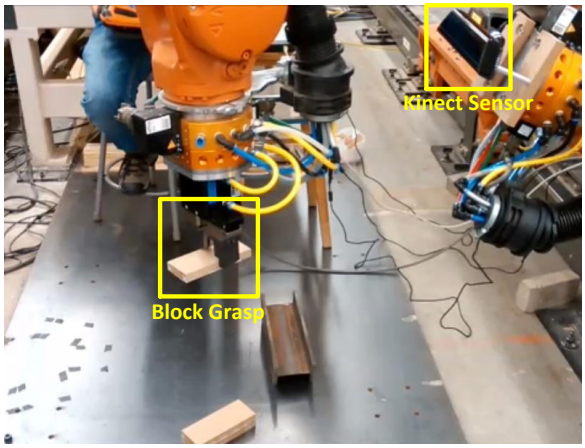


Figure 5-10: Block detection and grasp using Kinect.

5.4 Conclusions

This chapter reported algorithms and an implemented robotic system that is able to automatically generate assembly plans from computational architectural designs, achieve these plans autonomously on construction sites, and create as-built 3D point clouds. In order to address the localization accuracy challenge, this chapter proposed a computer-vision-based sub-centimeter-level metrology that enables pose estimation using planar markers. The conducted evaluation experiments used the designed robotic system to autonomously assemble various structures such as a curved wall of MDF blocks, proving the algorithms and the system's ability to meet the accuracy requirement when building computational architecture designs. In sum, with markers, an unstructured construction site can be rapidly configured to allow autonomous mobile manipulators to localize themselves and thus perform assembly and scanning tasks. In this way, the challenges of unstructured environment and mobility can be efficiently addressed.

In addition to the long term future goals mentioned in section 5.3.3, the current and planned work in this research direction in the short term is focused on continuously improving the fundamental methods used in this system along the following directions:

1. Perception: using camera on manipulator to further improve the pose estimation accuracy through multiple observations and sensor fusion; improving 3D perception to help improve the safety of the robotic system, preparing for its collaboration with human workers on-site.
2. Navigation: incorporating state-of-the-art SLAM algorithms with the proposed marker-based metrology in the system for increasing its range of autonomous movement.
3. Hardware and control: designing a suitable robotics platform with mobile base and on-board manipulator of sufficient payload capacity for indoor and outdoor construction activities that are more complex than the block laying activity described in this chapter, exploring more sophisticated control algorithms to enable such complex construction tasks.
4. Applications: extending the proposed autonomous system in other construction or architecture applications.

Chapter 6
Articulated Machine Pose Estimation for Excavation Monitoring

"What we need is a machine that will let us see the other guy's point of view."
—Arthur C. Clarke

6.1 Introduction

The construction industry has long been affected by high rates of workplace injuries and fatalities. According to the United States Bureau of Labor Statistics' 2013 Census of Fatal Occupational Injuries (CFOI) report (Bureau of Labor Statistics 2013), the construction industry had the largest number of fatal occupational injuries, and in terms of rate ranked the fourth highest among all industries.

In addition to the safety concerns, there are also increasing concerns of relatively stagnant productivity rates and skilled labor shortage in the construction industry. For example, recently the construction sector in the United Kingdom is reported to be in urgent need of 20% more skilled workers and thus 50% more training provision by 2017, to deliver projects in planning (LCCI/KPMG 2014).

Excavation is a typical construction activity affected by the safety and productivity concerns mentioned above. Excavator operators face two major challenges during excavation operations, described as follows.

First is *how to maintain precise grade control*. Currently, grade control is provided by employing grade-checkers to accompany excavators during appropriate operations. Grade-checkers specialize in surveying and frequently monitor the evolving grade profile. The evolving grade profile is compared to the target grade profile and this information is communicated by the grade-checker to the excavator operator. The operator reconciles this information and adjusts the digging strokes accordingly. This process is repeated until the target profiles are achieved. Employing grade-checkers is not only dangerous but also results in a significant loss in excavation productivity due to frequent interruptions required for surveying the evolving profile.

Second is *how to avoid collisions* to either human workers, buried utilities, or other facilities, especially when excavator operators cannot perceive the digging machine's position relative to hidden obstructions (i.e., workers or utilities) that it must avoid. According to the aforementioned CFOI report, among all the causes for the 796 fatal injuries in the U.S. construction industry in 2013, the cause of striking by object or equipment comprised 10 percent. This percentage is even higher in other industries such as agriculture (19%), forestry (63%), and mining (23%). Besides directly causing fatal injuries on jobsites, construction machines can also inadvertently strike buried utilities, thus disrupting life and commerce, and pose physical danger to workers, bystanders, and building occupants. Such underground strikes happen with an average frequency of about once per minute in the U.S., reported by the Common Ground Alliance[19], the nation's leading organization focused on excavation safety. More specifically, excavation damage is the third biggest cause of breakdowns in U.S. pipeline systems, accounting for about 17% of all incidents, leading to over 25 million annual utility interruptions (US DOT PHMSA 2015).

[19] http://www.commongroundalliance.com

Automation and robotics in construction (ARC) has been extensively promoted in the literature as a means of improving construction safety, productivity and mitigating skilled labor shortage, since it has the potential to relieve human workers from either repetitive or dangerous tasks and enable a safer collaboration and cooperation between construction machines and the surrounding human workers. In order to apply ARC and increase intelligence of construction machines to improve either safety or productivity for excavation and many other activities on construction jobsites, one of the fundamental requirements is the ability to automatically and accurately estimate the pose of an articulated machine (e.g., excavator or backhoe). The pose here includes the position and orientation of not only the machine base (e.g., tracks or wheels), but also each of its major articulated components (e.g., stick and bucket).

When a construction machine can continuously track its end-effector's pose on the jobsites, such information can be combined together with the digital design of a task, either to assist human operators to complete the task faster and more efficiently, or to eventually finish the task autonomously. For example, an intelligent excavator being able to track the pose of its bucket can guide its operator to dig trenches or backfill according to designed profiles more easily and accurately with automatic grade-check. This can eventually lead to fully autonomous construction machines. When construction machines becomes more intelligent, it can be expected to save time in training operators and thus to mitigate skilled labor shortage and also improve productivity.

On the other hand, when construction machines are aware of the poses of their components at any time and location on jobsites, combined with other abilities such as the recognition of human workers' poses and actions, such machines will be able to make decisions to avoid striking human workers, for example by sending alerts to their operators or even temporarily taking over

the controls to prevent accidents. Thus it will help to decrease the possibilities of those injuries and fatalities and improve the safety on construction jobsites. Similarly, with continuous tracking of the pose of its end-effector (e.g., a bucket of an excavator), an intelligent excavator could perform collision detection with an existing map of underground utilities and issue its operator a warning if the end-effector's distance to any buried utilities exceeds some predefined threshold.

Figure 6-1: Overview of SmartDig.

Thus, from a safety, productivity, and economic perspective, it is critical for such construction machines to be able to automatically and accurately estimate poses of any of their articulated components of interest. In this chapter, a computer vision based solution using planar markers is

proposed to enable such capability for a broad set of articulated machines that currently exist, but cannot track their own pose. A working prototype is implemented and shown to enable centimeter level excavator bucket depth tracking. Its overview is shown in Figure 6-1, with (A) camera cluster and stick marker, (B) benchmark with pre-surveyed pose in the project reference frame, (C) system calibration, (D) working prototype of automatic grade control, (E) comparison to manual grade.

The remainder of this chapter is organized as follows. Related work is reviewed in section 6.2. The technical approach is discussed next in detail in section 6.3. The experimental results are presented in section 6.4. Finally, in section 6.5, the conclusions are drawn and the future work is summarized.

6.2 Previous Work

The majority of the construction machines on the market do not have the ability to track their poses relative to some project coordinate frames of interest. To track and estimate the pose of an articulated machine, there are mainly four groups of methods.

First are the 2D video analysis methods, stimulated by the improvement in computer vision on object recognition and tracking. Static surveillance cameras were used to track the motion of a tower crane in (Yang et al. 2011) for activity understanding. Similarly in (Rezazadeh Azar and McCabe 2012) part based model was used to recognize excavators for productivity analysis. This type of methods generally require no retrofitting on the machine, but suffers from both possibilities of false or missed detection due to complex visual appearance on jobsites and the relative slow processing speed. Although real-time methods exist as in (Memarzadeh et al. 2012;

Brookshire 2014), they either cannot provide accurate 6D pose estimation, or require additional information such as a detailed 3D model of the machine.

Second are stereo vision based methods. A detailed 3D model of the articulated object was required in (Hel-Or and Werman 1994) in addition to stereo vision. A stereo camera was installed on the boom of a mining shovel to estimate pose of haul trucks in (Borthwick et al. 2009), yet the shovel's own pose was unknown. In (Lin et al. 2013) the shovel's swing rotation was recovered using stereo vision SLAM, yet the pose of its buckets was not estimated. This type of methods can be infrastructure independent if with SLAM, yet some problems (sensitivity to lighting changes or texture-less regions) remain to be resolved for more robust applications.

Third are laser based methods, e.g., (Duff 2006; Kashani et al. 2010; Cho and Gai 2014), which rooted from the extensive use of laser point clouds in robotics. This type of methods can yield good pose estimation accuracy if highly accurate dense 3D point clouds of the machine are observed using expensive and heavy laser scanners. Otherwise with low quality 2D scanners, only decimeter level accuracy was achieved (Kashani et al. 2010).

Finally are angular sensor based methods, such as (Ghassemi et al. 2002; Cheng and Oelmann 2010; Lee et al. 2012). They are usually infrastructure independent and light-weight, but the resulting pose estimation is either not accurate enough or sensitive to changes of magnetic environment which is not uncommon in construction sites and can lead to large variations in the final estimation of the object poses. Moreover this type of methods only estimate the articulated machine's pose relative to the machine base itself, if without the help of infrastructure dependent sensors like GPS. However the use of GPS brings several technical challenges. For example, construction sites in a lot of cases do not have good GPS signals to provide accurate position

estimation when these sites are located in urban regions or occluded by other civil infrastructure such as under bridges. Sometimes GPS signals could even be blocked by surrounding buildings on jobsites and thus fail to provide any position estimation. In addition, since the GPS only provides 3D position estimation, to get the 3D orientation estimation one needs at least two GPS receivers at different locations of a rigid object. When the object is small, such as a mini-excavator's bucket, the estimated 3D orientation's uncertainty will be high.

6.3 Technical Approach

In this section, firstly the basic marker based pose estimation for this application is discussed as a baseline design. After potential issues of the baseline design are identified, different versions of the proposed articulated machine pose estimation system design resulting from the previously developed camera marker networks are explained. Finally, two prototypes implementing these designs are discussed.

6.3.1 Baseline Design

As mentioned previously, this computer vision based articulated machine pose estimation solution relies on a method called marker based pose estimation. Generally, marker based pose estimation firstly finds a set of 2D geometry features (e.g., points or lines) on an image captured by a calibrated camera, then establishes correspondences between another set of 2D or 3D geometry features on a marker whose pose is known with respect to a certain coordinate frame of interest, and finally estimates the pose of the camera in that coordinate system. If 2D-2D correspondences are used, the pose is typically estimated by homography decomposition. If 2D-3D, the pose is typically estimated by solving the perspective-n-point (PnP) problem. Two typical marker-based pose estimation methods are AprilTag (Olson 2011) and KEG (Feng and Kamat 2013) proposed in Chapter 3.

There are two ways of applying marker based pose estimation for poses of general objects of interest. As shown in Figure 6-2, one way is to install the calibrated camera 1 rigidly on the object of interest (in this case, the cabin of the excavator), and pre-survey the marker 1's pose in the project coordinate frame. The other way is to install the marker 2 rigidly on the object (in this case, the stick of the excavator), and pre-calibrate the camera 2's pose in the project coordinate frame. As long as the camera 2 (or the marker 1) stays static in the project coordinate frame, the pose of the excavator's stick (or the cabin) can be estimated in real-time.

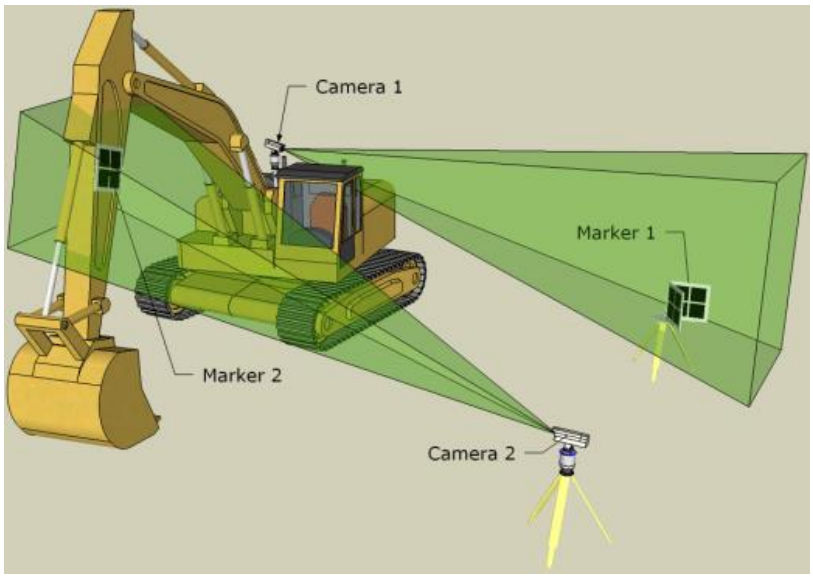


Figure 6-2: Two examples of basic camera marker configuration

However, these basic configurations don't always satisfy application requirements. For example, if only the camera 1 and the marker 1 are used, the excavator's stick pose cannot be estimated. On the other hand when only the camera 2 and the marker 2 are used, once the stick leaves the field of view (FOV) of the camera 2, the stick's pose becomes unavailable as well. Thus it is necessary to take a camera's FOV into consideration when designing an articulated machine pose

estimation system. This understanding leads to the camera marker network designs proposed as described in the following subsections.

6.3.2 Camera Marker Network Designs

As proposed in Chapter 4, a camera marker network is an observation system containing multiple cameras or markers for estimating poses of objects embedded in this system. It can be abstracted as a graph with three types of nodes and two types of edges. A node denotes an object (i.e. the local coordinate frame of that object), which can be a camera, a marker, or the world coordinate frame. An edge denotes the relative pose (i.e. transformation) between two objects connected by this edge, which can be either an observed pose estimated from the previously mentioned marker based pose estimation, or a known pose (e.g., through calibration). Thus, if at least one path exists between any two nodes in such a graph, the relative pose between them can be estimated. In addition, any loop in the graph means a constraint of poses that can be used to improve the pose estimation.

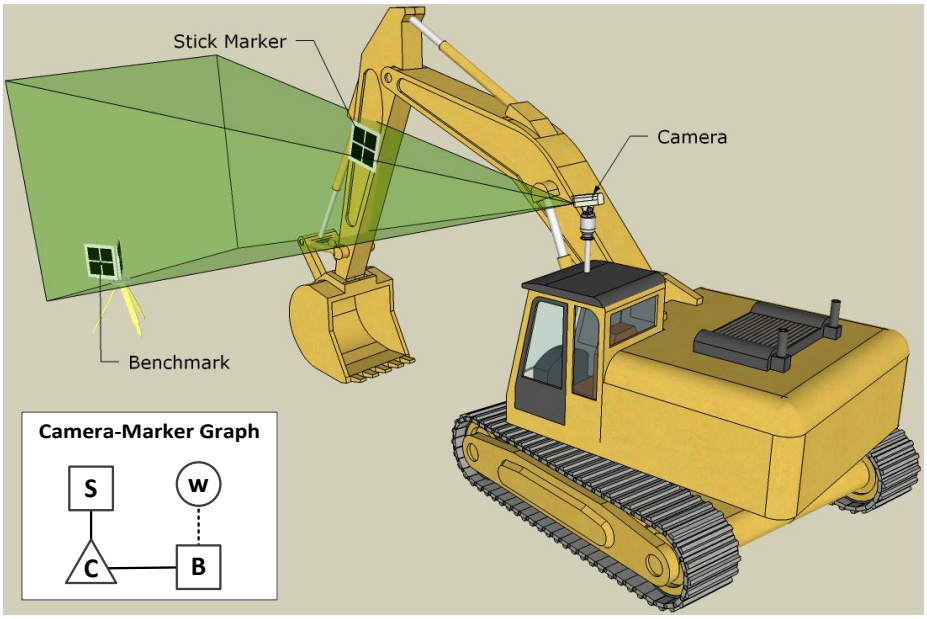


Figure 6-3: Single-camera multiple-marker configuration.

Applying this concept to articulated machine pose estimation results in numerous possible designs. One of the basic possible camera marker networks is shown in Figure 6-3. A single camera is rigidly mounted on the base of the machine observing a marker attached to the articulated component of interest (e.g., stick). In the meantime another benchmark marker with pre-surveyed pose is also inside the camera's FOV. Then the stick's pose in the world frame can be continuously estimated even if the machine base might be moving. This configuration is typically suitable for applications where the moving direction is aligned with the camera's viewing direction (e.g. trenching).

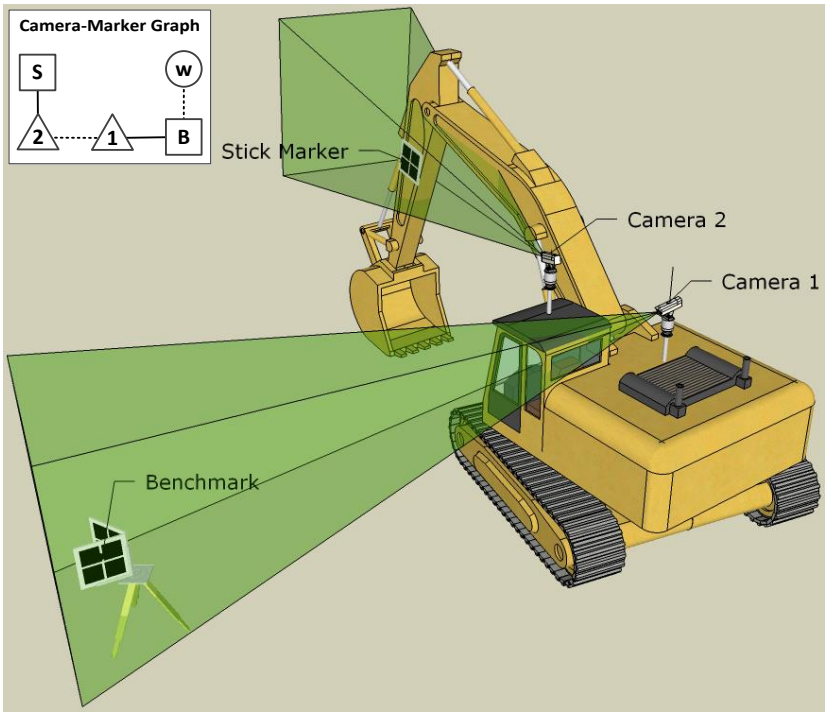


Figure 6-4: Multiple-camera multiple-marker configuration.

Such single-camera multiple-marker configuration can be further extended by adding more cameras. For example as shown in Figure 6-4, camera 1 observes the benchmark while camera 2 observes the stick marker, and the rigid transformation between the two cameras is pre-

calibrated. Thus as long as the two markers stay inside the two cameras' FOV respectively, the stick's pose in the world frame can be estimated. It is worth noting that the two figures only illustrate the simplest configurations. With more cameras and markers in the network, there are more chances of creating loops and thus improving pose estimation, especially considering that surveillance cameras are becoming popular in construction jobsites whose poses can be pre-calibrated.

6.3.3 Prototypes

Multiple prototypes have been implemented to realize the above described camera marker network designs. Figure 6-5 demonstrates one of the early prototypes implementing the single-camera multiple-marker configuration. A mechanical component using a timing belt was adopted to map the relative rotation between the excavator bucket and the stick to the relative rotation between the stick marker and the flip marker. This implementation enables pose tracking of the excavator bucket.

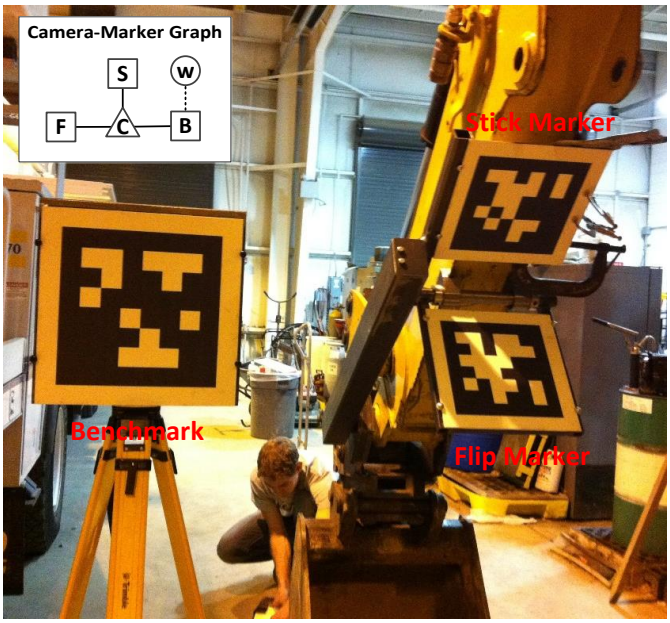


Figure 6-5: An early prototype configuration.

Due to the potential interference of the flip marker and any obstructions during excavation, the above early prototype was slightly modified and evolved to the current prototype as shown in Figure 6-1. The newer working prototype implements the multiple-camera multiple-marker configuration similar to Figure 6-4. Two cameras are rigidly mounted forming a camera cluster. A linear potentiometer is installed on the stick to track the relative motion of the excavator bucket and the stick even if the bucket is deep inside the earth.

6.4 Experimental Results

6.4.1 Feasibility Experiments

Before implementing the pose estimation system prototypes, a set of experiments were performed to test the feasibility of marker based pose estimation in different indoor/outdoor construction environments. In all the experiments, AprilTag (Olson 2011) was chosen as the basic marker detection and tracking algorithm.

Firstly, the outdoor detectability of markers was tested. A marker's detectability is a function of many factors including the marker size, the distance between the marker and the camera, included angle between the camera viewing direction and the marker plane's normal direction, and also image resolution. Since the distance between the marker and the camera is the most critical factor affecting the method's feasibility in real applications, this experiment is performed by fixing other factors and then gradually increasing the distance of the marker in front of the camera, until the algorithm fails to detect the marker, and recording the distance. Varying other factors and repeating this process results in Table 6–1. One can consult this table to decide how large the marker should be to fit application need.

Table 6–1: Outdoor detectability of AprilTag.

Max Detectable Distance (m)		Marker Angle (degree)			
		0	45	0	45
Marker Size (m^2)	0.2 x 0.2	6.10	4.88	11.28	8.84
	0.3 x 0.3	8.23	7.01	14.94	11.58
	0.46 x 0.46	13.41	11.28	25.91	21.64
	0.6 x 0.6	19.51	16.46	34.44	30.48
Image Resolution		640 x 480		1280 x 960	
Focal Length		850 pixels		1731 pixels	
Processing Rate		20 Hz		5 Hz	

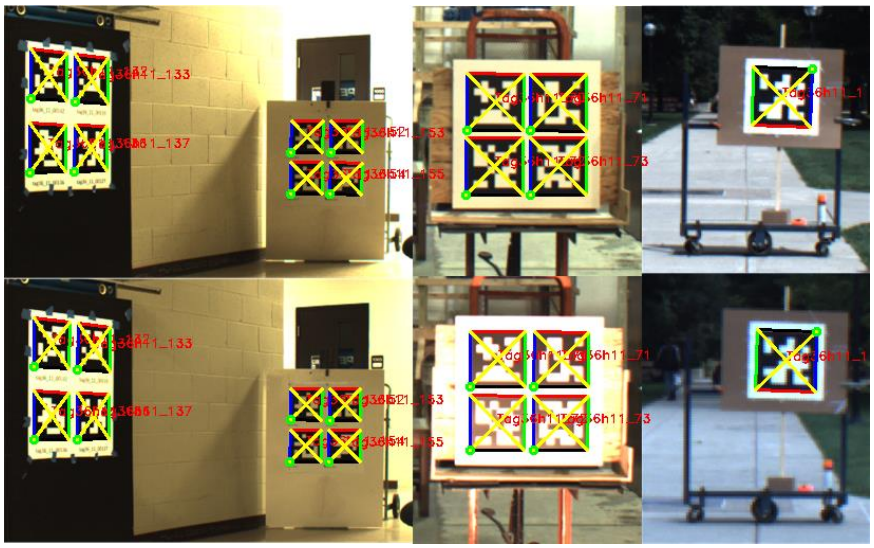


Figure 6-6: Marker detection vs illumination.

Secondly, illumination is a critical factor affecting performance of many computer vision algorithms. The AprilTag algorithm was thus tested under various illumination conditions to examine its robustness for construction applications. Figure 6-6 shows successful marker detection under different indoor/outdoor lighting conditions. These experiments and following extensive prototype tests proved AprilTag based marker detection method's robustness to illumination changes.

Finally, for uncertainty propagation, one needs to have a prior estimation of the image measurement noise's standard deviation σ_u. This is achieved by collecting multiple images under a static camera marker pose. Repeating this process for different poses and collecting corresponding image measurement statistics lead to an image measurement covariance matrix $\mathbf{\Sigma}_u$, which can be further relaxed to $\sigma_u^2\mathbf{I}$ to include all the data points. Figure 6-7 shows that $\sigma_u = 0.2$ pixel (adopted in section 4.4.1 on page 95) is reasonable.

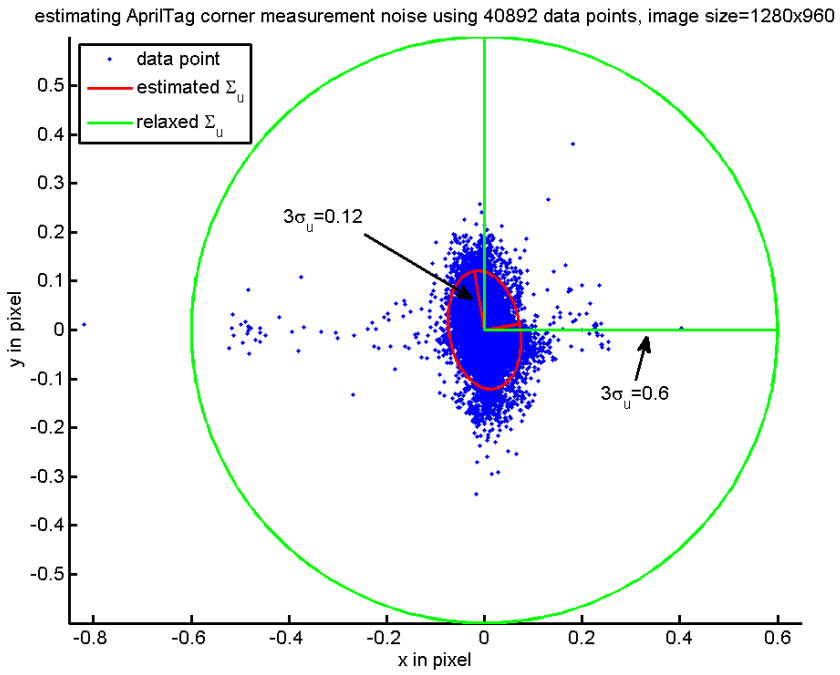
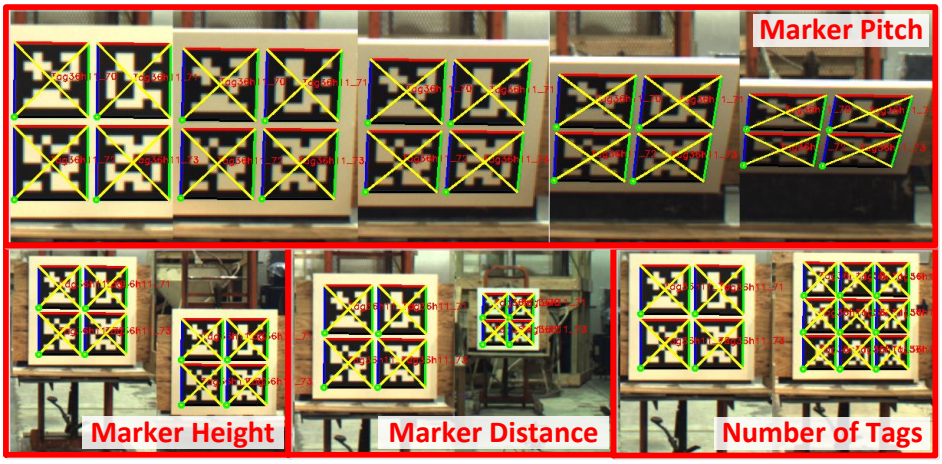


Figure 6-7: Image measurement noise estimation.

6.4.2 Prototype Experiments

As previously mentioned, a multiple-camera multiple-marker articulated machine pose estimation prototype has been implemented with the application of estimating an excavator's bucket depth in a project frame, which could be used for automatic excavation guidance and grade control.

(a) Setup



(b) Different configurations

Figure 6-8: Prototype experiments.

The top row of Figure 6-8(a) shows the camera cluster of the prototype in Figure 6-1, and different experiment configurations to test the depth estimate's accuracy. The experiments were setup by observing the two markers in the bottom row of Figure 6-8(a) using the two cameras in the cluster respectively. Then the depth difference between the two markers was estimated using the proposed method, while the ground truth depth difference between the two marker centers was measured by a total station with 1 mm accuracy. Figure 6-8(b) illustrate the configurations of different sets of such experiments, for comprehensive tests of the method's accuracy under

several system and design variations. The first set varies one of the marker's pitch angle (top row of the figure). The second set varies its height (bottom-left). The third set varies its distance to the camera (bottom-middle). And the fourth set varies the number of Apriltags used in that marker (bottom-right).

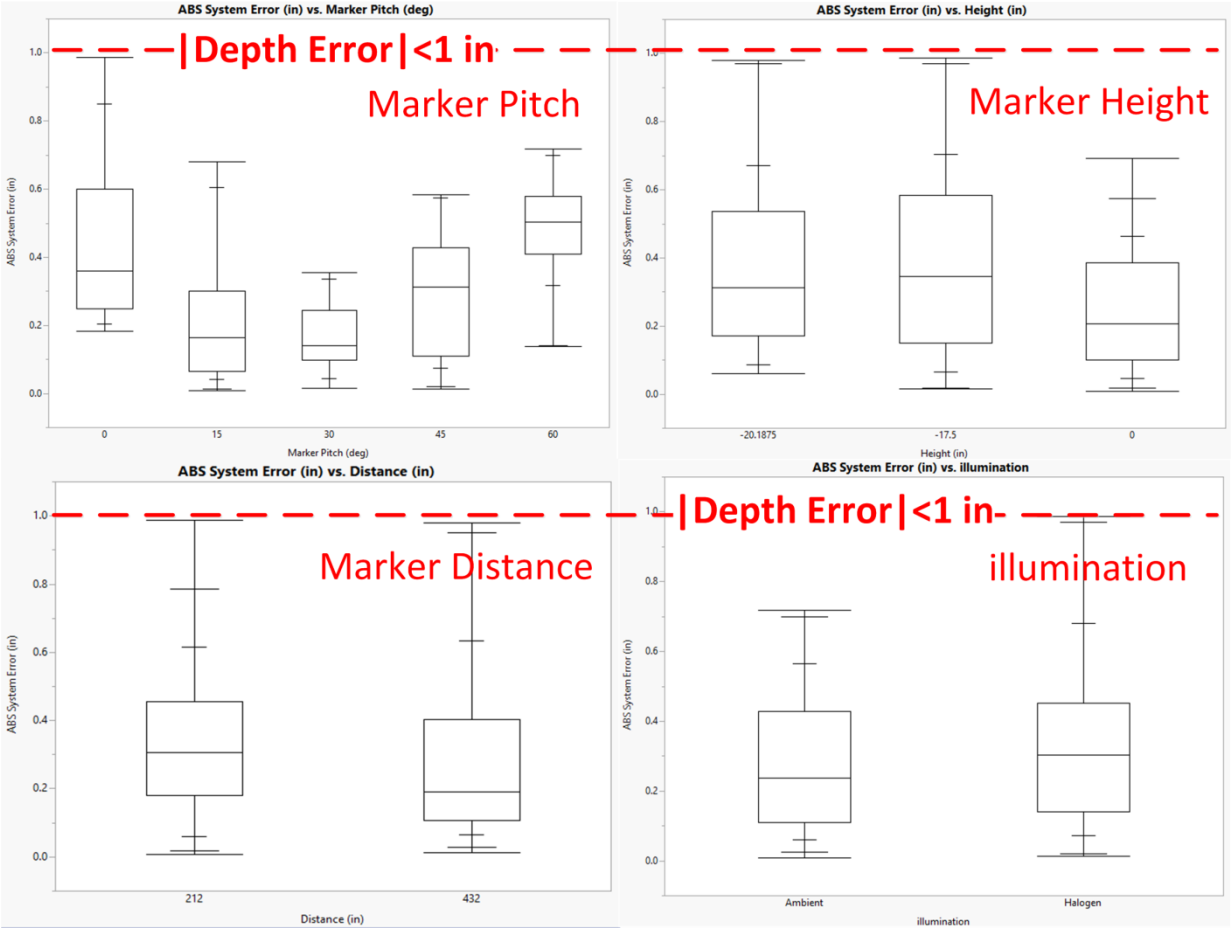


Figure 6-9: Prototype error vs. configuration.

Figure 6-9 shows the box plot of the absolute depth errors comparing the ground truths with the results from camera marker network pose estimation, in the above mentioned different sets of prototype experiments. Note that **all errors are less than 2.54 cm, even when observed from**

more than 10 meters away. Further experiments showed that the system worked up to 15 meters.

This prototype is then tested on a real excavation site for grade control as shown in Figure 6-1 (D) and (E). The resulting trench depth differences between the manual grade and the guided grade (by the prototype) are less than 1 inch, which fulfils the needs of many construction applications.

6.5 Conclusions

This chapter proposed a vision based pose estimation solution for articulated machines using previously proposed camera marker networks. The conducted experiments and a working prototype proved the proposed solution's feasibility, robustness, and accuracy (centimeter level) for real world construction applications.

The current and planned work in this research direction is focused on continuously improving the estimation accuracy, and increase the scale of the application from currently 10~20m to 50~100m by adopting different types of lenses (e.g., telescopic lens). This improvement will be in collaboration with the future research directions of general camera marker networks, so that different choices of network designs and lens types can be analyzed in advance to real experiments. Another future research direction is to incorporate with the state-of-the-art visual SLAM algorithms so that the application scale can be further increased in more flexible ways.

Part III: Applications in Construction Automation with Human-in-the-Loop

"Can machines think?"—Alan Turing

This part includes two construction automation applications with human-in-the-loop, also based on algorithms from part I, guided by the Human Machine Interaction principle introduced in Chapter 1, section 1.2.2.2, meaning that the collaboration between human and intelligent machines was rooted in the designs.

Americans spend "approximately 90 percent of their time indoors" on average (U.S. Environmental Protection Agency 1989). How to effectively link indoor physical locations to associated information so as to facilitate the work and life of humans living inside becomes critical. Indoor facility management is a typical automation application related to this. Chapter 7 described a possible topological navigation solution using marker networks and mobile BIM.

Due to inevitable differences between designs and as-built introduced in section 1.2.2.1, more and more construction contractors are required to provide owners as-built BIM models. A most important part of as-built BIM model is the 3D parametric geometry model of the building. Chapter 8 described a solution using camera marker networks and fast plane extraction together to efficiently scan the building and create a planar parametric model to accelerate further as-built BIM model generation.

Chapter 7
Markers as Spatial Indices for Indoor Mobile Facility Management

"It is impossible to underrate human intelligence - beginning with one's own."
—Henry Adams

7.1 Introduction

Context-aware information delivery has been recognized as a critical component in many Architecture, Engineering and Construction, and Facilities Management (AECFM) applications (Aziz et al. 2005; Khoury and Kamat 2009; Andoh et al. 2012). Identification of a user's location is one of the most fundamental and extensively studied problems in this area. Solutions that achieve good balance between cost and accuracy could lead to meaningful productivity improvement in applications such as facility management, construction inspection, and indoor way finding.

Previously, researchers' attention has focused more on traditional non-visual-sensor-based methods, such as Global Positioning System (GPS), Inertial Measurement Unit (IMU) (Akula et al. 2011), Radio Frequency Identification (RFID) (Sanpechuda and Kovavisaruch 2008; Andoh et al. 2012), Wireless Local Area Network (WLAN) (Aziz et al. 2005), and Ultra-Wide Band (UWB) (Teizer et al. 2008). Recently computer vision and robotics communities have proposed methods such as Simultaneous Localization and Mapping (SLAM) (Thrun 2008) and Visual

Registration (Olson 2011), utilizing visual sensors such as camera or lidar. Although most of these methods can track a user's 3D position continuously, they also have their own disadvantages respectively such as indoor unavailability, tracking drift, large infrastructure/special hardware requirement, cost inefficiency, and high computational power requirement.

However, in many of those methods' AEC domain applications, the fact that the end-user is human being makes human intelligence not only a non-negligible but rather important ingredient to achieve a good cost-accuracy balance (Akula et al. 2011). This is different from robot navigation since robots need to know their own location at any moment for further decision making, while a human already possesses the ability of maintaining its own location within a certain range. For example, given a typical hallway with limited turnings, a human can easily navigate from one end point to the other; while a robot may need help of continuous SLAM. Only when the hallway contains lots of turnings and exit stairs, navigation becomes useful for humans, which is also true for outdoor road navigation. This observation in fact suggests that continuous tracking of user's position might not be necessary in many AEC applications; for human inspectors, to automatically extract discrete-spatial-distributed information could be sufficient to accomplish their jobs faster and better.

Naturally, marker based visual registration method for Augmented Reality (AR) becomes a good candidate for such discrete localization (Olson 2011; Feng and Kamat 2013). Registration problem means how to find the relative position and orientation between two coordinate systems in AR (Azuma 1997). Thus, beside correspondences between different AR markers and discrete locations, these types of methods also provide estimation of not only position but also orientation. This additional dimension of information could help to obtain better visualization of

extracted information and hence extend the application domain to where traditional sensor based methods such as RFID could not reach.

In the meantime, mobile devices are becoming more and more popular recently. Especially for smart phones and tablets, camera, CPU and even GPU are standard configuration, which unsurprisingly makes them ideal equipment for ubiquitous visual computing. Combining those observations, the authors were motivated to explore a new method utilizing AR markers as spatial indices to create links between physical locations and virtual information stored in databases, which runs on mobile devices. The contribution in this chapter is mainly a general computing framework as well as an indoor way-finding application based on such a framework. The fact that more and more people have smart devices makes it easier for people to access information using this method than all previous methods such as RFID and UWB.

The remainder of this chapter is structured as follows. In section 7.2, previous work on both traditional and visual sensor based methods is reviewed; in section 7.3, the general computing framework of AR marker as spatial index is explained in detail; in section 7.4 an indoor way finding application based on that framework is developed to facility indoor facility management; section 7.5 describes a set of experiments conducted to prove the new indoor way finding method's efficiency; and finally section 7.6 presents the conclusions of the chapter.

7.2 Previous Work

As mentioned before, researchers have extensively studied two types of localization techniques for context aware computing, i.e. traditional non-visual-sensor-based methods, as well as newly emerging visual-sensor-based methods.

Among the first type, GPS is mainly used for outdoor scenarios; IMU's tracking drift issue requires error correction either by human (Akula et al. 2011) or by combining with other methods such as AR marker (Feng and Kamat 2013); RFID-based methods usually depend on large infrastructure (i.e. enough RFID tags must be available) and also requires special tag reader which is not easily accessible by common people (Sanpechuda and Kovavisaruch 2008; Andoh et al. 2012); WLAN-based methods also require large number of footprints (Aziz et al. 2005); UWB-based methods generally cost too much (Teizer et al. 2008; Khoury and Kamat 2009). Besides, none of these methods can easily provide instantaneous orientation information (even though there is angular sensors such as gyroscope, electrical compass or accelerometer, they themselves come with problems such as noise and sensitivity to the environment), which makes them not optimal for further 3D visualization purpose.

On the contrary, the second type of methods directly output orientation along with position information. By analyzing images captured from visual-sensors such as a simple webcam, the visual registration methods can recover that sensor's pose (position and orientation). Based on their different assumptions on the environment (i.e. the surrounding world where visual registration is going to be performed), these algorithms can be classified into two groups (Lepetit and Fua 2005; Feng and Kamat 2013): known environment vs. unknown environment (see Figure 3-1 on page 47), the proposed indoor way-finding application contains a module that can be classified as fiducial marker-based method).

The first group of visual registration algorithms is less computation-consuming and easier to design since the only unknown is the user's pose. Because they have been well studied, and many related powerful algorithms have been proposed in the last two decades, it is more realistic to apply them for solving real-world engineering issues. Within this class of methods, they can

be further categorized into two groups: planar environment vs. non-planar environment. The first group is again easier to design because of the simple assumption made regarding the environment—a planar structure with known visual features. The second group is more often applied in a controlled environment with limited space, such as a small manufacturing workspace.

In the indoor way-finding applications, the authors chose to take advantage of plane-based methods since planar structures are abundant in buildings, construction sites, and other built environments where engineering operations are conducted, which makes this type of method very convenient to apply. In addition, a planar structure can simply be an image printed out on a piece of paper and attached to a wall/floor of a corridor, with negligible cost.

Plane-based methods can be further classified based on different visual features they adopt: fiducial marker vs. natural marker. A fiducial marker is composed of a set of visual features that are "easy to extract" and "provide reliable, easy to exploit measurements for the pose estimation" (Lepetit and Fua 2005). Usually those features are a set of black and white patterns forming simple geometry by circles, straight lines, or sharp corners and edges. Well known fiducial markers include ARToolKit (Kato and Billinghurst 1999) and the newly proposed AprilTag (Olson 2011).

Distinct from a fiducial marker, a natural marker does not require special predefined visual features. Instead, it treats any visual features in the same way. In this sense, almost any common image, ranging from a natural view to a company logo, can immediately be used as a natural marker. Even though natural marker methods have advantages of more accurate and stable as well as larger tolerance to partial occlusion, its relatively higher computational requirement than

fiducial marker methods limits its sphere of application to high-end smart mobile devices. Taking all these factors into consideration, the authors chose to apply fiducial markers in the indoor way finding application so as to lower the requirement of its targeted devices.

7.3 Methodology

In this section, a general computing framework is described, representing the authors' proposed idea to use AR marker as spatial index which links between physical location and virtual information related to that location. Since the framework is generic, the descriptions are not fixed to any specific algorithms. As long as an algorithm can meet the requirements of each module to be described below, it can fit into the framework. This makes it possible to adapt new algorithms for existing system with little difficulties.

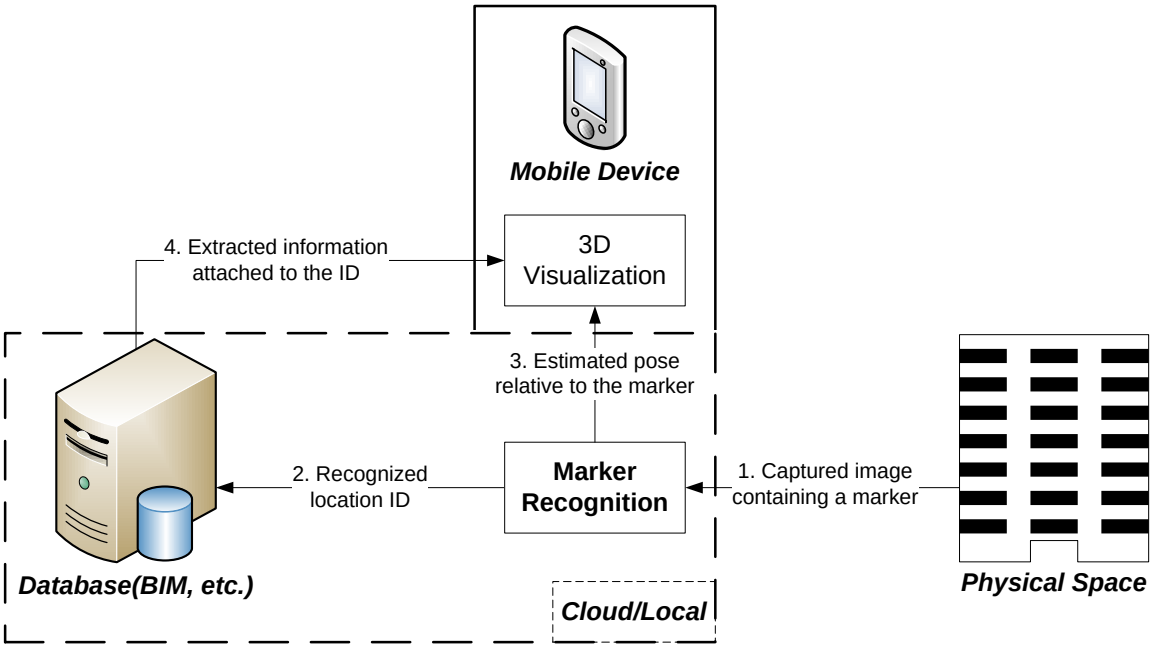


Figure 7-1: System overview of AR marker as spatial index.

The system overview is shown in Figure 7-1. The system operates based on the following procedure. Firstly, an image potentially containing an AR marker is captured by the camera on

the mobile device. This image is then sent to the marker recognition module. If the image contains an AR marker and it is recognized by the marker recognition module, then the ID of the AR marker is sent to the database as a key value to search for attached information. Simultaneously, the estimated pose of the mobile device relative to the marker is sent to the 3D visualization module. This visualization module then takes the estimated pose to render the information sent back from the database on the screen of the mobile device for further decision making by its end-user. In order to achieve such functionalities, each module needs to conform to certain requirements, which are explained in more detail below.

7.3.1 Physical Space

Generally speaking, the physical space in this framework, as the targeted environment of such a system, could be any indoor scene. For example, a complex shopping mall, an international airport, a subway station, a big warehouse or a building construction site. In fact, in some application scenarios, even outdoor scene such as a public park could be equipped with such a system.

In order to make the original physical space compliant with the system, AR markers must be attached to a discrete set of locations which are critical to the application. The AR markers should have their own IDs which serve as the spatial indices of that set of locations. The marker recognition algorithm corresponding to those markers should contain two functions: 1) the ability to extract the ID of a marker presents in an image; 2) the ability to recover the pose of that image, i.e. the mobile device, relative to the marker. Some possible choices of marker could be AprilTag (Olson 2011) or KEG marker (Feng and Kamat 2013).

Even though the requirement of one-to-one mapping between AR markers and physical locations makes this approach slightly infrastructure dependent, the facts that AR marker is flexible and easy to use/install, cost efficient, and more importantly capable of providing orientation information make it more ideal than RFID tag.

7.3.2 Mobile Device

A mobile device is the core of such a system. Its camera serves as the main input where information enters into the system. While cameras are part of the standard configuration of common mobile devices such as mobile phone or tablet, a rear facing camera setup is more desirable than a front facing one, since the direction of sight of the camera could be aligned with user's viewing direction more naturally so that user could see the 3D visualization more conveniently.

Also, the 3D visualization module naturally locates within the mobile devices. It visualizes the information sent from the database in the means of augmented reality by pose estimated from the marker recognition module. Here an implementation choice must be made for any application conforming to this framework: whether the marker recognition module should locate locally within mobile device or remotely in the server/cloud. This usually depends on the invoking frequency of this computation framework. For some applications, the whole procedure needs to be performed for every frame of the live video stream captured by mobile device's camera. In this situation, a local marker recognition module is more reasonable, otherwise the captured image needs to be transferred to the server/cloud side very frequently, which will increase the data volume transferred through the mobile network resulting in higher cost and longer latency. For some other applications, the invoking frequency is relatively low, say around 1 time per

minute. Then turning the marker registration module into an online service could be a good choice.

7.3.3 Database

A database is the foundation of such a system. With physical locations mapped into IDs through AR markers, the database module is very flexible and can be implemented from almost all kinds of databases available, such as MySQL, SQLite or even a self-defined text file. Similarly, the storage location of this database is also an implementation choice, depends on the size of the whole database as well as the size of information to be sent back to the 3D visualization module. For relatively smaller-size database, say less than 1 Mb, downloading the whole database from cloud/server side to mobile device once and for all is a reasonable choice. Otherwise, server-side storage of the whole database could be more efficient, as long as the size of the information to be sent back in step 4 of Figure 7-1 is relatively small.

7.4 Way-finding Application for Indoor Facility Management

As mentioned previously, the above methodology is best suitable for the decision making process which involves a set of discrete spatial locations. Indoor way finding is one good example application for which the above methodology could be very helpful.

Consider the following first-person scenario: you are a new student on the first day of class. You were given a room number which is the first class you are going to take. But the building is large and unfamiliar, and even though you entered it for a few times previously, you still find it difficult to find the correct direction. You try to look for an indoor floor plan view of the building but it takes you more than 5 minutes and you fail to understand the map. Since it is already time for class, very few people are available for inquiring the directions. When you

finally find someone for help, the instructions you get are "go along this hallway and turn left at room 1318, then walk to the end of the hallway and turn right. Then pass the exit door and go upstairs to second floor and turn right again. After another two exit doors and a right turn is your destination." After you find the room probably half of the class has passed. Similar situations happen frequently inside many complex buildings for a person who is unfamiliar with the environment or doesn't have a good sense of direction, and even for someone who works/studies inside the building every day since s/he has never been to the room before.

Figure 7-2: Example of mobile BIM for facility management[20].

Similar situations exist for facility managers or inspectors using mobile BIM applications (Figure 7-2). A facility manager needs to periodically access and update the maintenance information of a specific facility at a certain location. To access BIM on a mobile device (dubbed as mobile BIM, which extends BIM's applications from pre-construction to post-construction phase) for facility managers immediately when they are inspecting those facilities can greatly simplify their workflows and improve their work efficiency. However the BIM model of a project can be very complex, while existing mobile BIM implementations cannot automatically determine a user's

[20] Images come from Walter P Moore: http://www.walterpmoore.com/

location inside any indoor environment. Thus facility managers have to manually search and locate the object on mobile devices with relatively small screens, which is inconvenient and inefficient.

In both of the above two cases one can see that the most useful piece of information—either the instruction which greatly influences the decision making process of way finding, or the facility location in a BIM model which is necessary for accessing and updating corresponding data—contains a set of discrete locations (e.g. room 1318, exit door, corner, stair, etc., or a HVAC equipment's position) as described before. It is then natural to consider that an AR marker should be attached to each of such critical spatial locations to map these locations into spatial indices. In the database module, the position and orientation of each marker under the building's global coordinate system is stored, as well as positions of all the rooms of the building. Thus an oriented graph is generated from this information, with nodes as rooms and markers, edges as instructions of how to move from starting node to ending node, edge weights as lengths of physical paths (see Figure 7-3, shaded large circle means marker node, small circle means room node). There are two situations for adding edge to the graph:

1. When there is a physical path between a room node A and a marker node B without passing other markers, an edge from B to A can be added to the graph; No edge from room node to marker node should be added;
2. When there is a physical path between a marker node C and a marker node D without passing other markers, an edge from C to D, as well as an edge from D to C, should be added to the graph.

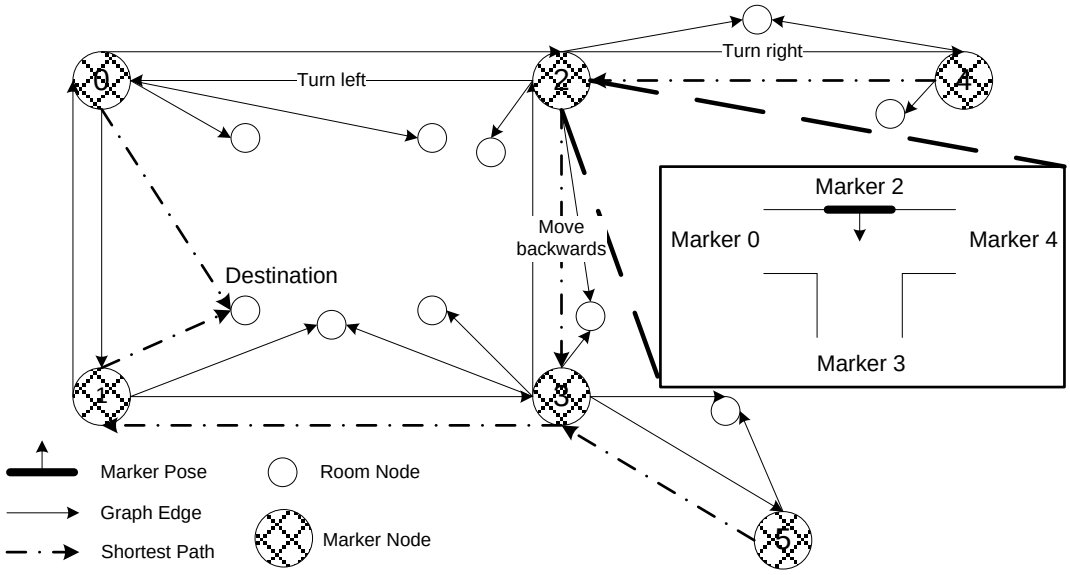


Figure 7-3: An example of oriented graph for indoor way finding application.

The instruction stored within each edge, say edge starting from node A to node B, is the method of how to move from A to B assuming user is currently facing towards the marker. For example, as the orientation of marker 2 shown as an arrow in the detailed floor plan view of marker 2 in Figure 7-3, instruction stored in edge from marker 2 to marker 0 could be "turn left", since when the user gets this instruction s/he should be looking at this marker 2.

After the graph is constructed, the user is asked to select a node as destination. Then the Dijkstra algorithm (Dijkstra 1959) is performed on the graph to compute the shortest path from all other marker nodes to this node. Thus whenever user comes to a marker, the marker recognition module will extract the corresponding spatial index and then find the marker node in the graph. Then the shortest-path edge starting from this node is retrieved and the moving instruction is displayed graphically in the 3D visualization module (see Figure 7-4). When user is standing at the marker node which has a direct edge pointing to the destination, special logo is shown (the

animated eye shown at the top-right corner of the left image in Figure 7-4) to remind user to slow down and look carefully around the surroundings since s/he is very close to the destination room.

Figure 7-4: Screenshots showing graphical instruction of how to move to the destination.

The authors implemented the described indoor way finding application on Android platform. Here the two design choices are made as follows. Marker recognition module is located in the mobile device, since real-time performance is desired; database is stored in server side as a simple self-defined text file describing the whole graph of a building (termed as a map file). This application assumes that user is aware of which building s/he is in. When user selects the building, the corresponding map file will be downloaded to the mobile device. The marker recognition module in this application adopts the AprilTag. As mentioned in section 7.2, the reason to adopt this specific algorithm is that AprilTag is proved to be superior than previous proposed fiducial markers in the sense of speed, accuracy and tolerance to critical view conditions (long distance, large viewing angle, partial occlusion etc.).

The authors have made this application, named as Mobile AR Navigator (MARvigator)[21]. Although similar methods or systems have been proposed before (Kalkusch et al. 2002; Wagner and Schmalstieg 2003; Augmented Reality & Assistive Technology Laboratory of NUS 2011),

[21] publicly available online at http://www.umich.edu/~cforrest/upload/marvigator/

MARvigator is superior since it takes advantage of the state-of-the-art fiducial marker system and is implemented conforming to the general framework described previously, resulting in faster real-time performance (about 15 frame-per-second as shown in Figure 7-4), high ease of use and flexibility (could be easily setup at any complicated indoor environment or some outdoor environment).

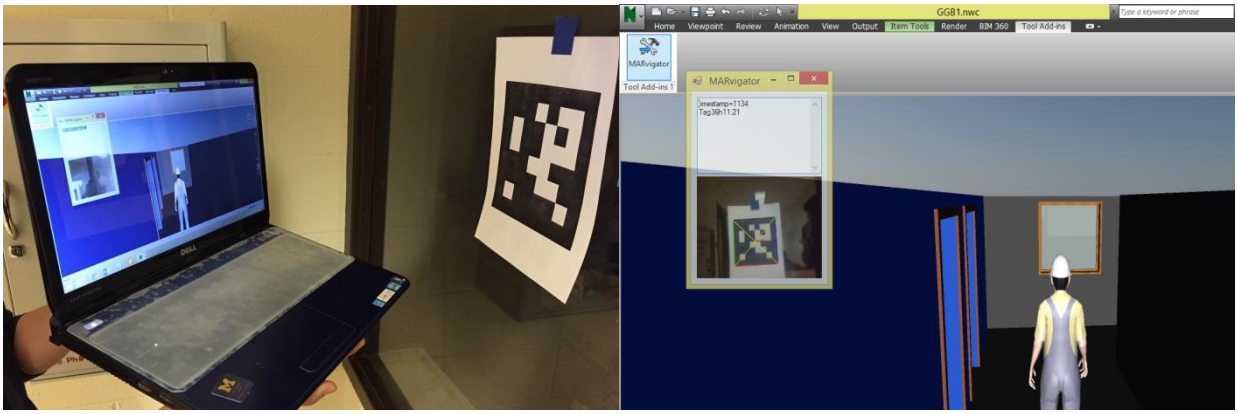


Figure 7-5: Mobile BIM with marker-based location recognition in Autodesk Navisworks.

Similarly, as shown in Figure 7-5, a mobile BIM prototype integrated with marker-based location recognition was implemented as a plugin in Autodesk Navisworks, a widely adopted BIM platform. Using similar idea discussed above, if the target indoor environment has been pre-installed with various markers at the various facility locations, when a facility manager is inspecting a particular equipment, s/he just needs to point the camera on the mobile device to the equipment and the marker nearby. Then this plugin will automatically navigate the virtual camera of the BIM platform (Navisworks) to the corresponding virtual location. Then the manager can easily access and update relevant information, without tedious and inefficient manual search and navigation. Moreover, when the manager moves the mobile device, as long as the marker is still detectable in the image, the virtual camera will be moved in synchronization with that actual movement, creating an immersive experience to facilitate the inspection.

7.5 Experimental Results

In order to show MARvigator's efficiency in helping people find destinations, a set of experiments is conducted on the first floor of a building on the author's university, which is known on the authors' campus for its complex network of corridors and hallways. Seventeen apriltags are placed on critical locations among the region whose positions and orientations are shown in Figure 7-6. Six target positions are selected (room 1351, 1318, 1069, 1188, 1040 and 1504) which need to be sequentially found by each of the 10 experiment volunteers, among which 6 volunteers work/study within a part of this region and the other 4 have never been to this region or are not familiar with it.

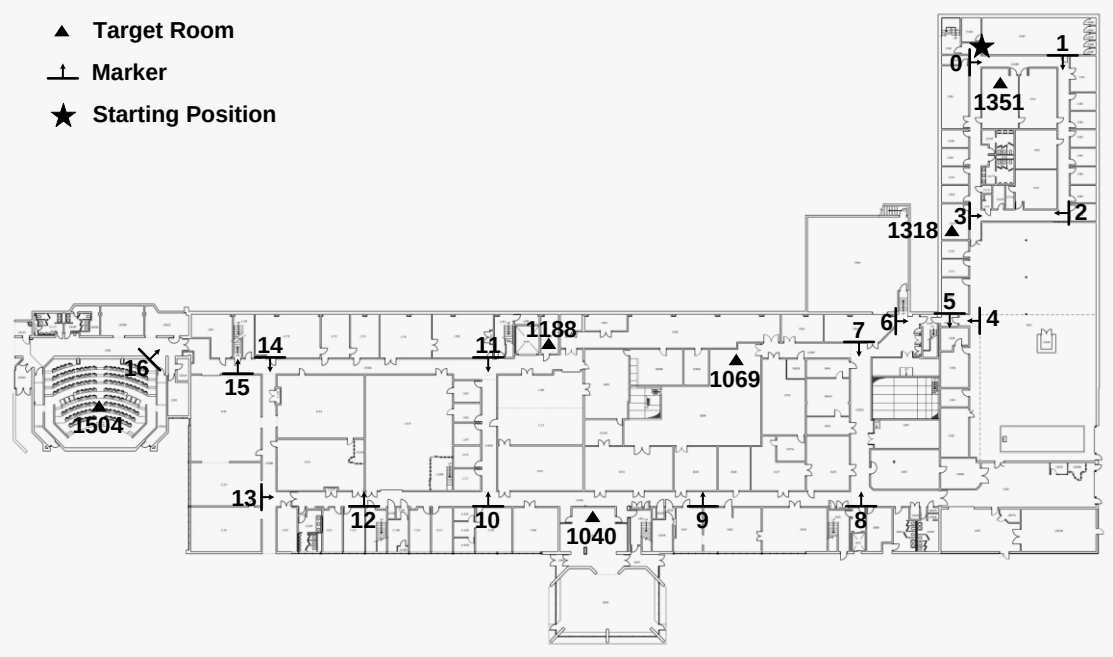


Figure 7-6: Indoor way finding experiment setup.

Each volunteer starts from nearby marker 0. They are instructed on the usage of MARvigator for one minute before the experiment. The time for each volunteer's reaching target room is recorded with resolution of half a minute. In order to reduce to the minimum the influence of

other factors such as variations in walking speed and sense of direction, volunteers are asked to switch between using and not using MARvigator during the way finding. For example, if a volunteer start to find the first target room 1351 with the help of MARvigator, then s/he should not use MARvigator but any other common means of way finding (looking at the map on some part of the building or asking other people for help) to discover the second target room 1318, and vice versa. In the experiment, 5 people start with MARvigator and the other 5 people start without.

The experiment results are shown in Table 7–1. The shaded grid shows the time to find the target room assisted with MARvigator. From these results it is clear that MARvigator does help people to find destinations faster. Notice that for the first two targets, using MARvigator took slightly longer time. This could be explained by the fact that the user needs time to learn how to use the application. Also the first two targets are very close to the start position and can be found easily. However, by comparing the average time of finding later targets, MARvigator's efficiency is better highlighted.

Table 7–1: Experimental results for the 10 volunteers.

Time(min)	**Volunteer who starts with MARvigator**					**Volunteer who starts w/o MARvigator**						
Target	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**Average**	
1351	0.5	1	1	1	0.5	0.5	0.5	0.5	0.5	0.5	0.80	0.50
1318	0.5	0.5	0.5	1	0.5	0.5	1	1	1	0.5	0.60	0.80
1069	0.5	1	1	1	1	0.5	2	5	2	2	**0.90**	2.30
1188	2	2	7	4	2	1.5	2	4	3	2	3.40	**2.50**
1040	1	2	1	1	2	2	0.5	1	4	4	**1.40**	2.30
1504	3	3	2	2	2	2	1	2	3	2	2.40	**2.00**

7.6 Conclusions

In conclusion, the proposed general computing framework of using AR marker as spatial index to link physical locations with virtual information related to that location offers a new perspective of utilizing AR markers. The indoor way finding application MARvigator conforming to this framework is proved by experiments to be very efficient and convenient.

Future research directions could be adopting more advanced computer vision and machine learning algorithms such as topological SLAM (Cummins and Newman 2008) to replace the marker recognition module with landmark recognition module which further decreases the infrastructure dependency of the framework.

Chapter 8

Marker Assisted 3D Scanning and Plane Recognition for As-Built Modeling

"Everything should be as simple as it can be, but not simpler."—Albert Einstein

8.1 Introduction

Modeling of the built environment in 3D is important for many construction and maintenance activities, such as interior design and facility management. As the most relevant information, the architects' designs of built environments, such as 3D geometric models stored in BIM, have a limitation that they do not capture the discrepancies between the designs and the as-built environments, as introduced previously in section 1.2.2.1. Also, such designs may not be readily accessible for end users of the built environment.

This as-built modeling task may be addressed by different methods. As-built survey is a widely applied method using conventional surveying equipment such as total stations to measure positions of key points in the built environment and then generating as-built 2D plans or 3D models usually in wireframe form. Such point-by-point surveys are accurate but inefficient.

3D scanning technology is thus being increasingly adopted due to the fact that it can efficiently collect thousands of 3D points forming a point cloud to describe the environment being scanned. The data collection device in 3D scanning is typically a terrestrial laser scanner (TLS) due to its high accuracy. Various algorithms have been proposed in this area, including data collection,

registration, shape representation, and object recognition, as aforementioned in section 5.1.2.2, as well as the relevant literature review by Tang et al. (2010).

The TLS based 3D scanning methods also have disadvantages, such as high costs for purchasing TLS and corresponding data processing software, requirement of trained experts to design and perform the scan and post-process scan data, and the large volume and weight of TLS. For example, tenants or even owners of an unfurnished residential building may not have, or not be able to conveniently access a detailed as-built 3D model of this building. When they want to perform certain interior designs in this building, the TLS based as-built modeling might be cost prohibitive and offer unnecessary detail and accuracy.

Thus it is of interest to develop cost-efficient, easy-to-use, and lightweight as-built modeling solutions with sufficient accuracy. Laser-based 3D scanning devices are generally expensive, TLS being typical examples. Even the cheapest 2D line-scan lidar such as Hokuyo URG-04LX-UG01 currently costs about a thousand US dollars. Compared with them, off-the-shelf commercial digital cameras are more attractive due to their relatively lower costs. Especially with the recent progress in Structure from Motion (SfM) and Visual SLAM introduced in previous chapters, digital cameras become promising data collection devices to achieve the abovementioned tradeoffs between cost and accuracy.

When investigating those existing vision-based methods, an important technical challenge needs to be effectively addressed, which is the fact that a lot of target environment in need of as-built modeling are featureless or of repeated features, such as unfurnished indoor apartments. As explained in section 1.2.2.1, this violates an important constraint of existing vision-based methods that can provide 3D descriptions of surrounding environment, because locally

distinguishable features are necessary for feature matching, or data association in robotics terms, so that correct and accurate correspondences can be identified and proper geometric relationships can be established across various images.

With previously developed marker-based pose estimation in Chapter 3 and camera marker networks in Chapter 4, this challenge can be readily resolved. By printing out planar markers and attaching them in the target environment, correspondences can be directly and automatically identified through detections and recognitions of those markers in each image. Then using the theoretical framework of camera marker networks, one can efficiently estimate all markers' poses, which can be used to measure critical information of the target environment such as dimensions and dihedral angles between any two planes.

If more detailed as-built modeling of the target environment is desired, e.g., with the purpose of generating realistic models of rooms for online advertisement, a commercial RGBD camera, such as Kinect, can be applied to replace the ordinary RGB camera. Since the camera marker network also estimates poses of each cameras/views, the associated 3D point clouds can be directly registered into the same world coordinate frame, forming a single point cloud describing the target environment. Moreover, recalling the fast plane extraction algorithm that was proposed in Chapter 2, plane extraction and recognition can be performed in each frame of those point clouds to provide additional observation and help to create a more compact and concise description of the target environment mixing points and planes, similar to (Taguchi et al. 2013).

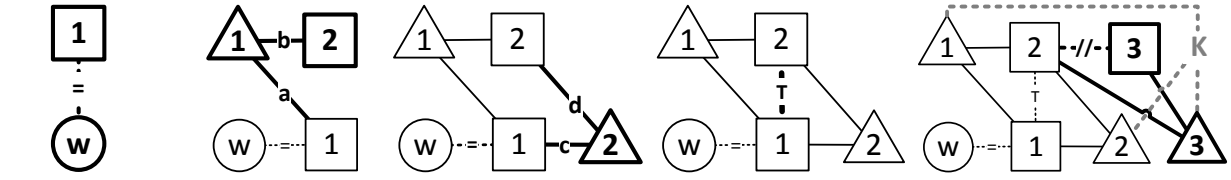
Following the above train of thought, this chapter proposes a marker assisted as-built modeling solution as a cost-efficient and easy-to-deploy alternative to the TLS based methods. The algorithms are explained in section 8.2, and the experimental results are discussed in section 8.3.

8.2 Technical Approach

As briefly described in the previous section, the marker assisted as-built modeling solution proposed here contains two methods. The first and basic method using only an ordinary RGB camera will be explained in section 8.2.1. The second method extends the first one by replacing the RGB camera with an RGBD camera, which will be explained in section 8.2.2.

8.2.1 Marker Structure from Motion

This first method is a direct application of the aforementioned camera marker networks, which is also the foundation of the proposed solution. The basic idea is to attach printed paper markers on planes of the target environment. Then an intrinsically calibrated RGB camera is moved to different proper poses to take a sequence of images of those markers, forming a sequence of views. This results in a dynamic camera marker network of multiple views and multiple markers. When the marker poses are estimated in this camera marker network, poses of the corresponding planes can be determined, since it is reasonable to assume these printed paper markers are on those planes. Since the poses of all markers cannot be finally estimated without moving the single camera to different views, this method is dubbed as marker structure from motion, as it is very similar to traditional point-based SfM.



(a) Initialization | (b) Discover New Markers | (c) Re-observe Old Markers | (d) Detect Marker Constraints | (e) Final Optimization

Figure 8-1: Different operations in marker structure from motion.

As illustrated in Figure 8-1, this marker SfM contains several operations to grow and maintain the dynamic camera marker network, which are detailed in the following subsections.

8.2.1.1 Initialization

Before taking the first image, the camera marker graph needs to be initialized. As shown in Figure 8-1(a), this initialization is to specify the relationship between at least one marker node and the world node. Essentially this operation is to define what the world coordinate frame is in this as-built model. In camera marker graphs, such relationships are represented as soft constraints (dotted lines), i.e. a triplet $\mathbf{c} = (s, e, \mathrm{g})$ where s and e are the indices of the two connected nodes $\mathbf{x}_s, \mathbf{x}_e$ and g is the constraint function. In initialization, this constraint function is to directly specify the values of pose parameters of a marker s in the world coordinate frame e and thus termed as the *fixed-node* constraint (denoted with symbol '='):

$$\mathrm{g}_{=}(\mathbf{x}_s, \mathbf{x}_e) = \mathbf{P}_{=}^{-1/2}(\mathbf{x}_s - \bar{\mathbf{x}}_s) \tag{8.1}$$

where $\bar{\mathbf{x}}_s = [\bar{\mathbf{e}}_s^{\mathbf{T}}, \bar{\mathbf{t}}_s^{\mathbf{T}}]^{\mathbf{T}}$ is the surveyed pose parameters of the markers s, and $\mathbf{P}_{=}$ is the cross-covariance matrix of the surveyed values $\bar{\mathbf{x}}_s$ for properly weighting this constraint (in the sense of Mahalanobis distance). The markers added in this operation are termed as control markers and have similar purposes as control points in conventional surveying and photogrammetry. For end users without surveying equipment, this initialization can be done by simply setting the world coordinate frame as the first marker to be observed, i.e. $\bar{\mathbf{x}}_s = [0,0,0,0,0,0]^{\mathbf{T}}$, while the cross-covariance matrix $\mathbf{P}_{=} = \mathrm{diag}([\sigma_e^2, \sigma_e^2, \sigma_e^2, \sigma_t^2, \sigma_t^2, \sigma_t^2]^{\mathbf{T}})$ with sufficiently small σ_e^2, σ_t^2 that ensure no numerical problems, usually $\sigma_e = 0.01\mathrm{rad}$, $\sigma_t = 0.1\mathrm{mm}$.

8.2.1.2 Discover New Markers

After initialization, the end user can start taking photos of those markers. Each photo will correspond to a new view node to be added to the initialized camera marker graph. An important

rule of thumb of choosing proper poses for taking photos is that at least two markers (and generally the more the better) should be detectable in the photo on the selected pose, and their images should be as far apart as possible in this photo, as shown in Figure 8-2. This is because that this view's pose can be more accurately and stably estimated if all of the observed markers' poses are known.

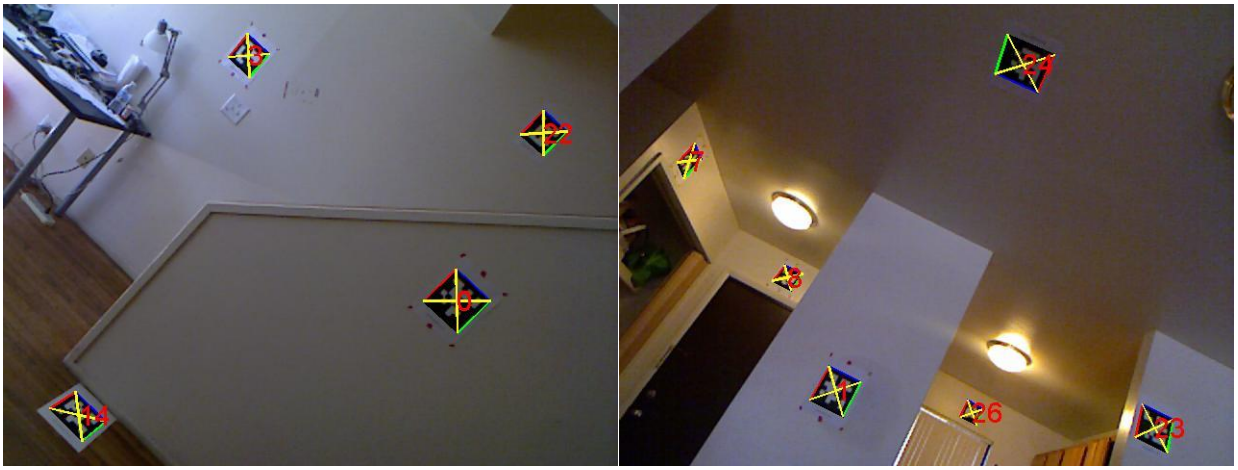


Figure 8-2: Example photos and corresponding marker detections for as-built modeling.

Whenever a new photo is taken, the corresponding view node needs to be initialized and added to the graph. There are three situations. The first one is that none of the markers detected in this new photo have been seen before. In this case, this view cannot be readily initialized and added to the graph because of no connections to existing markers. Thus this photo can be either discarded or temporarily cached for later processing whenever such connections can be found.

The second situation is that among all detected markers there exist some markers that have been observed and added to the graph before, while others are newly detected. In this case, the new view node can be initialized by calculating the relative pose between those observed markers and this view, using either homography decomposition or solving the perspective-n-point problem, as

explained in Chapter 4, section 4.2.3.1. Once the view is successfully initialized and added to the graph, those newly detected markers which have not been observed before can now be subsequently initialized and added to the graph. An example is shown in Figure 8-1(b). After initialization in (a), firstly a new view (view 1) is added by initialize the view's pose using edge a. Then the new marker (marker 2) is added using edge b.

8.2.1.3 Re-observe Old Markers

The third situation is that all of the markers detected in this new photo are old markers that have been observed and added to the graph before. In this case, the new view node can be added using the same method as in the second situation. The only difference is no new markers can be added. For example Figure 8-1(c), the new view (view 2) can be added using edge c or d or both.

It is however worth noting that in marker SfM, if only one old marker is detected in a new photo, this photo is of little value to be added to the graph. This is because that if conditioned on this re-observed old marker's pose, the corresponding view's pose is and will always be independent with poses of all other views and markers, since no more edges can be possibly linked back to this view again. This is different with marker nodes since by adding more views the conditional independence could be removed, e.g., the marker 2 in Figure 8-1(b) is conditional independent on the view 1, but not any more after the view 2 is added in Figure 8-1(c).

8.2.1.4 Detect Marker Constraints

After multiple marker nodes are initialized and added to the graph, their geometric relationship can be examined to detect potential pose constraints between markers. Typical constraints include *parallelism*, *perpendicularity*, *coplanarity*, and the aforementioned *fixed-node* constraint. Generally there are two approaches for enforcing constraints between marker nodes. The first

one is to change the parameterization of the marker pose, so that the estimation solution is inherently ensured to exactly fulfill those constraints, also dubbed as hard constraints. For example, two markers' poses were originally parameterized by two 6D column vectors $\mathbf{x}_1, \mathbf{x}_2$ as described in the Appendix. To enforce the coplanarity hard constraint since they were attached to a same plane, one needs to change $[\mathbf{x}_1^\mathbf{T}, \mathbf{x}_2^\mathbf{T}]^\mathbf{T}$ to $[\mathbf{p}_{12}^\mathbf{T}, \mathbf{q}_1^\mathbf{T}, \mathbf{q}_2^\mathbf{T}]^\mathbf{T}$ where the 4D column vector $\mathbf{p}_{12}$ encodes the normal and position of that plane, and the 2D column vector $\mathbf{q}_1, \mathbf{q}_2$ encodes the 2D position of the two markers in that plane. After this change, the parameter dimension reduces from the original 12 to 8. After any optimization, the parameterization ensures the estimated two markers' poses are on a same plane exactly. The advantage of this approach is that it can usually reduce parameter dimension and thus reduce the complexity of the estimation problem. Yet due to the change of parameters, the implementation of this approach becomes complicated.

The other more convenient approach is to use the so-called soft constraints, as used in Chapter 4. It preserves the uniform representation of each node's pose parameter, and enforces constraints as a special type of observations. When performing optimizations, these constraint residuals are minimized together with ordinary observation residuals. For example, the perpendicularity constraint, denoted with symbol '⊥' in Figure 8-1(d), can be represented as:

$$\mathrm{g}_\perp(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{g}_\perp\left(\begin{bmatrix}\mathbf{e}_i \\ \mathbf{t}_i\end{bmatrix}, \begin{bmatrix}\mathbf{e}_j \\ \mathbf{t}_j\end{bmatrix}\right) = \frac{\mathbf{r}_3(\mathbf{e}_i)^\mathbf{T}\mathbf{r}_3(\mathbf{e}_j)}{\sigma_\perp}, \tag{8.2}$$

and the parallelism constraint, denoted with symbol '//' in Figure 8-1(e), can be represented as:

$$\mathrm{g}_{//}(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{g}_{//}\left(\begin{bmatrix}\mathbf{e}_i \\ \mathbf{t}_i\end{bmatrix}, \begin{bmatrix}\mathbf{e}_j \\ \mathbf{t}_j\end{bmatrix}\right) = \frac{1 - \mathrm{abs}\left(\mathbf{r}_3(\mathbf{e}_i)^\mathbf{T}\mathbf{r}_3(\mathbf{e}_j)\right)}{\sigma_{//}}, \tag{8.3}$$

and the coplanarity constraint, denoted with symbol ‘p’, can be represented as:

$$g_p\left(\begin{bmatrix}\mathbf{e}_i\\ \mathbf{t}_i\end{bmatrix},\begin{bmatrix}\mathbf{e}_j\\ \mathbf{t}_j\end{bmatrix}\right)=\left[\frac{1-\mathrm{abs}\left(\mathbf{r}_3(\mathbf{e}_i)^{\mathbf{T}}\mathbf{r}_3(\mathbf{e}_j)\right)}{\sigma_p},\frac{\mathrm{abs}\left(\mathbf{r}_3(\mathbf{e}_i)^{\mathbf{T}}\mathbf{t}_i\right)-\mathrm{abs}\left(\mathbf{r}_3(\mathbf{e}_j)^{\mathbf{T}}\mathbf{t}_j\right)}{\sigma_p}\right]^{\mathbf{T}},\qquad(8.4)$$

where $\mathbf{r}_3(\mathbf{e})$ is the third column of the rotation matrix $\mathbf{R}(\mathbf{e})$ computed in the Appendix, representing a marker's normal direction in the world coordinate frame; $\sigma_{//}^{-1}$, $\sigma_{\perp}^{-1}$, and σ_p^{-1} are the weighting factors for the constraints indicating the user's confidence of these constraints.

These constraints can be either manually or more intelligently specified. Whenever a new marker node is added to the graph, it can be checked with each marker node existed in the graph for any of the constraints mentioned above. If a resulting constraint residual is below the pre-defined threshold, then a constraint candidate is detected and proposed for user approval.

8.2.1.5 Final Optimization

After a new photo is processed by the above three operations described in section 8.2.1.2, 8.2.1.3, and 8.2.1.4, a full optimization of the current graph can be performed using equation (4.5) as described in section 4.2.3.2, to adjust all current poses in the graph. Note that since all the soft constraints described above have been already properly weighted, the Mahalanobis norm for the constraint residuals vector $\mathbf{G}(\mathbf{X})$ can be replaced with the ordinary vector L^2 norm.

When all photos are processed, a final optimization adjusting the camera intrinsic parameters together with all poses is performed considering the fact that camera intrinsic parameters were calibrated previously independent to this estimation. This slightly modifies equation (4.5) as:

$$\hat{\mathbf{K}},\hat{\mathbf{X}}=\arg\min_{\mathbf{K},\mathbf{X}}\left\|\hat{\mathbf{Z}}-\mathbf{F}_c(\mathbf{K},\mathbf{X};\mathbf{Y})\right\|_{\mathbf{P}_{\hat{\mathbf{Z}}}}^2+\left\|\mathbf{G}(\mathbf{X})\right\|^2,\qquad(8.5)$$

where the camera intrinsic parameters vector **K** becomes a part of the function $\mathbf{F}_c(\mathbf{K}, \mathbf{X}; \mathbf{Y})$'s state/configuration to be adjusted, instead of the original parameters for the function **F**, as explained in section 4.4.2. This is illustrated in Figure 8-1(e) where the **K** is shown in grey with dotted edges linking to all the view nodes whose poses are directly affected when adjusting **K**.

The above five operations can thus be summarized in Algorithm 8–1, which described a post processing version of marker SfM. It can be conveniently converted to online processing by 1) performing step 1 to 8 for each newly taken photo; 2) evaluating pose uncertainties using equation (4.6); 3) performing the final optimization using equation (8.5) after the user stops taking new photos when all markers are estimated with sufficiently small uncertainties.

Algorithm 8–1: Marker Structure From Motion.

Initialize a camera marker graph ***G*** using equation (8.1);

For i = 1 to N:

1. Detect markers in photo $\boldsymbol{I}_i$;
2. If no detected marker exists in ***G***, swap $\boldsymbol{I}_i$ and $\boldsymbol{I}_{i+1}$, and redo step 1;
3. Initialize a new view node ***v*** in ***G*** using detected markers that exist in ***G***;
4. Add an edge between ***v*** and each detected existing marker in ***G***;
5. Initialize a new marker node for each detected markers that is not yet in ***G***;
6. Add an edge between ***v*** and each of these newly added marker nodes;
7. Perform an optimization of all nodes using equation (4.5);
8. For each new marker node ***n*** added in step 5, and each node ***m*** in ***G*** other than ***n***:
    a. Calculate different soft constraint residuals between node ***n*** and ***m***, e.g. using equation (8.2), (8.3) and (8.4);
    b. If any of such residuals is smaller than a pre-defined threshold, request user approval for adding a corresponding constraint edge between node ***n*** and ***m***;

Finally perform an optimization of intrinsic parameters and all nodes using equation (8.5).

8.2.2 Marker and Plane Structure from Motion

Many applications can be readily addressed using the marker SfM proposed above, for example, measuring dimensions for interior design. This is because markers' poses can be used to straightforwardly calculate distances between parallel planes (e.g., height of a room, width of a hallway, etc.), dihedral angles between two planes (e.g., walls, roofs, etc.), and so on. Essentially the markers and the camera serve as a virtual and more accurate tape and protractor.

In some more advanced as-built modeling applications, such as rapidly creating a virtual 3D model of a room for online sales, the marker SfM might not be satisfactory. Considering the fact that low cost 3D imaging devices like Kinect become more popular, a marker and plane SfM using a low cost RGBD camera as data collection device is thus proposed in this section. This method extends camera marker networks with a new type of observations, i.e., 3D planes extracted from depth image using Algorithm 2–1 (page 31).

8.2.2.1 Extended Camera Marker Graph

Just as the original camera marker networks, such an extended one can be considered as a graph **G** = (**V**, **E**) with a set of two types of nodes $\mathbf{V} = (\mathbf{X}, \mathbf{\Pi})$ and a set of three types of edges **E** = (**O**, **Q**, **C**). The only two extended elements are a set of plane nodes $\mathbf{\Pi} = \{\mathbf{p}_i \equiv (\mathbf{e}_i, d_i) \mid i = 1, \cdots, S\}$ and a set of plane observation edges $\mathbf{Q} = \{\mathbf{q}_j \equiv (v_j, p_j, \mathbf{A}_j) \mid j = 1, \cdots, S\}$. Each plane node $\mathbf{p}_i$ is a 3D column vector parameterizing the corresponding plane's 3D orientation and location, as described in the Appendix. Each plane observation edge contains a camera/view node's index v_j and a plane node's index p_j, and also an observation matrix $\mathbf{A}_j$ holding in each column a 3D anchor point sampled from all points belonging to that plane in the raw point cloud observed at

view v_j, similar to (Taguchi et al. 2013). Like equation (4.1), (4.2), and (4.3), the residuals of each plane anchor point $\mathbf{a}_i$ in such an edge $\mathbf{q}$=(v, p, $\mathbf{A}$) is calculated as:

$$\mathrm{h}(\mathbf{x}_v, \mathbf{p}_p, \mathbf{A}) = \mathrm{h}(\begin{bmatrix}\mathbf{e}_v \\ \mathbf{t}_v\end{bmatrix}, \begin{bmatrix}\mathbf{e}_p \\ d_p\end{bmatrix}) = \mathbf{P_q}^{-1/2}[\cdots, \mathbf{n}(\mathbf{e}_p)^{\mathbf{T}}\left(\mathbf{R}(\mathbf{e}_v)\mathbf{a}_i + \mathbf{t}_v\right) + d_p, \cdots]^{\mathbf{T}}, \tag{8.6}$$

where function $\mathbf{R}(.)$ and $\mathbf{n}(.)$ is explained in the Appendix, and $\mathbf{P_q}^{-1/2}$ is the weighting matrix for this edge. This essentially calculates the point-plane distances between the plane and each anchor point transformed into the world coordinate frame, and stack as a column vector.

Stacking such equations for all plane observation edges results in:

$$\mathbf{H}(\mathbf{X}, \mathbf{\Pi}) = \left[\mathrm{h}(\mathbf{x}_{v_1}, \mathbf{p}_{p_1}, \mathbf{A}_1)^{\mathbf{T}}, \cdots, \mathrm{h}(\mathbf{x}_{v_S}, \mathbf{p}_{p_S}, \mathbf{A}_S)^{\mathbf{T}}\right]^{\mathbf{T}}. \tag{8.7}$$

Thus the original optimization in equation (4.5) is extended to:

$$(\hat{\mathbf{X}}, \hat{\mathbf{\Pi}}) = \underset{\mathbf{X},\mathbf{\Pi}}{\arg\min} \left\|\hat{\mathbf{Z}} - \mathbf{F}(\mathbf{X}; \mathbf{Y}, \mathbf{K})\right\|^2_{\mathbf{P}_{\hat{\mathbf{Z}}}} + \left\|\mathbf{H}(\mathbf{X}, \mathbf{\Pi})\right\|^2 + \left\|\mathbf{G}(\mathbf{X})\right\|^2, \tag{8.8}$$

and the full optimization including camera intrinsic parameters in (8.5) is extended to:

$$(\hat{\mathbf{K}}, \hat{\mathbf{X}}, \hat{\mathbf{\Pi}}) = \underset{\mathbf{K},\mathbf{X},\mathbf{\Pi}}{\arg\min} \left\|\hat{\mathbf{Z}} - \mathbf{F}_{\mathrm{c}}(\mathbf{K}, \mathbf{X}; \mathbf{Y})\right\|^2_{\mathbf{P}_{\hat{\mathbf{Z}}}} + \left\|\mathbf{H}(\mathbf{X}, \mathbf{\Pi})\right\|^2 + \left\|\mathbf{G}(\mathbf{X})\right\|^2. \tag{8.9}$$

8.2.2.2 Plane Matching

It is worth noting another difference in this extended graph, which is the matching of a currently observed plane to a plane in the graph. While matching markers across different views in the original graph is straightforward with markers' unique IDs, 3D planes extracted from point clouds cannot be matched in that way. However with marker SfM, a newly added view's pose is

already estimated in the world coordinate frame approximately. Thus each extracted 3D plane in this view can be transformed to the world coordinate frame and matched to its most "similar" plane in the extended graph, in terms of criteria such as the normal and distance deviations:

$$\mathrm{d}(\mathbf{p}_a, \mathbf{p}_b) = [\mathrm{acos}\left(\mathbf{n}(\mathbf{e}_a)^{\mathbf{T}}\mathbf{n}(\mathbf{e}_b)\right), \mathrm{abs}(d_a - d_b)]^{\mathbf{T}}, \tag{8.10}$$

where the second term may be replaced by the average distance between one plane and anchor points of the other plane, to increase matching robustness.

Algorithm 8–2: Marker and Plane Structure From Motion.

Initialize a camera marker graph ***G*** using equation (8.1);

For i = 1 to N:

9. Perform step 1 to 8 of Algorithm 8–1 (page 176) on the photo $\boldsymbol{I}_i$;
10. If the photo was swapped in step 2 of Algorithm 8–1, swap the point cloud $\boldsymbol{D}_i$ with $\boldsymbol{D}_{i+1}$;
11. Extract planes using Algorithm 2–1 (page 31) on $\boldsymbol{D}_i$;
12. For each extracted plane ***p***:
    a. Transform ***p*** to the world coordinate frame using the pose of the new view node ***v*** just added in step 3 of Algorithm 8–1;
    b. Find the best matching plane ***q*** of ***p*** in all plane nodes in ***G***, using equation (8.10);
    c. If the differences between ***q*** and ***p*** are within pre-defined thresholds:
        i. Add a new plane observation edge between ***v*** and ***q*** to ***G***;
        ii. Expand the boundary of ***q*** by ***p***;
    d. Otherwise:
        i. Add ***p*** as a new plane node to ***G***;
        ii. Add a new plane observation edge between ***v*** and ***p*** to ***G***;
13. Perform an optimization of all nodes using equation (8.8);

Finally perform an optimization of intrinsic parameters and all nodes using equation (8.9).

Finally, the extended marker and plane SfM is summarized in the above Algorithm 8–2. Note that constraint edges described in section 8.2.1.4 can be add between plane nodes in similar ways as described in step 8 of Algorithm 8–1, which need not be repeated here.

8.3 Experimental Results

8.3.1 Accuracy of Marker Structure from Motion

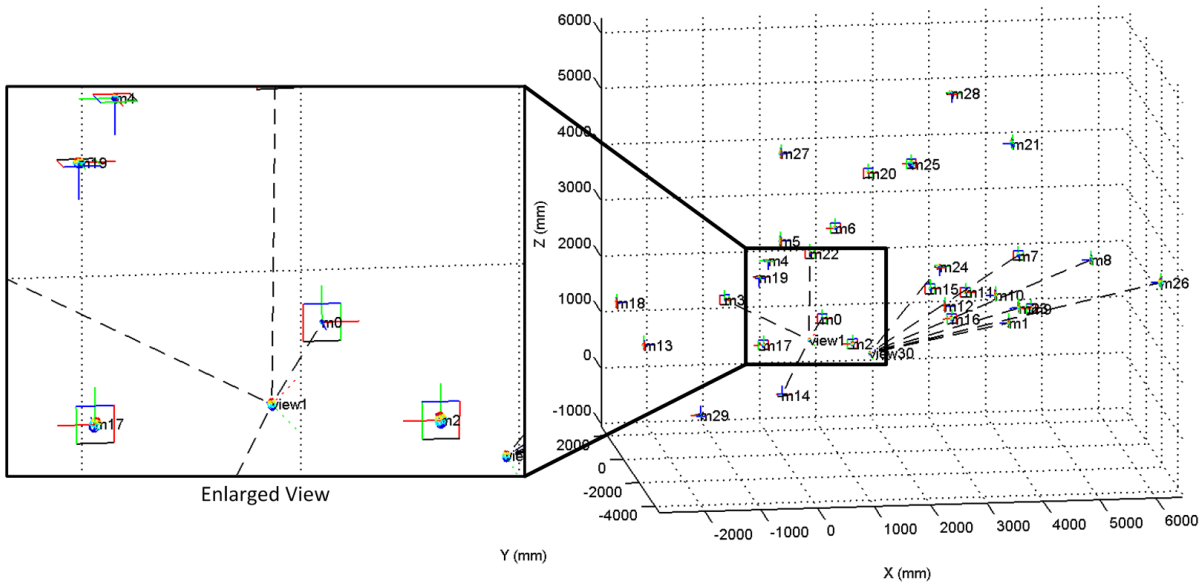


Figure 8-3: Marker SfM results.

Both Algorithm 8–1 and Algorithm 8–2 were implemented in MATLAB using the *lsqnonlin* function in its optimization toolbox to compute the aforementioned nonlinear optimizations. A set of 30 Apriltags (tag family: Tag36h11) were printed on ordinary A4 papers, each tag of the same size 172mm. These markers are then attached on walls, floors and ceilings of a two-story apartment, as partly shown in Figure 8-2. The camera marker graph was initialized by setting marker 14 (attached on the floor) as the world coordinate frame. After 66 views of photos were taken, the marker structure results are shown in Figure 8-3. Only the view 1 (corresponds to the left image of Figure 8-2) and the view 30 (the right image) and their associated edges were

plotted, while other views and edges were hided for clarity purpose. The small ellipsoid centered on each marker denotes the largest position estimation uncertainty of that marker's center in the world coordinate frame (ellipsoid was again plotted by $\chi^2 = 9$ and $\sigma_u = 0.2$ pixels, as in Figure 4-4 on page 96).

To evaluate the accuracy of marker SfM, a Topcon PS 101A total station (2mm nominal distance measurement precision within 100m, and 0.5 second nominal angular measurement precision) was employed to survey those markers' poses as a baseline for comparison. To avoid introducing additional station registration errors in this baseline result, the total station was setup in a single station that can directly observe a maximum of 17 among all of the 30 markers. Then each of the four corners of these 17 markers was surveyed, resulting in a 3 by 68 matrix ${}^{s}\mathbf{X}$. The corresponding estimated corner positions, ${}^{w}\mathbf{X}$, were calculated using those markers' poses from marker SfM. Using the well-known rigid body registration algorithm (Besl and McKay 1992), one can calculate the transformation ${}^{w}\mathbf{R}_s, {}^{w}\mathbf{t}_s$ that transforms each surveyed point ${}^{s}\mathbf{X}_i$ in the total station's coordinate frame to the world coordinate frame, and then compare the discrepancies with the corresponding estimated point ${}^{w}\mathbf{X}_i$ as:

$$\mathbf{E}_i = {}^{w}\mathbf{X}_i - ({}^{w}\mathbf{R}_s {}^{s}\mathbf{X}_i + {}^{w}\mathbf{t}_s). \tag{8.11}$$

As shown in the top-left image of Figure 8-4, green '+' shows each transformed surveyed point ${}^{w}\mathbf{R}_s {}^{s}\mathbf{X}_i + {}^{w}\mathbf{t}_s$, red '**.**' shows the marker SfM estimated point ${}^{w}\mathbf{X}_i$. The top-right image shows the error vector $\mathbf{E}_i$ in black '**.**', and its projection onto each side planes ($\mathbf{E}_x$, $\mathbf{E}_y$, $\mathbf{E}_z$). The bottom-left image shows a histogram of $\mathbf{E}_i$ and bottom-right shows the relative error ($\mathbf{E}_i$ dividing by the scale of all these markers' distribution on the X, Y and Z directions). One can verify that the

majority of errors are within 10mm with the largest positional error of 29.5mm, while the absolute errors $|\mathbf{E}_i|$ on average are 5, 4, 2mm on X, Y, Z directions respectively. Note that these markers are distributed with a scale of about 9m along the X direction, shown in the top-left image. This suggests that the maximum relative error of this marker SfM is about 0.3%, which is of sufficient accuracy considering the fact that it was achieved using an ordinary webcam-style RGB camera (of image size 640x480 pixels) on the Kinect device (depth image was not yet used in this experiment yet). The processing rate for these 66 photos was about 1 minute per photo on average in this MATLAB code, which could be hugely improved with C++ code in the future.

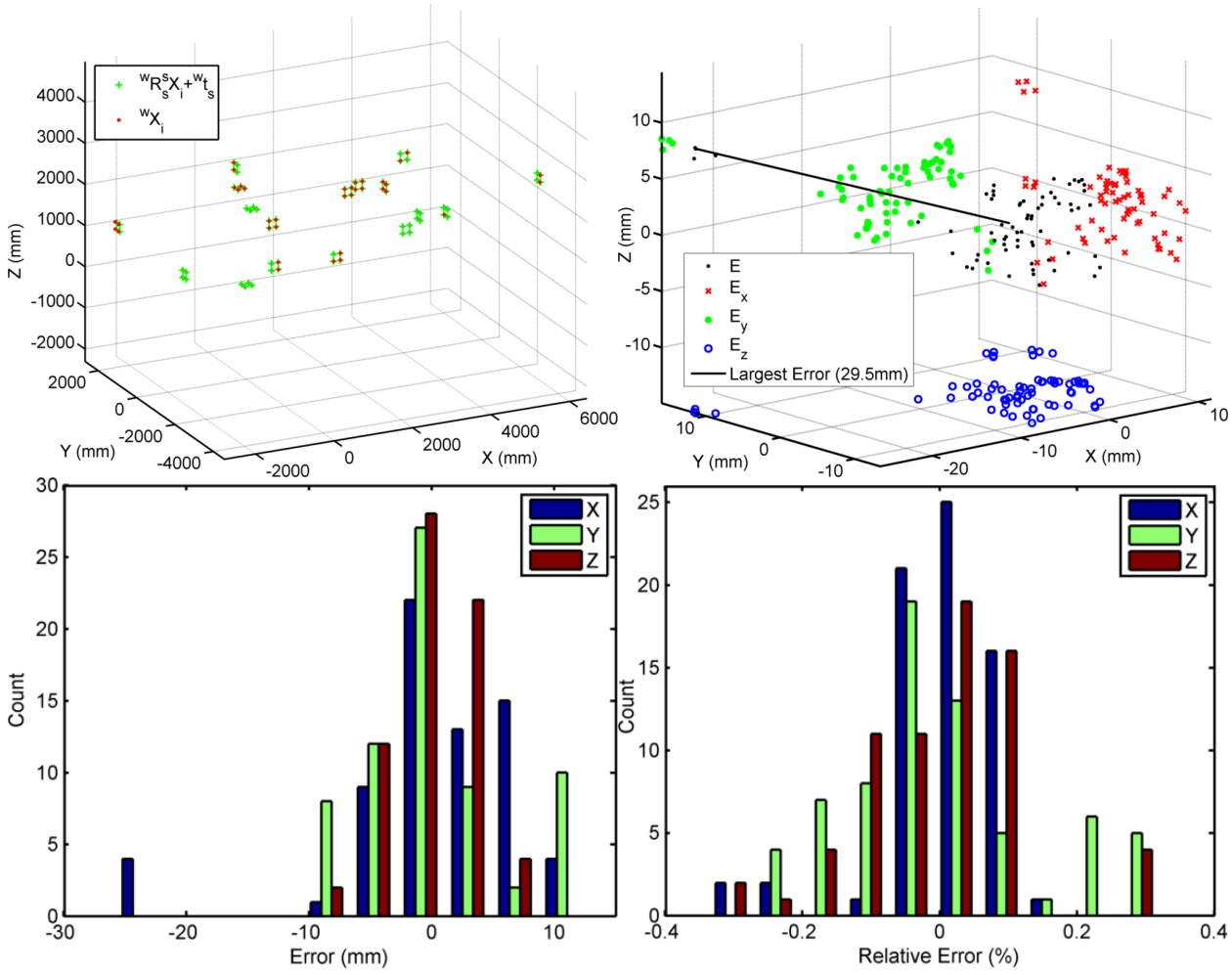


Figure 8-4: Marker corner position differences between surveyed and estimated results.

8.3.2 As-built Models from Marker and Plane Structure from Motion

A second experiment was performed to test the marker and plane SfM algorithm, using the previous 66 RGB images, and the corresponding 66 depth image. Some intermediate results are shown in Figure 8-5, such as plane segments in each frame (top row, corresponding to the two images in Figure 8-2), and planes boundaries extracted in each frame and remaining non-planar points all plotted together in the world coordinate frame (bottom).

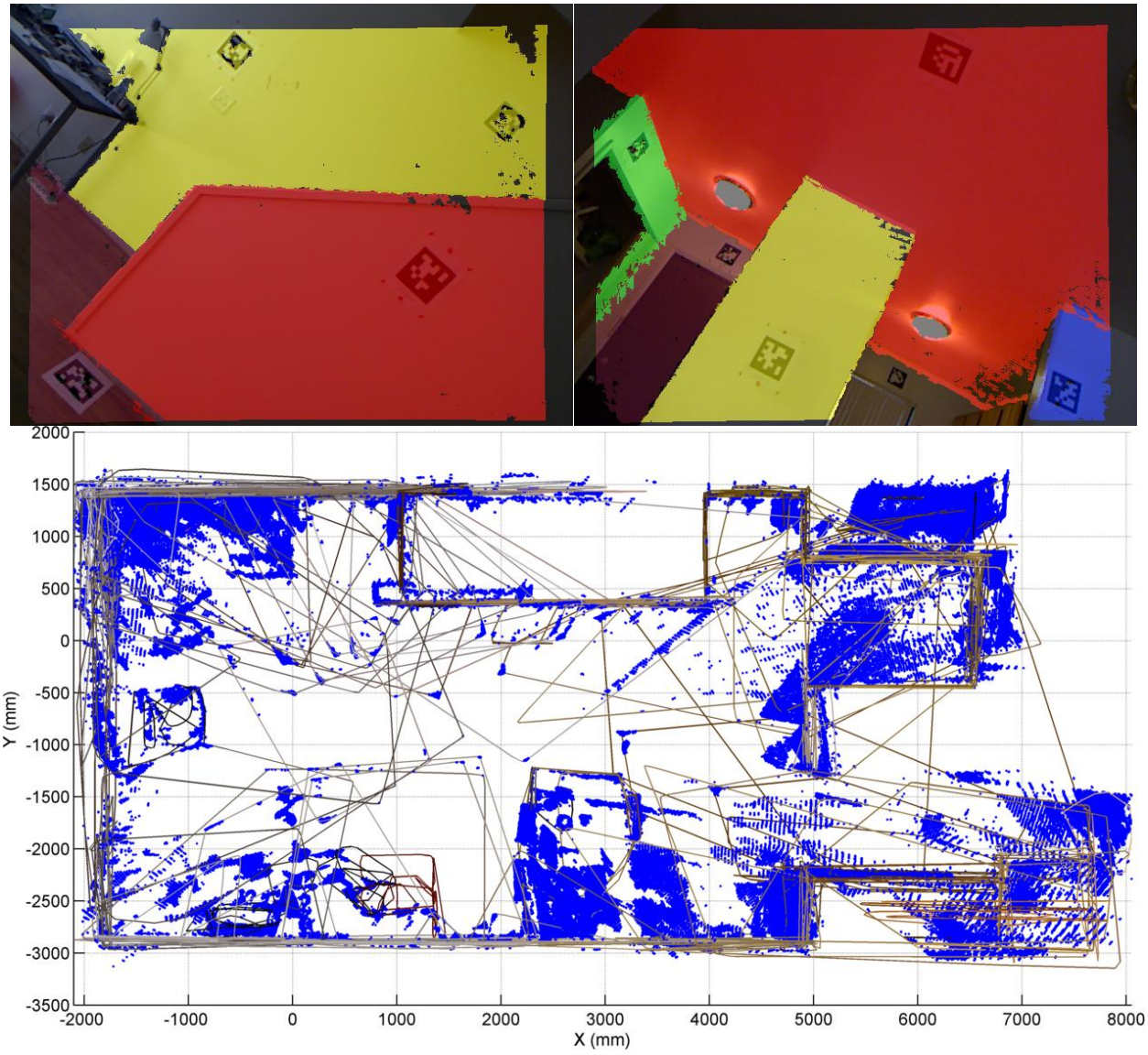


Figure 8-5: Intermediate results for marker and plane SfM.

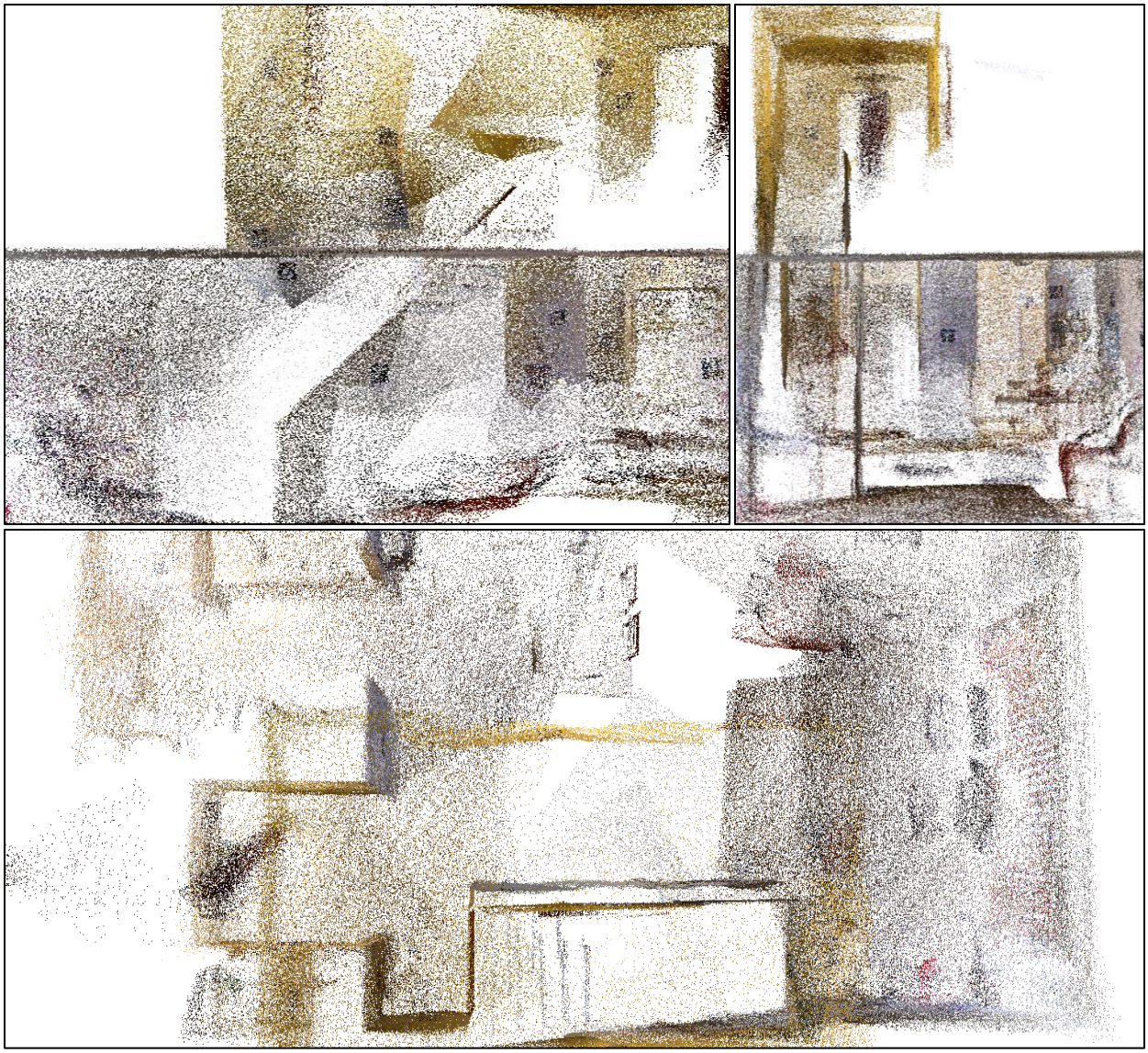


Figure 8-6: Point cloud results for marker and plane SfM.

After Algorithm 8–2 was performed on these 66 RGBD images, the 66 point clouds were transformed into the world coordinate frame using estimated poses for each view and then merged into a single point cloud, whose side, front, and top views are shown in the top-left, top-right, and bottom of Figure 8-6. This is the point cloud form of the as-built model of the apartment using the proposed method, with only a few paper markers and a low cost Kinect camera.

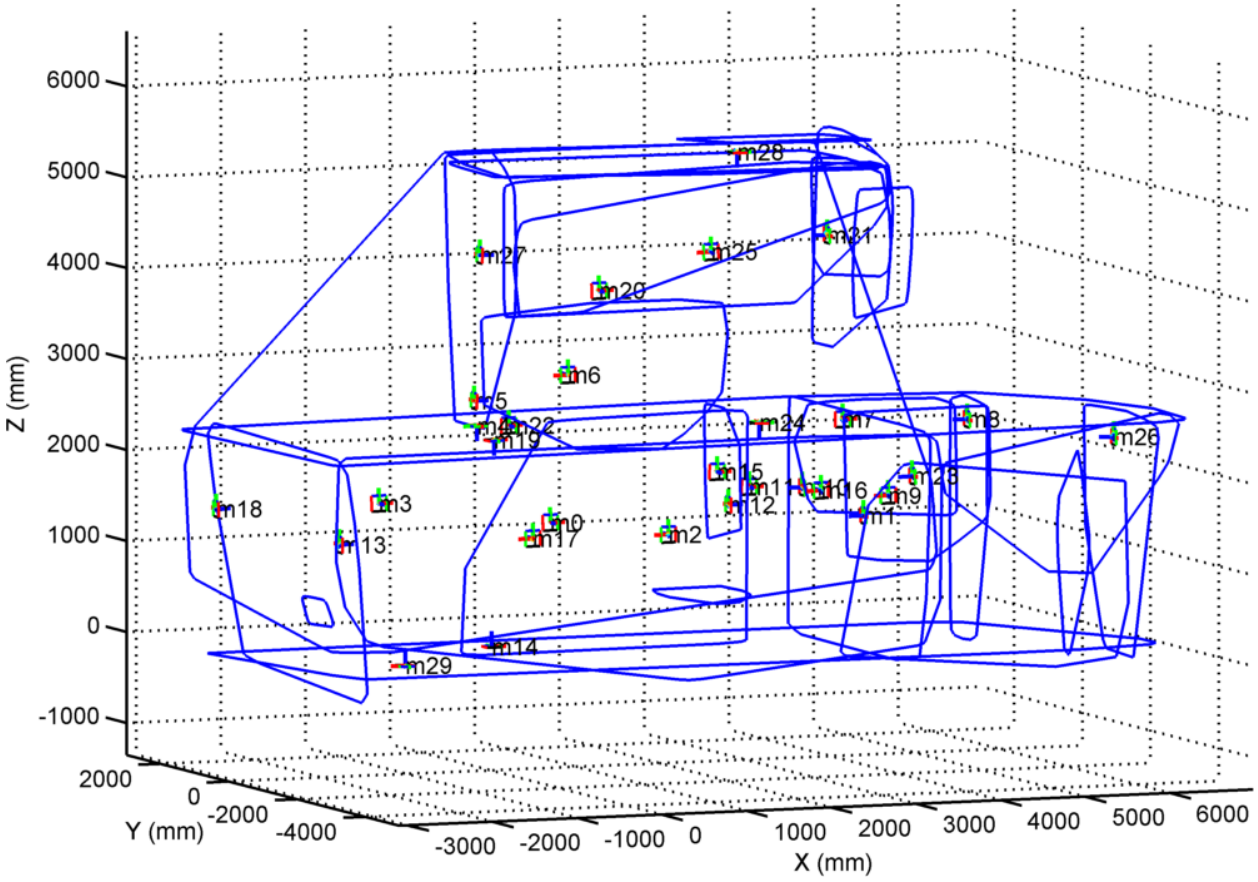


Figure 8-7: Wireframe results for marker and plane SfM.

Another more concise form of the as-built model is shown in Figure 8-7, where each plane node was plotted as a single planar 3D polygon. These polygons are the boundaries of each plane, merged from the boundaries of the same plane observed in each single view of point cloud, in step 12.c.ii of Algorithm 8–2. Compared to the previous point cloud form, this polygon form is more similar to the parametric models used in BIM and has more semantics.

8.4 Conclusions

In conclusion, this chapter combined the fast plane extraction algorithm and the camera marker network method, resulted in two types of novel as-built modeling techniques, marker SfM, and marker and plane SfM. The two proposed techniques require only a low cost RGB/RGBD

camera and a few paper markers to perform dimension measurements and 3D scanning and modeling, which essentially enable rapid and cost-efficient as-built modeling. Furthermore, the experiment has demonstrated that these techniques can achieve satisfactory accuracy (average absolute error within 5mm; maximum positional error 29.5mm over a 9m scale, i.e. 0.3% relative positional error) at sufficiently large scales. Moreover, the marker and plane SfM algorithm enables automatic wireframe as-built model generation which could help automatic as-built BIM generation.

The future direction for this research contains several aspects. Firstly the marker and plane SfM algorithm's accuracy needs to be further improved especially considering the fact that the systematic error in the depth image and thus in the raw point cloud from Kinect will adversely affect the accuracy of the marker SfM. A more accurate RGBD camera calibration might help to reduce such influence. Secondly the implementations of the two algorithms need to be transplanted to C++ using the aforementioned efficient generic bundle adjustment software libraries such as Ceres (Agarwal and Mierle 2012) or g2o (Kummerle et al. 2011), so as to improve the time performance. Thirdly a more intelligent user guidance algorithm needs to be integrated similar to the AprilCal (Richardson et al. 2013) where end users will be guided by the algorithm step by step to take photos at the best poses generated by the algorithm. This will greatly shorten the amount of time for end users to adopt such techniques. Fourthly, the potential of using multiple cameras forming a camera cluster in this as-built modeling application needs to be explored to increase the flexibility and reduce the amount of views needed to complete a modeling process. Last but not least, visual SLAM techniques needs to be integrated to further increase the efficiency and range of applications.

Chapter 9
Conclusions

"If you will tell me precisely what it is that a machine cannot do, then I can always make a machine which will do just that."—John von Neumann

9.1 Significance of the Research

Currently, skill-intensive industries like construction are facing several challenges. The first one is high rates of workplace injuries and fatalities. Due to various reasons such as emotional or environmental ones, even skilled labors will inevitably make mistakes at work, which may result in accidents at job sites. The second one is relatively stagnant productivity rates, especially in construction industry compared to others. Unlike industries largely accelerated through integrations of machines, the essence of construction arguably remains unchanged. Large numbers of workers, skilled or not, are needed to complete a construction project. The third one is the shortage of skilled labor, which attracts more attention nowadays when the baby boomer generation is retiring while birth rates of many developed and even developing countries are decreasing. Moreover, when construction careers' public images are still "low-tech, low-safety, low-income, and unstable", such skilled labor shortage will become worse.

Under such background, Automation and Robotics in Construction (ARC) is becoming more and more attractive. Firstly, machines, especially robots, once properly setup, have much lower

possibilities of making mistakes at work. Secondly, machines and robots can work in environments that are dangerous and unhealthy for human workers. Thirdly, machines can augment human worker's abilities for avoiding dangerous situations and perform better and faster.

However, to realize all the above merits of ARC, some fundamental problems must be addressed properly. Pose estimation and scene understanding—the core focus of this research, and belonging to two broader topics in robotics: perception and navigation—are among such problems. They are critical and significant because of the two challenges of applying ARC: the unstructured and featureless environment, as well as the differences between designs and the as-built infrastructures. These two challenges make it particularly difficult to directly implement conventional forms of automation and robotics that have been successfully adopted in manufacturing industries. This is because manufacturing and construction, though appearing to be very similar, have fundamental differences regarding those two challenges. Manufacturing environment is usually indoor, controlled factory settings. Manufacturing designs and as-built products have very tight tolerances so designs can accurately reflect the true states of products. However construction has opposing circumstances. Conventionally, construction often needs to be performed onsite where the final products, the buildings or civil infrastructure, are to be used. Thus the construction jobsites are often outdoors, unstructured and rugged (even indoors). The architecture designs and the final buildings also could have larger discrepancies.

Thus, pose estimation and scene understanding become essential for ARC. Reliable and accurate pose estimation can enable construction machinery and devices to maintain estimation of their locations and orientations on the unstructured jobsites. Fast and semantic scene reconstruction and understanding can empower them to make sense of the environment that might be different

than the designs in their records. With these abilities, machinery can be easier to control and collaborate with, require less human supervision, and even become autonomous on complex worksites. Wearable and mobile devices can facilitate construction and post-construction activities by helping human workers make better decisions. These could finally alleviate the issues mentioned before.

Previous research on pose estimation for ARC focuses on technologies that are heavily dependent on powered hardware infrastructures, such as GPS, WLAN, UWB, and RFID. Thus they have disadvantages such as relatively high-cost, inability to support rapid setup and reconfiguration, or no direct 6DOF pose estimation. Previous research on scene reconstruction and understanding for ARC focuses on heavy-duty equipment and specialized software such as terrestrial laser scanner (TLS) which could be not easily accessible or simply cost prohibitive and technically redundant. Recently ARC researchers have looked into cost-efficient alternatives such as image-based 3D reconstruction (Golparvar-Fard et al. 2009) and vision-based motion capture (Han and Lee 2013).

A potential gap in this new research stream is the overlook of an important challenge in construction environment: featureless or repeated features. If not properly addressed, this challenge can affect performances of many vision-based pose estimation and scene understanding algorithms. This is because one of their core operations is the robust feature matching or data association, which depends on the fundamental assumption that there are enough locally distinguishable visual features in the images.

This dissertation documented the research that follows the above train of thoughts. This research presents a pose estimation and scene understanding framework for filling the unnoticed gap by

taking advantages of markers together with cameras, and then addresses the aforementioned technical challenges. This framework is centered on planes, markers, and cameras. The fast plane extraction algorithm provides a concise and efficient way of modeling and understanding the environment. The marker-based pose estimation, or plane registration, provides a robust, accurate, cost-efficient, and rapidly reconfigurable solution to the featureless challenge. Finally the camera (both RGB and RGBD) marker networks provide unified links between markers and planes in the previous two algorithms.

After these three fundamental algorithms, this research applied them in two groups of ARC applications to demonstrate effectiveness of the proposed framework, while the applications themselves are of great value to the relevant industry sectors. The first group is related to robotic construction machinery where pose estimation is critical for increasing the safety and productivity of tasks that involve such machinery. The applications in this group include an autonomous robotic manipulator that can assemble complex digital architecture designs, and a guiding and monitoring system for articulated machinery. The second group is related to ARC with human-in-the-loop, where a core principle is to enable human-machine interaction so as to be effective. The applications in this group include a mobile BIM enabled indoor facility management, and an as-built modeling solution.

9.2 Research Contributions

This research contributes to the automation and robotics in construction and other relevant industries by improving jobsite safety, increasing productivity and potentially alleviating skilled labor shortage with the implementation of ARC applications developed based on the three fundamental algorithms on pose estimation and scene understanding. For construction contractors, engineering, maintenance, inspection and management personnel, this will alter

traditional methods of using construction machinery and devices, change conventional means of accessing and updating project information, and thus improve their safety and productivity.

The individual research challenges that were successfully examined and addressed in the proceeding chapters are summarized as follows:

- A general-purpose fast and accurate plane extraction algorithm that runs in real-time (35Hz) for better understanding of 3D environment.
- A general-purpose robust, stable, accurate and fast marker-based pose estimation algorithm as the basis for all following applications.
- A general-purpose theoretical framework for rapid reconfigurable pose estimation in large scales, with mathematical solutions and uncertainty analysis for design guidance.
- An autonomous robotic manipulator that is capable of assembling digital architecture designs on jobsites.
- A guiding and monitoring system for articulated machines such as excavator to avoid collisions or guiding excavation according to design profiles.
- A location recognition solution for indoor facility management on mobile BIM.
- An as-built modeling solution that is accurate, efficient and cost-efficient.

9.3 Future Directions of Research

As previously stated, this research centered on markers, planes and cameras. The future directions of this research are thus to extend these three factors to overcome the outstanding limitations from the current work. Detailed and specific limitations have been pointed out in the conclusion sections of each individual chapter. In addition, the following sections discuss several specific thrusts as directions for future research.

9.3.1 Extending to Non-Fiducial Features

In this research, markers were used to provide reliable and fast feature matching. This is particularly useful in featureless or repeated-feature environment. Yet even in such environment, there could still exist some locally distinguishable features, although not as dense as common indoor environment after finishing, or outdoors. Moreover, state-of-the-art visual SLAM algorithms are mostly purely based on non-fiducial features. Thus it could be worth considering to take these non-fiducial features into account to gain more observations and reduce the number of markers needed to be manually installed.

9.3.2 Extending to Non-Planar Structures

In this research, no matter 3D planes or planar markers are all plane structures in the environment. In many industrial environments or modern architecture designs, non-planar structures such as curved surfaces, are not uncommon. Thus it could be interesting to develop fast non-planar structure extraction, for completeness and also increasing the potential range of applications.

9.3.3 Extending to Non-Central-Projection Cameras

In this research, both RGB and RGBD cameras are assumed to be compliant with the central-projection model, e.g., usually without much distortions on the image. However the field-of-view of such a camera is usually small, which is a significant limitation. Cameras with non-central-projection lenses could potentially address this issue and thus worth including into the framework in the future.

Appendix

6DOF Pose Parameterization

The 6DOF pose parameterization $\mathbf{x} = [\mathbf{e}^\mathbf{T}, \mathbf{t}^\mathbf{T}]^\mathbf{T}$ adopted in this research can be described as:

$$\mathbf{T}(\mathbf{x}) = \mathbf{T}(\begin{bmatrix} \mathbf{e} \\ \mathbf{t} \end{bmatrix}) = \begin{bmatrix} \mathbf{R}(\mathbf{e}) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix},$$

where the **R**(**e**) is the well-known Rodrigues' rotation formula:

$$\mathbf{R}(\mathbf{e}) = \exp\left(\|\mathbf{e}\|[\mathbf{e}]_\times\right) = \mathbf{I} + [\mathbf{e}]_\times \sin\|\mathbf{e}\| + [\mathbf{e}]_\times^2 (1 - \cos\|\mathbf{e}\|)$$

3D Plane Parameterization

The 3D plane parameterization $\mathbf{p} = [\mathbf{e}^\mathbf{T}, d]^\mathbf{T}$ adopted in this research can be described as:

$$\mathbf{n}(\mathbf{e}) = \mathbf{n}(\begin{bmatrix} e_1 \\ e_2 \end{bmatrix}) = \begin{bmatrix} \cos e_2 \cos e_1 \\ \cos e_2 \sin e_1 \\ \sin e_2 \end{bmatrix}$$

where $\mathbf{n}(\mathbf{e})$ is the 3D unit normal vector of the plane, encoded using spherical coordinates. Note that the **e** here is a 2D column vector instead of 3D in the pose parameter. The third parameter *d* is simply the distance from the world origin point to this plane.

Bibliography

Abdel-Aziz, Y. (1971). "Direct linear transformation from comparator coordinates in close-range photogrammetry." *ASP Symposium on Close-Range Photogrammetry in Illinois, 1971*.

Agarwal, S., and Mierle, K. (2012). "Ceres solver: Tutorial & reference." *Google Inc*.

Ahlquist, S., and Menges, A. (2011). "Introduction: Computational Design Thinking." In A. Menges, and S. Ahlquist (Eds.), *Computational Design Thinking* (10-29). West Sussex: John Wiley & Sons Ltd.

Akula, M., Dong, S., Kamat, V., Ojeda, L., Borrell, A., and Borenstein, J. (2011). "Integration of infrastructure based positioning systems and inertial navigation for ubiquitous context-aware engineering applications." *Advanced Engineering Informatics, 25*(4), 640--655.

Akula, M., Lipman, R., Franaszek, M., Saidi, K., Cheok, G., and Kamat, V. (2013). "Real-time drill monitoring and control using building information models augmented with 3D imaging data." *Automation in Construction, 36*, 1--15.

Andoh, A. R., Su, X., and Cai, H. (2012). "A Boundary Condition-based Algorithm for Locating Construction Site Objects Using RFID and GPS." *Proceedings of 2012 Construction Research Congress.*, West Lafayette, IN

Associated General Contractors of America. (2014). *"2014 Workforce Survey Results."* Retrieved February 6, 2015, from http://www.agc.org/galleries/news/2014_Workforce_National.pdf

Augmented Reality & Assistive Technology Laboratory of NUS. (2011). *AR Navi NUS*. Retrieved December 6, 2012, from http://serve.me.nus.edu.sg/ongsk/ARLab/arnavi.htm

Aziz, Z., Anumba, C., Ruikar, D., Carrillo, P., and Bouchlaghem, D. (2005). "Context aware information delivery for on-site construction operations." *Proceedings of the 22nd CIB-W78 Conference on Information Technology in Construction, Institute for Construction Informatics, Technische Universitat Dresden, Germany*, CBI Publication 321-32.

Azuma, R. (1997). "A survey of augmented reality." *Presence:Teleoperators and Virtual Environments*, 355-385.

Balaguer, C. (2004). "Nowadays trends in robotics and automation in construction industry: Transition from hard to soft robotics." *In 21st International Symposium on Automation and Robotics in Construction (ISARC'04).*, Jeju. Korea

Bao, S. Y., and Savarese, S. (2011). "Semantic Structure from Motion." *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.*

Bay, H., Tuytelaars, T., and Gool, L. V. (2008). "SURF: speeded-up robust features." *Proceedings of 9th European Conference on Computer Vision*, Graz, Austria, *110*, 346-359.

Behzadan, A. H., and Kamat, V. R. (2009). "Automated generation of operations level construction animations in outdoor augmented reality." *Journal of Computing in Civil Engineering, 23*(6), 405-417.

Bekkali, A., Sanson, H., and Matsumoto, M. (2007). "RFID indoor positioning based on probabilistic RFID map and Kalman filtering." *IEEE*, Proceedings of 3rd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 21-21.

Beliveau, Y. J., Fithian, J. E., and Deisenroth, M. P. (1996). "Autonomous vehicle navigation with real-time 3D laser based positioning for construction." *Automation in Construction, 5*(4), 261-272.

Benhimane, S., and Malis, E. (2004). "Real-time image-based tracking of planes using efficient second-order minimization." *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1*, 943-948.

Benhimane, S., Malis, E., Rives, P., and Azinheira, J. (2005). "Vision-based control for car platooning using homography decomposition." *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2161-2166.

Besl, P. J., and McKay, N. D. (1992). "A Method for Registration of 3-D Shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence, 14*(2), 239--256.

Borthwick, J., Lawrence, P., and Hall, R. (2009). "Mining haul truck localization using stereo vision." *Proceedings of the Robotics and Applications Conference, 664*, 9.

Brilakis, I., Lourakis, M., Sacks, R., Savarese, S., Christodoulou, S., Teizer, J., and Makhmalbaf, A. (2010). "Toward automated generation of parametric BIMs based on hybrid video and laser scanning data." *Advanced Engineering Informatics, 24*(4), 456--465.

Brookshire, J. (2014). *"Articulated pose estimation via over-parametrization and noise projection."* MIT: Cambridge

Bureau of Labor Statistics. (2013). *"Census of Fatal Occupational Injuries (CFOI) - Current and Revised Data."* Retrieved February 3, 2015, from http://www.bls.gov/iif/oshcfoi1.htm

Byrd, R., Gilbert, J., and Nocedal, J. (2000). "A trust region method based on interior point techniques for nonlinear programming." *Mathematical Programming, 89*(1), 149--185.

Cheng, P., and Oelmann, B. (2010). "Joint-angle measurement using accelerometers and gyroscopes—A survey." *IEEE Transactions on Instrumentation and Measurement, 59*(2), 404--414.

Cho, Y., and Gai, M. (2014). "Projection-Recognition-Projection Method for Automatic Object Recognition and Registration for Dynamic Heavy Equipment Operations." *Journal of Computing in Civil Engineering, 28*(5), A4014002.

Christensen, H., Batzinger, T., Bekris, K., Bohringer, K., Bordogna, J., Bradski, G., . . . others. (2009). "A roadmap for US robotics: From Internet to robotics." *Computing Community Consortium and Computing Research Association, Washington DC (US)*.

Chum, O., and Matas, J. (2005). "Matching with PROSAC-progressive sample consensus." *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1*, 220-226.

Chung, J., Lee, S., Yi, B.-J., and Kim, W. (2010). "Implementation of a foldable 3-DOF master device to a glass window panel fitting task." *Automation in Construction, 19*(7), 855--866.

Coleman, T., and Li, Y. (1996). "An interior trust region approach for nonlinear minimization subject to bounds." *SIAM Journal on optimization, 6*(2), 418--445.

Cui, Y., and Ge, S. (2003). "Autonomous vehicle positioning with GPS in urban canyon environments." *IEEE Transactions on Robotics and Automation, 19*(1), 15--25.

Cummins, M., and Newman, P. (2008). "FAB-MAP: Probabilistic localization and mapping in the space of appearance." *The International Journal of Robotics Research, 27*(6), 647--665.

Davison, A., Reid, I., Molton, N., and Stasse, O. (2007). "MonoSLAM: Real-time single camera SLAM." *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29*(6), 1052-1067.

Deschaud, J.-E., and Goulette, F. (2010). "A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing." *Proc. Int'l Symp. 3D Data Processing, Visualization, and Transmission (3DPVT).*

Dierichs, K., Schwinn, T., and Menges, A. (2012). "Robotic Pouring of Aggregate Structures." In S. Bell-Cokcan, and J. Braumann (Ed.), *Robotic Fabrication in Architecture, Art, and Design*, SpringerWienNewYork: New York 196-205.

Dijkstra, E. (1959). "A note on two problems in connexion with graphs." *Numerische mathematik, 1*(1), 269--271.

Dong, S., Behzadan, A. H., Feng, C., and Kamat, V. R. (2013). "Collaborative visualization of engineering processes using tabletop augmented reality." *Advances in Engineering Software, 55*(0), 45 - 55.

Drummond, T., and Cipolla, R. (2002). "Real-time visual tracking of complex structures." *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(7), 932-946.

Dubor, A., and Diaz, G.-B. (2012). "Magnetic Architecture." In S. Brell-Cokcan, and J. Braumann (Ed.), *Robotic Fabrication in Architecture, Art, and Design*, SpringerWienNewYork: New York 206-213.

Duff, E. (2006). "Tracking a vehicle from a rotating platform with a scanning range laser." *Proceedings of the Australian Conference on Robotics and Automation, 12.*

Engel, J., Schops, T., and Cremers, D. (2014). "LSD-SLAM: Large-scale direct monocular SLAM." In *Computer Vision--ECCV 2014* (834--849). Springer.

Enjarini, B., and Graser, A. (2012). "Planar segmentation from depth images using gradient of depth feature." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 4668-4674.

Everett, J. G. (1991). *"Construction automation--basic task selection and development of the CRANIUM."* Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Everett, J. G., and Slocum, A. (1994). "Automation and robotics opportunities: construction versus manufacturing." *Journal of construction engineering and management, 120*(2), 443--452.

Feng, C., and Kamat, V. R. (2012). "Augmented Reality Markers as Spatial Indices for Indoor Mobile AECFM Applications." *Proceedings of the 2012 Conference on Construction Applications of Virtual Reality*, Taipei, Taiwan, 235-242.

Feng, C., and Kamat, V. R. (2013). "Plane Registration Leveraged by Global Constraints for Context-Aware AEC Applications." *Computer-Aided Civil and Infrastructure Engineering, 28*(5), 325-343.

Feng, C., Dong, S., Lundeen, K. M., Xiao, Y., and Kamat, V. R. (2015). "Vision-Based Articulated Machine Pose Estimation for Excavation Monitoring and Guidance." *Proceedings of the 32nd International Symposium on Automation and Robotics in Construction.*, Oulu, Finland

Feng, C., Xiao, Y., Willette, A., McGee, W., and Kamat, V. R. (2014). "Towards Autonomous Robotic In-Situ Assembly on Unstructured Construction Sites Using Monocular Vision." *Proceedings of the 31st International Symposium on Automation and Robotics in Construction*, Sydney, Australia, 163-170.

Fernandez-Moral, E., Mayol-Cuevas, W., Arevalo, V., and Gonz\'alez-Jim\'enez, J. (2013). "Fast place recognition with plane-based maps." *IEEE International Conference on Robotics and Automation (ICRA)*, 2719--2724.

Fischler, M., and Bolles, R. (1981). "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM, 24*(6), 381-395.

Forsberg, J., Aarenstrup, R., and Wernersson, A. (1997). "A construction Robot for Autonomous Plastering of Walls and Ceilings." *Proceedings of the 14th International Symposium on Automation and Robotics in Construction (ISARC)*, 8-11.

Fukuda, T., Fujisawa, Y., Arai, F., Muro, H., Hoshino, K., Miyazaki, K., . . . Uehara, K. (1991). "A new robotic manipulator in construction based on man-robot cooperation work." *Proc. of the 8th International Symposium on Automation and Robotics in Construction*, 239--245.

Gambao, E., Balaguer, C., and Gebhart, F. (2000). "Robot assembly system for computer-integrated construction." *Automation in Construction, 9*(5), 479-487.

Gambao, E., Hernando, M., and Surdilovic, D. (2012). "A new generation of collaborative robots for material handling." *Gerontechnology, 11*(2), 368.

Georgiev, K., Creed, R. T., and Lakaemper, R. (2011). "Fast plane extraction in 3D range data based on line segments." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 3808-3815.

Ghassemi, F., Tafazoli, S., Lawrence, P., and Hashtrudi-Zaad, K. (2002). "An accelerometer-based joint angle sensor for heavy-duty manipulators." *Proceedings of IEEE International Conference on Robotics and Automation, 2*, 1771--1776.

Golparvar-Fard, M., Pena-Mora, F., Arboleda, C. A., and Lee, S. (2009). "Visualization of construction progress monitoring with 4D simulation model overlaid on time-lapsed photographs." *Journal of Computing in Civil Engineering, 23*(6), 391-404.

Gotardo, P., Bellon, O., and Silva, L. (2003). "Range image segmentation by surface extraction using an improved robust estimator." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2*, II--33.

Groves, P. (2011). "Shadow matching: A new GNSS positioning technique for urban canyons." *Journal of Navigation, 64*(03), 417--430.

Hahnel, D., Burgard, W., and Thrun, S. (2003). "Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot." *Robotics and Autonomous Systems, 44*(1), 15-27.

Han, C. (2011). "Human-Robot cooperation technology--An ideal midway solution heading toward the future of robotics and automation in construction." *Proceedings of the 28th ISARC International Symposium on Automation and Robotics in Construction*, 13-18.

Han, S., and Lee, S. (2013). "A vision-based motion capture and recognition framework for behavior-based safety management." *Automation in Construction, 35*, 131–141.

Han, S., Cho, H., Kim, S., Jung, J., and Heo, J. (2012). "Automated and efficient method for extraction of tunnel cross sections using terrestrial laser scanned data." *Journal of Computing in Civil Engineering, 27*(3), 274--281.

Hansson, A., and Servin, M. (2010). "Semi-autonomous shared control of large-scale manipulator arms." *Control Engineering Practice, 18*(9), 1069--1076.

Harris, C., and Stephens, M. (1988). "A combined corner and edge detector." *Alvey vision conference*, Manchester, UK, *15*, 50.

Hartley, R., and Zisserman, A. (2000). *"Multiple view geometry in computer vision."* Cambridge University Press.

Helm, V., Ercan, S., Gramazio, F., and Kohler, M. (2012). "In-Situ Robotic Construction: Extending the Digital Fabrication Chain in Architecture." In M. Cabrinha, J. Kelly

Johnon, and K. Steinfeld (Ed.), *Synthetic Digital Ecologies: Proceedings of the 32nd Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, San Francisco, USA, 169-176.

Hel-Or, Y., and Werman, M. (1994). "Model based pose estimation of articulated and constrained objects." *ECCV*, Springer 262--273.

Holz, D., and Behnke, S. (2012). "Fast Range Image Segmentation and Smoothing using Approximate Surface Reconstruction and Region Growing." *Intelligent Autonomous Systems*, 61-73.

Holz, D., Holzer, S., Rusu, R. B., and Behnke, S. (2011). "Real-Time Plane Segmentation using RGB-D Cameras." *Proc. RoboCup Symposium*.

Holzer, S., Rusu, R., Dixon, M., Gedikli, S., and Navab, N. (2012). "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 2684-2689.

Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P., Bunke, H., Goldgof, D., . . . Fisher, R. (1996). "An experimental comparison of range image segmentation algorithms." *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*(7), 673--689.

Hulik, R., Beran, V., Spanel, M., Krsek, P., and Smrz, P. (2012). "Fast and Accurate Plane Segmentation in Depth Maps for Indoor Scenes." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 1665-1670.

Kahane, B., and Rosenfeld, Y. (2004a). "Real-time "Sense-and-Act" operation for construction robots." *Automation in construction, 13*(6), 751-764.

Kahane, B., and Rosenfeld, Y. (2004b). "Balancing Human-and-Robot Integration in Building Tasks." *Computer-Aided Civil and Infrastructure Engineering, 19*(6), 393-410.

Kalkusch, M., Lidy, T., Knapp, M., Reitmayr, G., Kaufmann, H., and Schmalstieg, D. (2002). "Structured visual markers for indoor pathfinding." *Proceedings of the First IEEE International Workshop On Augmented Reality Toolkit*, Darmstadt, Germany, 8--pp.

Kamat, V., and El-Tawil, S. (2007). "Evaluation of augmented reality for rapid assessment of earthquake-induced building damage." *Journal of computing in civil engineering, 21*, 303-310.

Kashani, A., Owen, W., Himmelman, N., Lawrence, P., and Hall, R. (2010). "Laser Scanner-based End-effector Tracking and Joint Variable Extraction for Heavy Machinery." *The International Journal of Robotics Research, 29*(10), 1338-1352.

Kato, H., and Billinghurst, M. (1999). "Marker tracking and hmd calibration for a video-based augmented reality conferencing system." *Proceedings of 2nd IEEE and ACM International Workshop on Augmented Reality*, 85-94.

Khoshelham, K., and Elberink, S. O. (2012). "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications." *Sensors, 12*(2), 1437-1454.

Khoury, H., and Kamat, V. (2009). "Indoor User Localization for Context-Aware Information Retrieval in Construction Projects." *Automation in Construction, 18*(4), 444-457.

Kim, Y.-S., and Haas, C. T. (2000). "A model for automation of infrastructure maintenance using representational forms." *Automation in Construction, 10*(1), 57-68.

Klein, G., and Murray, D. (2007). "Parallel tracking and mapping for small AR workspaces." *Proceedings of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 225-234.

Kopp, J. (2008). "Efficient numerical diagonalization of hermitian 3x3 matrices." *International Journal of Modern Physics C, 19*(03), 523--548.

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). "g2o: A general framework for graph optimization." *IEEE International Conference on Robotics and Automation (ICRA), 2011*, 3607--3613.

Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., and Kleiner, A. (2009). "On measuring the accuracy of SLAM algorithms." *Autonomous Robots, 27*(4), 387-407.

Ladikos, A., Benhimane, S., and Navab, N. (2007). "A real-time tracking system combining template-based and feature-based approaches." *Proceedings of International Conference on Computer Vision Theory and Applications.*

LCCI/KPMG. (2014). *"Skills to Build: LCCI/KPMG Construction Skills Index 2014."* Retrieved February 6, 2015, from http://www.kpmg.com/uk/en/issuesandinsights/articlespublications/pages/construction-skills-index-2014.aspx

Lee, S., Kang, M.-S., Shin, D.-S., and Han, C.-S. (2012). "Estimation with applications to dynamic status of an excavator without renovation." *Gerontechnology*, *11*(2), 414.

Lee, T.-k., Lim, S., Lee, S., An, S., and Oh, S.-y. (2012). "Indoor mapping using planes extracted from noisy RGB-D sensors." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 1727-1733.

Lepetit, V., and Fua, P. (2005). "Monocular Model-Based 3D Tracking of Rigid Objects." *Foundations and Trends in Computer Graphics and Vision*, 1-89.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). "Epnp: An accurate O(n) solution to the pnp problem." *International journal of computer vision*, *81*(2), 155--166.

Li, B., Salter, J., Dempster, A., Rizos, C., and others. (2006). "Indoor positioning techniques based on wireless LAN." *Proceedings of the first IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*, Sydney, Australia, 13-16.

Lin, L.-H., Lawrence, P., and Hall, R. (2013). "Robust outdoor stereo vision SLAM for heavy machine rotation sensing." *Machine vision and applications*, *24*(1), 205--226.

Lindeberg, T. (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision*, *30*(2), 79-116.

Lindsey, Q., Mellinger, D., and Kumar, V. (2012). "Construction with quadrotor teams." *Autonomous Robots*, *33*(3), 323-336.

Lowe, D. (2004). "Distinctive image features from scale-invariant keypoints." *International journal of computer vision*, *60*(2), 91-110.

Lucas, B., and Kanade, T. (1981). "An Iterative Image Registration Technique with an Application to Stereo Vision." *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), 3*, 674-679.

Luhmann, T. (2009). "Precision potential of photogrammetric 6DOF pose estimation with a single camera." *ISPRS Journal of Photogrammetry and Remote Sensing, 64*(3), 275--284.

Malis, E., and Vargas, M. (2007). *"Deeper understanding of the homography decomposition for vision-base control."* Technical Report.

May, A., Mitchell, V., Bowden, S., and Thorpe, T. (2005). "Opportunities and challenges for location aware computing in the construction industry." *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, 255-258.

Memarzadeh, M., Heydarian, A., Golparvar-Fard, M., and Niebles, J. (2012). "Real-time and automated recognition and 2D tracking of Construction workers and equipment from Site video streams." *Int. Workshop on Computing in Civil Engineering.*

Microsoft. (2012). *"Kinect Sensor."* Retrieved April 5, 2015, from Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/hh438998.aspx

Microsoft. (2015). *"Kinect for Windows Sensor Components and Specifications."* Retrieved April 5, 2015, from Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/jj131033.aspx

Milberg, C., and Tommelein, I. (2003). "Role of tolerances and process capability data in product and process design integration." *In Proceedings of Construction Research Congress*, *8*.

Milberg, C., and Tommelein, I. D. (2005). "Application of Tolerance Mapping in AEC Systems." *In Proceedings of Construction Research Congress.*

Mitchell, W. J. (2001). "Roll Over Euclid: How Frank Gehry Designs and Builds." In *Frank Gehry, Architect* (354). New York: The Solomon R. Guggenheim Foundation.

Navon, R. (2000). "Process and quality control with a video camera, for a floor-tilling robot." *Automation in construction, 10*(1), 113-125.

Nguyen, V., Martinelli, A., Tomatis, N., and Siegwart, R. (2005). "A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 1929-1934.

Oehler, B., Stueckler, J., Welle, J., Schulz, D., and Behnke, S. (2011). "Efficient multi-resolution plane segmentation of 3d point clouds." *Intelligent Robotics and Applications*, 145-156.

Olson, E. (2010). *The APRIL robotics toolkit*. Retrieved September 12, 2013, from http://april.eecs.umich.edu/wiki/index.php/Main_Page\#April_Robotics_Toolkit

Olson, E. (2011). "AprilTag: A robust and flexible visual fiducial system." *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, 3400-3407.

Ozuysal, M., Fua, P., and Lepetit, V. (2007). "Fast keypoint recognition in ten lines of code." *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1-8.

Pathak, K., Birk, A., Vaskevicius, N., and Poppinga, J. (2010). "Fast registration based on noisy planes with unknown correspondences for 3D mapping." *IEEE Transactions on Robotics, 26*(3), 424-441.

Paul, G., Liu, D., Kirchner, N., and Webb, S. (2007). "Safe and efficient autonomous exploration technique for 3D mapping of a complex bridge maintenance environment." *Proc. 24th International Symposium on Automation and Robotics in Construction*, 99--104.

Pigram, D., Maxwell, I., McGee, W., Hagenhofer-Daniell, B., and Vasey, L. (2012). "Protocols, Pathways, and Production." In S. Brell-Cokcan, and J. Braumann (Ed.), *Robotic Fabrication in Architecture, Art, and Design*, SpringerWienNewYork: New York 143-147.

Point Grey. (2015). *"Spherical Vision."* Retrieved April 5, 2015, from Point Grey: http://www.ptgrey.com/360-degree-spherical-camera-systems

Poppinga, J., Vaskevicius, N., Birk, A., and Pathak, K. (2008). "Fast plane detection and polygonalization in noisy 3D range images." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 3378-3383.

Powell, M. (1978). "A fast algorithm for nonlinearly constrained optimization calculations." In *Numerical analysis* (144--157). Springer.

Pritschow, G., Dalacker, M., and Kurz, J. (1993). "Configurable control system of a mobile robot for on-site construction of masonry." *Proceedings of the International Smyposium on Automation and Robotics for Construction and Mining.*

Pustka, D., Huber, M., Bauer, M., and Klinker, G. (2006). "Spatial relationship patterns: Elements of reusable tracking and calibration systems." *IEEE/ACM International Symposium on Mixed and Augmented Reality, 2006*, 88--97.

Rezazadeh Azar, E., and McCabe, B. (2012). "Part based model and spatial-temporal reasoning to recognize hydraulic excavators in construction images and videos." *Automation in construction, 24*, 194-202.

Richardson, A., Strom, J., and Olson, E. (2013). "AprilCal: Assisted and repeatable camera calibration." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1814--1821.

Rojas, E., and Aramvareekul, P. (2003). "Is Construction Labor Productivity Really Declining?" *Journal of Construction Engineering and Management, 129*(1), 41-46.

Rosten, E., and Drummond, T. (2006). "Machine learning for high-speed corner detection." *Proceedings of the 9th European Conference on Computer Vision*, Springer 430-443.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). "ORB: an efficient alternative to SIFT or SURF." *Proceedings of the 2011 IEEE International Conference on Computer Vision*, 2564-2571.

Saidi, K., O'Brien, J., and Lytle, A. (2008). "Robotics in Construction." In *Springer Handbook of Robotics* (1079-1099). Springer Berlin Heidelberg.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., and Davison, A. J. (2013). "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects." *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR).*

Sanpechuda, T., and Kovavisaruch, L. (2008). "A review of RFID localization: Applications and techniques." *Proceedings of 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, Krabi, 769-772.

Schnabel, R., Wahl, R., and Klein, R. (2007). "Efficient RANSAC for Point-Cloud Shape Detection." *Computer Graphics Forum, 26*(2), 214-226.

Shi, J., and Tomasi, C. (1994). "Good features to track." *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 593-600.

Shohet, I. M., and Rosenfeld, Y. (1997). "Robotic mapping of building interior—precision analysis." *Automation in construction, 7*(1), 1-12.

Simon, G., Fitzgibbon, A., and Zisserman, A. (2000). "Markerless tracking using planar structures in the scene." *Proceedings of IEEE and ACM International Symposium on Augmented Reality*, 120-128.

Slocum, A., and Schena, B. (1988). "Blockbot: A robot to automate construction of cement block walls." *Robotics and Autonomous Systems, 4*(2), 111--129.

Snavely, N., Seitz, S., and Szeliski, R. (2006). "Photo tourism: exploring photo collections in 3D." *ACM transactions on graphics (TOG), 25*, 835--846.

Son, H., Kim, C., and Kim, C. (2014). "Fully automated as-built 3D pipeline extraction method from laser-scanned data based on curvature computation." *Journal of Computing in Civil Engineering*.

Son, H., Kim, C., Kim, H., Han, S. H., and Kim, M. K. (2010). "Trend analysis of research and development on automation and robotics technology in the construction industry." *KSCE Journal of Civil Engineering, 14*(2), 131-139.

Strom, J., Richardson, A., and Olson, E. (2010). "Graph-based Segmentation for Colored 3D Laser Point Clouds." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS).*

Sturm, P., and Triggs, B. (1996). "A factorization based algorithm for multi-image projective structure and motion." *Computer Vision—ECCV'96*, 709--720.

Taguchi, Y., Jian, Y.-D., Ramalingam, S., and Feng, C. (2013). "Point-Plane SLAM for Hand-Held 3D Sensors." *Proceedings of IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 5182-5189.

Tang, P., Huber, D., Akinci, B., Lipman, R., and Lytle, A. (2010). "Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques." *Automation in construction, 19*(7), 829--843.

Taylor, S., Rosten, E., and Drummond, T. (2009). "Robust feature matching in 2.3 microsecond." *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 15-22.

Teizer, J., Venugopal, M., and Walia, A. (2008). "Ultrawideband for automated real-time three-dimensional location sensing for workforce, equipment, and material positioning and tracking." *Transportation Research Record: Journal of the Transportation Research Board, 2081*, 56-64.

Thrun, S. (2008). "Simultaneous localization and mapping." *Robotics and cognitive approaches to spatial mapping*, 13-41.

Torr, P., and Zisserman, A. (2000). "MLESAC: A new robust estimator with application to estimating image geometry." *Computer Vision and Image Understanding, 78*(1), 138-156.

Trevor, A. J., Rogers III, J. G., and Christensen, H. I. (2012). "Planar Surface SLAM with 3D and 2D Sensors." *Proc. IEEE Int'l Conf. Robotics Automation (ICRA)*, 3041-3048.

Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (2000). "Bundle adjustment—a modern synthesis." In *Vision algorithms: theory and practice* (298--372). Springer.

Tsai, R., and Lenz, R. (1989). "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration." *IEEE Transactions on Robotics and Automation, 5*(3), 345--358.

U.S. Environmental Protection Agency. (1989). *"Report to Congress on indoor air quality: Volume 2."* EPA/400/1-89/001C, Washington, DC.

US DOT PHMSA. (2015). *"Pipeline Incident 20 Year Trends."* Retrieved February 4, 2015, from http://www.phmsa.dot.gov/pipeline/library/datastatistics/pipelineincidenttrends

Wagner, D., and Schmalstieg, D. (2003). "First steps towards handheld augmented reality." *Proceedings of the 7th International Symposium on Wearable Computers (ISWC'2003)*, White Plains, New York, USA, 127--127.

Wagner, D., Langlotz, T., and Schmalstieg, D. (2008). "Robust and unobtrusive marker tracking on mobile phones." *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 121-124.

Wang, T., Chen, L., and Chen, Q. (2013). "A graph-based plane segmentation approach for noisy point clouds." *Proc. Chinese Control and Decision Conference (CCDC)*, 3770-3775.

Weingarten, J., and Siegwart, R. (2006). "3D SLAM using planar segments." *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 3062-3067.

Willmann, J., Augugliaro, F., Cadalbert, T., D'Andrea, R., Gramazio, F., and Kohler, M. (2012). "Aerial Robotic Construction Towards a New Field of Architectural Research." *International Journal of Architectural Computing, 10*(3), 439-460.

Yang, J., Vela, P., Teizer, J., and Shi, Z. (2011). "Vision-based crane tracking for understanding construction activity." *Proc. ASCE IWCCE*, 258--265.

Yu, S.-N., Ryu, B.-G., Lim, S.-J., Kim, C.-J., Kang, M.-K., and Han, C.-S. (2009). "Feasibility verification of brick-laying robot using manipulation trajectory and the laying pattern optimization." *Automation in Construction, 18*(5), 644--655.

Zhang, G., Karasev, P., Brilakis, I., and Vela, P. (2012). "A Sparsity-Inducing Optimization Algorithm for the Extraction of Planar Structures in Noisy Point-Cloud Data." *Proc. Computing in Civil Engineering*, 317-324.

Zhang, Z. (2000). "A flexible new technique for camera calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(11), 1330--1334.

Zhu, Z., and Donia, S. (2013). "Potentials of RGB-D cameras in as-built indoor environments modeling." *Proceedings of the ASCE International Workshop on Computing in Civil Engineering*, Los Angeles, CA, 23--25.