AUTOMATED 3D VISION-BASED TRACKING OF CONSTRUCTION ENTITIES

A Dissertation
Presented to
The Academic Faculty

by

Man-Woo Park

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Civil and Environmental Engineering

Georgia Institute of Technology
December, 2012

AUTOMATED 3D VISION-BASED TRACKING OF CONSTRUCTION ENTITIES

Approved by:

Dr. Ioannis Brilakis, Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Jochen Teizer
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Randall L. Guensler
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Patricio A. Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Michael P. Hunter
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date Approved: July 13, 2012

ACKNOWLEDGEMENTS

First and above all, I praise God, the almighty for providing me this opportunity and granting me the capability to proceed successfully. This thesis appears in its current form due to the assistance and guidance of several people. I would therefore like to offer my sincere thanks to all of them.

First and foremost, I offer my sincere gratitude to my advisor, Dr. Ioannis Brilakis. He has supported me during my doctoral studies with his patience and knowledge whilst allowing me the room to work in my own way. His perpetual enthusiasms in research have motivated me, and as a result, my research life at Georgia Tech is rewarding.

I am delighted to have Dr. Randall Guensler, Dr. Michael Hunter, Dr. Jochen Teizer, and Dr. Patricio Vela become my dissertation committee members. Their expertise and experience broaden my perspectives and nourish my intellectual maturity. I gratefully acknowledge them for their valuable guidance and comments. I would like to particularly thank Dr. Guensler who gave me an opportunity to participate in the transportation project and supported me with constructive advice and suggestions.

Many thanks go to my current and former lab mates in the Construction Information Technology Laboratory. They make the lab a convivial place to work. They are: Dr. Zhenhua Zhu, Dr. Christian Koch, Dr. Fei Dai, Ms. Gauri Jog, Mr. Habib Fathi, Mr. Abbas Rashidi, Ms. Stephanie German, Mr. Evangelos Palinginis, Ms. Stefania Radopoulou, Ms. Linda Hui, Ms. Atefe Makhmalbaf, Ms. Aswathy Sivaram, Mr. Keitaro Kamiya, and Mr. Matthew Sandidge. I really thank you all for all the fun and discussions we have had in the last four years.

I would like to express my gratitude to Korean students in CEE, who were always there to encourage and comfort me whenever I had a hard time and was in trouble. They have shared joys and sorrows with me all the time. I would have never been able to make it through this point without all my friends in CEE.

I owe my loving thanks to my wife, Doyoon Lee. She has lost a lot due to my research abroad. She had a cancer surgery and went through all treatments by herself. I cannot recall it without tears. I really appreciate her patience to get over it. She gave me encouragement even during the treatments. Her patient love enabled me to complete this work. I would like to thank my parents for their unconditional support, both financially and emotionally throughout my degree. The patience and understanding shown by my mother and father during the honours year is greatly appreciated. They always showed their confidence in me which encouraged and strengthen me through my life. Last but not least, my loving thanks are due to my brother and his family. I love you all more than words can express.

I will not forget my doctoral study years at Georgia Tech. Again, I would like thank everybody who is important to the successful realization of this dissertation. Also, I like to thank the National Science Foundation for its indirect and direct financial support on this research under grant #0933931 and #0904109.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In construction sites, tracking project-related entities such as construction equipment, materials, and personnel provides useful information for productivity measurement, progress monitoring, on-site safety enhancement, and activity sequence analysis. Radio frequency technologies such as Global Positioning Systems (GPS), Radio Frequency Identification (RFID) and Ultra Wide Band (UWB) are commonly used for this purpose. However, on large-scale congested sites, deploying, maintaining and removing such systems can be costly and time-consuming because radio frequency technologies require tagging each entity to track. In addition, privacy issues can arise from tagging construction workers, which often limits the usability of these technologies on construction sites. A vision-based approach that can track moving objects in camera views can resolve these problems.

The purpose of this research is to investigate the vision-based tracking system that holds promise to overcome the limitations of existing radio frequency technologies for large-scale, congested sites. The proposed method use videos from static cameras. Stereo camera system is employed for tracking of construction entities in 3D. Once the cameras are fixed on the site, intrinsic and extrinsic camera parameters are discovered through camera calibration. The method automatically detects and tracks interested objects such as workers and equipment in each camera view, which generates 2D pixel coordinates of tracked objects. The 2D pixel coordinates are converted to 3D real-world coordinates based on calibration. The method proposed in this research was implemented in .NET Framework 4.0 environment, and tested on the real videos of construction sites. The test

results indicated that the methods could locate construction entities with accuracy comparable to GPS.

CHAPTER 1

INTRODUCTION

This research tests the feasibility of using rapidly innovative in computer vision and high definition cameras for the purpose of tracking project-related entities in construction sites. Tracking of construction entities provides useful information for various site management tasks including safety management, productivity analysis, and progress monitoring and activity sequence analysis. The rest of this chapter consists of the research motivation, objectives, methodology, and the organization of this dissertation.

1.1 Background and Motivation

One of the greatest challenges of engineering noted by National Academy of Engineering is the need for higher level of automation in construction (National Academy of Engineering 2008). Even though there are great opportunities to enhance construction engineering through the state-of-the-art technologies in computer science and robotics, relatively conservative attitudes of construction industry have slowed down the migration of those technologies (Anumba 1998; Shapira and Rosenfeld 2011; Park et al. 2004). In addition to the tendency to be reluctant to adopt new technologies and ideas, complexity and diversity inherent in most construction projects make it difficult to directly apply the technologies to construction. However, many researchers have recently endeavored to change the trends and introduce new technologies for increased efficiency and accuracy in all aspects of construction from planning and design (Osman et al. 2003; Veeramani et al. 1998), through construction of the facility (Bernold 2007; Bock 2008), its operation and maintenance (Ko 2009; Victores et al. 2011). Significant efforts have been made to

automate acquisition of real time site information to provide an additional layer of control over the project. Automated tracking is one of the topics in this area.

Tracking provides location data of construction entities (e.g. personnel, equipment and materials) over time. This spatiotemporal information is useful for various construction applications such as productivity measurement, travel path conflict detection, safety management, progress monitoring, and activity sequence analysis, etc. For a simple example, as in Figure 1.1, tracking of a concrete bucket can provide the duration of a crane cycle for moving the concrete bucket, which is useful for productivity measurement. Also, the destination of the concrete bucket can identify the activity being processed. Furthermore, workers who may pass underneath the moving concrete bucket can be detected and warned if they are tracked with automated systems.

Tracking methods applied in construction industry include Radio Frequency Identification (RFID), Ultra Wide Band (UWB), and global positioning system (GPS). These technologies have an established record of tracking resources robustly for most

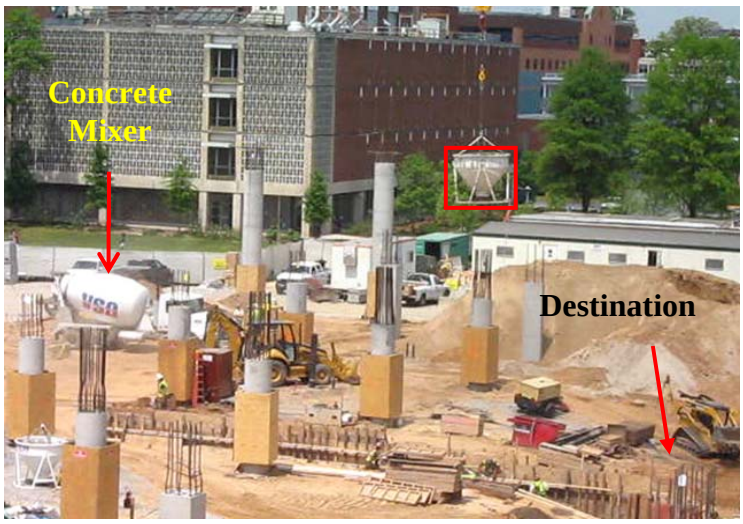


Figure 1.1: Tracking of a concrete bucket

construction scenarios, such as proactive work zone safety, and material localization and installation. RFID has been employed for tracking precast concrete elements from their casting to assembly, and UWB is well-known for indoor tracking of construction materials. The latest construction heavy equipment is equipped with GPS units by the manufacturer for better control of its tasks and tracking. However, the requirement for a separate tag/sensor on each entity to be tracked can limit their applicability in large-scale, congested construction sites where a large number of entities need to be tagged. Also, privacy issues can arise out of tagging workers who may be uncomfortable with being tagged and tracked. In such cases, there is a need for a more efficient tracking technology in terms of time and cost, that accurately locates construction entities, maximizes the number of entities to track, and minimizes labor costs.

Surveillance cameras are often deployed in construction sites to monitor a broader perspective of construction activities and operations (Bohn and Teizer 2009). Many contractors understand advantages of utilizing the on-site cameras. Cheap and high resolution cameras can capture construction video while extensive data storage and network capacities enables users to monitor the sites in real-time. Such a camera network system alleviates the need for personnel to walk around the site and take photos. Allowing for access to several site views at different locations, the camera network facilitates monitoring of project progress and site safety. Given that on-site construction cameras are used in an increasing number of construction sites, the vision-based technology has a high potential to add the values of the cameras as an efficient tracking sensor. Archived video data can be used for visualization of construction processes, remote visual inspections, and progress monitoring. Also, important evidence, in the case

of litigation associated with the project, can be found from the video data (Brilakis 2005). With the advent of robust computer vision algorithms, it is possible to aumatically extract information of on-site activities from video streams (Gong and Caldas 2010).

Vision-based tracking can provide real-time visual information of construction job sites. Vision-based tracking tracks moving objects in a video on the basis of their visual patterns and motion patterns. Vision-based tracking is unobtrusive, since it can track multiple entities concurrently without installing sensors and ID tags of any kind on the tracked entities. This advantage makes this technology highly applicable in dynamic, busy construction sites, where large numbers of equipment, personnel and materials are involved, and more desirable from personnel who wish to avoid being "tagged" with sensors. Also, Vision-based tracking features a vast size of traceable area. In addition, due to the simplicity, commercial availability and low costs associated with video equipment, vision-based tracking can be clearly profitable in various construction operations (Caldas et al. 2004). However, the general vision trackers have two limitations to overcome. First, vision-based tracking only provides 2D pixel coordinates which does not reflect real motions well. The 2D results may be useful in cases where predefined measurements on an entity's trajectory are available as in Gong and Caldas' research work (2010). However, the lack of depth information limits application of the general vision trackers. Spatial distance between two entities as well as any motion in the perpendicular direction to the camera image plane is not measurable. To obtain 3D location information and be competitive to other radio frequency technologies, additional procedures are required. Second, to initialize the trackers, it is necessary to manually

mark the region of interested objects to be tracked. Therefore, an automated way of initialization is demanding to get rid of the burdensome manual processes.

1.2 Research Hypothesis

The hypothesis behind this research is: real world 4D coordinates (3D positions across time) of construction entities can be automatically retrieved from videos by creating and applying appropriate algorithms of visual pattern recognition, digital filtering, camera calibration, and machine learning. The proposed vision-based tracking method promises to determine the spatial location of objects of standard sizes and shapes, across time, without installation of any sensors so that it can be considered an appropriate alternative to other technologies such as GPS, RFID, and UWB. Specifically, stereo camera system is considered, which allows the acquisition of 3D spatial information.

The emphasis in this research is placed on creating or selecting capable methods for 1) automatically detecting construction entities in video streams when they first appear, 2) tracking the detected entities in the subsequent frames which provides 2D location across time, 3) integrating detection and tracking algorithms for robust and stable 2D localization, and 4) calibrating stereo camera system and triangulating 2D pixel coordinates from two camera views to obtain depth information and calculate 3D real-world coordinates for each frame. The developed methods need to be validated on the videos collected in real construction sites, and their corresponding performance needs to be measured with appropriate performance metrics.

1.3 Objectives and Scope

There are various types of construction entities involved in most projects such as construction equipment, workers, and materials. Moreover, equipment (e.g. tuck, loader,

excavator, etc.) and materials (precast concrete elements, steel beams and plates, bricks, etc.) can be classified into a variety of categories. Out of such varied entities, this research effort is focused on construction workers and equipment. Specifically, the wheel loader, which is one of the most frequently used equipment for construction earth works, is mainly dealt with as a representative of equipment. Construction materials are not the main focus since RFID and UWB are supposed to work with high accuracy and efficiency (Teizer et al. 2008). RFID is known to be able to track materials from manufacturing to installation, and UWB is typically devised for tracking in indoor environment. Even though the needs for attaching tags and deploying associated facilities on the site limit their application to a certain degree, the potential application of vision-based tracking of materials is very limited when compared to tracking of other entities. For example, radio frequency technologies better fit to tracking of precast concrete elements from an off-site factory to the site because the coverage area of vision tracking is limited to only the construction site. Furthermore, material detection has been an independent active research topic and has been investigated in previous works (Brilakis and Soibelman 2008; Zhu and Brilakis 2010).

The main objective of this research is to test feasibility of acquiring 4D spatio-temporal information of interested entities on construction sites (i.e. construction equipment and workers) from video streams with an enhanced degree of automation. To support this main objective, specifically, the research effort is divided into the following three sub-objectives.

1. Create novel detection methods to automatically recognize and localize construction entities for the purpose of initializing the tracking process. Once an

object of the interested type appears first in the view, it has to be detected so that the tracking process is triggered to track the detected object in subsequent frames. For this purpose, image features that effectively characterize visual patterns of construction equipment and workers are investigated and integrated into a detection process.

2. Find the best methods to track construction entities in 2D. Since there exist numerous methods that are capable of tracking construction entities, the research performs appropriate categorization and thorough comparison to assess the most effective 2D trackers.
3. Integrate detection and 2D tracking methods in a single line so that they compliment each other resulting in more stable and reliable process.
4. Investigate the way of calibrating stereo cameras for effective monitoring construction sites. Generally, the longer the baseline (distance between cameras), the higher the accuracy that can be achieved. Also, the longer the distance from cameras to objects, the lower the accuracy. Therefore, a long baseline is necessary to track objects at long distance from the cameras in large scale construction sites. Calibration methods compatible with such conditions are investigated and tested.
5. Discuss the research findings.

It should be noted that real-time processing is not a concern in this research. This research devotes attention more on the feasibility of the automated 4D data acquisition than on real-time processing. It is believed that real-time processing can be achieved in a commercializing step through program optimization or employing a GPU (Graphics Processing Unit). Besides, certain types of applications do not necessarily require real-

time information. While real-time processing is ultimately required for incident detection and safety applications, productivity analysis and activity sequence analysis can be performed in post processing of acquired data on a daily or weekly basis. Though the research seeks faster methods for each step, accuracy takes precedence over processing time. In this work, trajectory is one of the most important materials for various tasks of monitoring construction sites or highway traffic; hence, this research delves into how to extract trajectories of entities from video data regardless of application.

1.4 Methodology and Results

The research work in this study includes: 1) detecting construction workers, equipment and vehicles, 2) tracking all detected entities, 3) adjusting tracking results through the integration with detection results, and 4) calibrating cameras to convert 2D pixel coordinates to 3D real-world spatial coordinates. The entire framework is illustrated in Figure 1.2.

For monitoring construction sites, stereo camera system is proposed to estimate 3D spatial coordinates (Figure 1.2). Two cameras need to have partially overlapping viewing spectrums so that objects to be tracked can be seen in both views. Each camera view is continuously searched for new construction resources (i.e. workers, equipment, materials). Once a resource is detected, its image region is marked and given to a 2D vision tracker for tracking the entity in each subsequent frame. Such concurrent detection and tracking allows for handling occlusions (worker moving behind an object, and reappearing further down), and adjusting the tracker window to ensure robust tracking.

Detection is possible by characterizing the visual patterns of various entities and detecting the regions in the video frame that match the visual pattern. This is achieved by

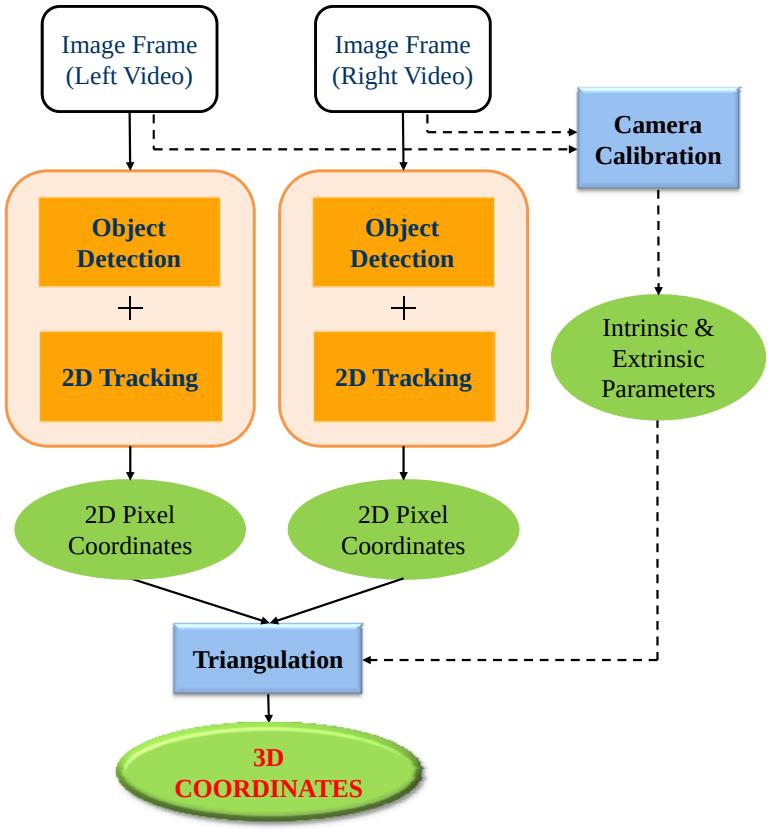


Figure 1.2: The overall framework of tracking construction entities using two cameras

identifying common visual patterns for each entity type (e.g. workers, wheel loaders, and dump trucks, etc.) by training the algorithm with images of similar objects taken from various view angles and under different illumination conditions (Chapter 3). Tracking the entity across time involves locating the same entity over time within the view (Chapter 4). The centroid of the detected and tracked entity is then calculated in a 2D pixel coordinate (x and y) at each frame. Generally, the 2D data are not enough to extract substantial information for most construction management tasks since it is unknown how far entities are located from the camera. Due to the lack of depth data (z), even approximate distance measurements between two entities (e.g. workers and mobile equipment) are not reliable, but necessary for e.g. safety management. Also, any

movement along the z axis is not measurable. Obtaining 3D coordinates is possible by correlating 2D coordinates from two or more views (Chapter 5). This is achieved by calibrating the on-site cameras once (at the beginning of the project) such that intrinsic (e.g. focal length) and extrinsic (translation and rotation between cameras) camera parameters are known. The 3D coordinates are then calculated by triangulating the 2D positions of each element from two or more views (Hartley and Zisserman 2004).

Every process of the proposed framework in Figure 1.2 is evaluated step by step. The detection process is evaluated on the basis of precision and time-delay since the main purpose of the detection process is to initiate the 2D tracking process (Chapter 3). In other words, it is desirable for the process to detect only interested types of objects immediately after the objects newly appear in the view (higher precision and shorter time-delay). The 2D tracking process is assessed based on the errors of centroid of tracked regions and the number of frames in which objects are successfully tracked (Chapter 4). The two metrics measure the accuracy of tracked position and the stability of the process. The error of centroid position is calculated only for the successful frames. Finally, overall 3D tracking performance is measured by errors of determined 3D locations (Chapter 6).

The proposed framework in Figure 1.2 is implemented using Microsoft Visual C# in .NET Framework 4.0 environment. EmguCV (Emgu CV 2010) was used as a wrapper to allow OpenCV (Bradski and Kaehler, 2008) functions to be called in the prototype. Both OpenCV and EmguCV are open source. Each step of the framework in Figure 1.2 is tested on construction videos and validated based on the defined metrics. As will be discussed in Chapter 6, the overall framework composed of the validated methods is

tested on videos of construction sites with controlled conditions. Tracking of a worker, a steel plate, and an SUV (Sport Utility Vehicle) at a construction site results in 3D location errors less than 0.7 m with 95% confidence level which are comparable to general GPS, and validates feasibility of the proposed methodology.

1.5 Dissertation Organization

The motivation, hypothesis, objectives, methodology and results, and contributions behind this research have been introduced. The remaining chapters in the dissertation are organized as follows.

Chapter 2 is a background literature review chapter. The chapter first outlines the current practices of monitoring construction resources and the state of research in optical sensor based tracking for construction applications. The current practices and the state of research in construction are followed by an overview of the fundamental knowledge and previous research studies in 1) vision-based object detection, 2) vision-based 2D tracking, and 3) stereo view geometry which this research plans to build on and augment for the purpose of tracking and localizing construction entities in 3D. The chapter ends with a summary on discussing the issues and limitations of current practices of tracking for construction as well as the potential of vision-based tracking.

Chapter 3 presents the first part of the proposed framework. Novel, automated methods for recognizing and locating construction workers and equipment are explained in separate subchapters. Each step of the methods is explained in detail. The implementation and experimental test results of construction worker detection are also presented. The performance metrics used to evaluate the method's performance are explained. This is then followed by the definition of 'construction worker' which is

important for validation. The chapter ends with an overview on the designed method, implementation, and experiments.

Chapter 4 describes the second part of the proposed frameworks which is 2D tracking. Details of a comparative study are explained, which aims to find the best 2D tracking methods for tracking construction entities. Independent and dependent vaiables of the comparison experiments are discussed. The reasons for removing contour-based methods from the main comparison expeirments are explained through investigation on their features and preliminary tests. Then, results of thorough comparison of template-based methods and point-based methods are presented. In the following subchapter, the methodology of integrating detection and tracking is explained. Experiments and results are also presented. The chapter ends with an overview on the designed method, implementation, and experiments.

Chapter 5 deals with the last part of the proposed framework, which is calculation of 3D coordinates. First, the methods of intrinsic and extrinsic calibraton, which are executed only once after cameras are setup, are described in detail one by one. Then, the selected triangulation method is explained.

Chapter 6 presents overall experiments performed to validate the proposed framework. The experiment design, implementation of the framework, and results are presented in detail from one after another. The experiments include tracking of a construction worker, a steel plate, and an SUV. The chapter ends with an overview on the experiment design and results.

In Chapter 7, the findings and contributions of this research are described. Initialization of general 2D vision tracking is automated by construction entity detection

methods. Template-based 2D tracking methods are determined as the most appropriate methods for tracking construction entities. Depth information and 3D spatial coordinates are obtained by emplying stereo camera systems, and a long baseline allows comparable accuracy to GPS. This chapter also discusses about next steps to extend and enhance the research – hybrid 2D tracking methods, real-time processing, and three- or four-view geometry.

CHAPTER 2

LITERATURE REVIEW

2.1 State of Practice in Monitoring Construction Resources

In recent years, state-of-the-art technologies are employed in practice for monitoring construction resources. Many researchers have investigated on these technologies for the purpose of site monitoring, and practical applications of the technologies have recently been reported by several contractors. This chapter introduces radio frequency technologies such as GPS (Global Positioning System), RFID (Radio Frequency Identification), and UWB (Ultra Wide Band) which are well-known for tracking construction entities.

2.1.1 Global Positioning System (GPS)

GPS is the most famous technology which is generally used to track a fleet of heavy equipment. It has become a well-established monitoring system in construction sites. The system consists of a constellation of satellites, GPS sensors mounted on each equipment asset, and a central module that communicates with the sensors. Each sensor captures the asset's location and transmits the data to the central module over the mobile network, which makes it possible to visualize the location of all equipment on a single map in real time (Eecke 2010; Henderson 2008). Numerous products of the GPS system have been emerged for tracking and monitoring construction equipment (Cable 2010; Engineering News-Record 2008a, 2008b). Equipment manufacturers are adopting the products to provide for their customers (Construction Equipment 2010). The GPS sensors are capable

of reporting locations as well as idle and work time, and odometer readings. Analyzing the reported data can bring about significant benefits for contractors such as reduced fuel budget, better equipment utilization, and timely maintenance, etc. (Eecke 2010). Theft protection is also available via user-defined curfews, perimeter alerts, and homing beacon function (Construction Equipment 2010). For instance, a Texas-based contractor had found a stolen rubber tire backhoe costing $85,000 via its GPS system (ForConstructionPros.Com 2006). Though GPS has been applied to outdoor construction practices like positioning of construction equipment (Oloufa et al. 2003) and vehicles (Lu et al. 2007), the maximum error of general GPS is about 10 m, and it can be reduced to 1 cm by using kinematic GPS with trade off in increased costs (Caldas et al. 2004). Moreover, a recent report states that construction business in the United States will face serious disruption to their GPS-involved operations because of 4G-LTE open wireless broadband network that incorporates nationwide satellite coverage (ForConstructionPros.Com 2011).

2.1.2 Radio Frequency Identification (RFID)

RFID-based systems are also widely used in construction projects. The systems consist of RFID tags, a reader, and a data management system. When a reader requests, each tag sends its own ID so that the reader can identify the unique object. It is generally used to access and track construction materials or workers. When a critical piece of material arrives on site or it is placed in the structure, a tag attached to or embedded in the piece is read to register the event with its ID and timestamp of accomplishment (Engineering News-Record 2008c). Engineering News-Record reported about a notable case in which Skanska made use of RFID technologies to track pre-cast structural elements from

casting to assembly in a football stadium construction (Sawyer 2008a). In the project, RFID tagging was integrated with BIM (Building Information Modeling). A 4D model is updated by uploading RFID scanned information (Yoders 2008). The system enabled project managers to easily monitor whether the elements were in the correct sequence. RFID-based systems have also been applied to labor and equipment control (Engineering News-Record 2008c; Sawyer 2008b). A system commercialized by the DoallTech corporation combines tags on ID cards with a digital camera in the reader to verify identities (Sawyer 2008b). The RFID-based system was reported to be used in more than 400 projects around the world (Engineering News-Record 2008c). Furthermore, RFID systems facilitate safety control. The systems have been utilized to prevent the collisions among tower cranes (Gomez 2007a). Also, fall protection equipment combined with a RFID system is now available, which enhances the effectiveness of safety control (Gomez 2007b).

Researchers have continuously made efforts on the application of RFID in construction. It has been applied to quality management (Wang 2008), prevention of collision accident (Elghamrawy and Boukamp 2010), and automatic identification of construction concepts (Chae and Yoshida 2010). Recently, Ko (2010) proposed a methodology of 3D sensing with RFID which locates the positions of various construction entities. However, the near-sighted effect prohibits its use in tracking applications. There have also been efforts to integrate RFID and GPS technology. Ergen et al. (Ergen et al. 2007) applied this combination to track precast pieces in a storage yard. These research efforts led to the practical use of RFID technologies in a construction project.

2.1.3 Ultra Wide Band (UWB)

UWB is another type of radio technology that can be applied to short-range communications. UWB is able to detect time-of-flight of the radio transmissions at various frequencies, which enables it to perform effectively in providing precision localization even in the presence of severe multipath effects (Fontana et al. 2003). Another advantage is the low average power requirement that results from the low pulse rate (Fontana, 2004). Teizer et al. (2007a) applied the UWB technology to construction. It was used for a material location tracking system with primary applications to active work zone safety. Its ability to provide accurate 3D locations in real-time is a definite benefit to tracking in construction sites.

2.2 State of Research in Optical Sensor Based Tracking for Construction

Vision technologies and laser technologies are attracting increasing interests for tracking in large-scale, congested sites because they are free of tags. A 3D range imaging/video camera (e.g. a Flash LADAR) provides not only the intensity but also the estimated range of the corresponding image area. When compared to 3D laser scanners which have been used in construction, the device is portable and inexpensive. Testing various kinds of data filtering, transformation and clustering algorithms, Gong and Caldas (2008) used 3D range cameras for spatial modeling. Teizer et al. (2007b) demonstrated tracking with 3D range cameras and the potential of its use for site safety enhancement. However, the low resolution and short range make it difficult to be applied to large-scale construction sites. Few tests have been executed in outdoor construction sites where the environments are more cluttered and less controlled. Also, it is reported that the reflectance of a surface

varies extremely even in indoor environments (Gächter et al. 2006). Moreover, when multiple cameras are used, they can interfere with each other (Fuchs 2010).

Remote-controlled web-based cameras are currently available and used for remote monitoring of construction sites (Engineering News-Record 2008d, 2008e; Gomez 2007). The cameras are controlled remotely through pan/tilt/zoom functions, and transmit video frames wirelessly to the central system. The wireless communication system enables one person to monitor an entire site, which substantially reduces security cost (Gomez 2007). Along with the increasing of use construction cameras, vision-based tracking has recently been investigated. Traditional 2D vision tracking is simply based on a sequence of images and can be a proper alternative to RFID methods because it removes the need for installing sensors and ID tags of any kind on the tracked entity. For this reason, this technology is (a) highly applicable in dynamic, busy construction sites, where large numbers of equipment, personnel and materials are involved, and (b) more desirable from personnel who wish to avoid being "tagged" with sensors. In Gruen's research (1997), it is highly regarded for its capability to measure a large number of particles with a high level of accuracy. Teizer and Vela (2009) investigated vision trackers for construction worker tracking, and Yang et al. (2010) proposed a vision tracker that can track multiple construction workers. Gong and Caldas (2010) validated that vision tracking can be applied to automate productivity analysis. However, in these works, the results were limited to 2D pixel coordinates and the entities to be tracked were manually marked.

2.3 Vision-Based Object Detection

This chapter presents a literature review on the computer vision algorithms which are used for object detection in this research. The purpose of object detection is to recognize

and localize an object category (e.g. face, vehicle, and animal) by image features which are common to all objects of the type. In this research, object detection plays an important role in localizing construction entities and vehicles in 2D. The challenge of object detection is to construct a feature template compatible to various appearances of the object category. Shape, color, and motion features are generally used for object detection.

Haar-like features (Viola and Jones 2001; Lienhart and Maydt 2002) and Histogram of Oriented Gradients (HOG) (Dalal and Triggs 2005; Zhu et al. 2006) are well-known shape features. Both features are on the basis of gradient values, but utilize them in different ways. Haar-like features are vectors of image gradients which are differences of intensities between adjacent regions. In order to detect various appearances of an object type, the features are trained through a machine learning process: Haar-like features of various appearances are extracted from abundant training images and the features are trained with machine learning algorithms. For example, Viola and Jones (2001) used vertical and horizontal Haar-like features with an AdaBoost (Adaptive Boosting) algorithm (Freund and Schapire 1997) for human face detection. Lienhart and Maydt (2002) introduced additional 45˚ Haar-like features to account for diagonal edges and enhance the detection. Even though training with AdaBoost takes several days depending on the processor specifications, the method allows for real time detection once the training is completed. The HOG feature is a collection of local histograms of gradient directions. It divides an image based on a grid of uniformly spaced blocks. For each block, a histogram is calculated by counting the occurrence of gradient orientations. Similar to Haar-like features, HOG features also need training with a large number of

images. Dalal and Triggs (2005) applied HOG features trained with SVM (Support Vector Machine) (Joachims 1999) for human detection. They showed the superiority of HOG features over Haar-like features in human detection. Zhu et al. (2006) sped up the human detection by using AdaBoost while retaining the equivalent accuracy.

Color is also a useful and intuitive feature for recognizing an object type. A color histogram (Swain and Ballard 1991) is one of the typical color features. Simple calculation and invariance to rotation and translation are its useful qualities for object detection. It is suitable for detecting an object in distinctive colors. However, the sensitivity to illumination and the lack of spatial information limits its applications. The color histogram has been broadly used for image segmentation and content-based image retrieval in which spatial information is not critical (Zhang et al. 2009; Huang et al. 2004). It has also been employed in tracking applications. In tracking processes, color histograms combined with HOG features were used as observation models of pedestrians (Sugano and Miyamoto 2009), hockey players (Lu et al. 2009), etc. However, color histograms are not appropriate for the detection of pedestrians or hockey players since their colors widely vary from person to person according to their clothing. In Lu et al.'s work (2009), Haar-like features are used for detection, and the color histogram is used exclusively for tracking the detected entities.

A set of eigen-images is a famous feature used for human face detection. It is usually combined with principal component analysis (PCA) (Turk and Pentland 1991) or Fisher's linear discriminant (FLD) (Belhumeur et al. 1997) which reduces the dimensionality of the feature. The feature template is obtained by computing the covariance of the pixels and removing meaningless components. It contains both color

and shape information as it exploits vectors of pixel values which are lined up based on spatial location. The features work well for recognizing human faces with various illumination conditions and various expressions. However, they have been tested mostly on frontal faces, and had troubles with recognizing the faces of different angles.

Background subtraction (Mcfarlane and Schofield 1995; Stauffer and Grimson 2000; Li et al. 2002) detects moving objects on the basis of their motion cues. This method can be applied only to fixed camera views. The static background scene is modeled by taking a dominant value for each pixel across a certain number of frames. The model has to be updated throughout the frames in order to reflect changes of illumination. Mcfarlane and Schofield (1995) presented an approximated median filter which estimates the background with a simple update process. It increases/decreases the background estimate if the pixel value of the new video frame is larger/smaller than the estimate. The MoG (Mixture of Gaussians) method (Stauffer and Grimson 2000) is one of the most popular background modeling methods. It models the background pixel values as a mixture of multiple Gaussian distributions. Moving objects are detected by thresholding the difference of pixel values between the current frame and the background. Li et al. (2002) screened out moving background objects such as wavering trees and shadows by modeling them with the color co-occurrence feature. Background subtraction is computationally efficient since it can detect all moving objects simultaneously regardless of their appearances or types. However, motion features are not suitable to identify object types. Furthermore, background subtraction may recognize multiple objects that partially overlap each other as a single object.

Recently, several approaches for detecting construction entities have been proposed. Jog et al. (2011) presented methods for detection of trucks. They used the Semantic Texton Forests approach (Shotton et al. 2008), which detects objects through segmentation and classification. Though the method correctly marked regions of trucks with high accuracy, it is not capable of differentiate partially overlapped objects and an additional process for grouping segmented regions is required. Chi and Caldas (2011) proposed a method that detects and classifies construction entities including construction equipment and workers. Several characteristics of foreground blobs which result from background subtraction are trained for the classification. However, their work lacks information regarding how to e.g. differentiate a worker from a pedestrian, which is vital in construction sites located in residential areas. Moreover, their method is sensitive to illumination changes since it relies only on the foreground blob features.

2.4 Vision-Based 2D Tracking

Even though object detection methods localize interested types of objects in video frames, the results are lack of identification of the objects. There are no links between the results of different frames, thus trajectories are not available from using only object detection methods. The methods only provide 2D positions of objects for each frame. Therefore, in order to fill this gap, tracking algorithms are required. Once an object is detected in the current frame, 2D tracking algorithms search the most probable position of the object in the next frame based on its visual pattern and motion pattern. 2D vision trackers can be classified as contour-based, Template-based, and point-based trackers, according to the manner of representation of the objects (Yilmaz 2006). Figure 2.1 shows the examples of their representations. This chapter provides a review of each method.

Figure 2.1: The representations of the object (contour-based, Template-based, and point-based methods from left to right)

2.4.1 Contour-Based Methods

In contour-based methods, the object in the image is represented by contours or silhouettes which encompass the area of the object. These tracking methods use the contours to track objects by estimating and updating the region or boundary of the target in the current frame and comparing that with the results acquired from the previous frame (Nguyen 2002). These methods generally use edge features, which are easy to implement and stable to illumination changes. Also, they can maintain successful tracking regardless of any change inside the contours. However, contour-based methods commonly have problems with the images whose edges are not strong enough to extract edge features because the contours from weak edges usually fail to represent the actual boundaries. There are two types of contours which are used in contour-based tracking methods; parameterized and non-parameterized contours (Nguyen 2002). Methods that use parameterized contours approximate the contour using a parametric model (e.g. B-Splines) or detect the contour by minimization of the contour's energy function. The latter, which is referred to as "Snake" is the typical model of the parameterized contours (Yokoyama and Poggio 2005; Tsechpenakis et al. 2004). In this approach, an internal and

external energy are minimized along the contour and result in development of the shape of the contour (Tsechpenakis et al. 2004). On the contrary, the non-parameterized contour is defined merely as a boundary of a region e.g. a border between the object and the background and is obtained mainly by removing background edges (Nguyen 2002). The strength of such an approach is that it can be used to represent the contour of an arbitrary shape (Nguyen 2002), and as a result, it has been used in most contour-based tracking methods of recent years.

Algorithms employed in contour-based trackers either use “photometric variables” (e.g. intensity, color, texture), “geometric variables” (e.g. edges, lines, corners), or a combination of both (Freedman and Zhang 2004). It is argued that trackers that use photometric variables are more advantageous to geometric trackers because they are more reliable in the presence of illumination variations and cluttering, and they also take into account the rich amount of information existing in the images (Freedman and Zhang 2004).

Described contour-based methods are inappropriate for the purpose of 3D tracking of construction entities. First, the main advantage of contour-based trackers is that they can detect the exact contour, which is useful in applications where exact posture of the target is needed. However, the desirable 2D tracking results for the proposed stereo vision based 3D tracking are constant centroid points of the objects, and the flexibility of the contours actually degrades the accuracy of the calculated centroids. Second, most project related objects in construction sites are rigid objects that do not deform heavily, and even workers are commonly tracked relying on the unique colors of their hard hats and vests, to differentiate them from pedestrians. Consequently, the merits of contour-

based methods cannot contribute to a 3D vision tracking framework for construction. Third, compared to Template-based methods, contour-based methods were found weak to illumination variation, and were likely to lose the object which occupies a small area.

2.4.2 Template-Based Methods

Template refers to the basic shape and appearance of the object (Yilmaz et al. 2006). As a result, these trackers are also referred to as region-based or appearance-based methods. Region-based methods track connected regions which appropriately represent the shapes of the objects. Based on the regions' information such as color and texture, the methods compute and update the motion of the template in each frame (Marfil et al. 2007; Schreiber 2008). In region-based methods, color histograms generally play an important role since color is typically a good distinguishing feature and helps to manage partial occlusion (Marfil et al. 2007). However, relying only on the color may cause the tracker to lose objects of similar colors when they are occluded by one another.

Two sub-categories of template-based trackers are template and density-based appearance models, and multi-view appearance models. Template-based models, which take into account both color histograms and spatial information, are widely-used target models. The basic concept of template tracking is finding the region that best matches with the template that is manually determined in the first frame. One approach to do this is the mean-shift procedure, which is an analysis technique that works based on the comparison of the object histograms with the window approximated around the location of that object (Yilmaz et al. 2006). Templates and density-based trackers are advantageous because of their simplicity and low computational cost. However, they are not suitable if the objects change considerably in each frame (Yilmaz et al. 2006).

Schreiber (2008) presented a region-based method that generalizes the template matching Lucas-Kanade algorithm (Lucas and Kanade 1981). This method combines template matching and histogram-based tracking methods. It divides the region of interest into several overlapping patches, each of which is represented by a histogram, so that spatial information can be kept. Within this framework, partial occlusion and relatively slight changes in the object's appearance are handled well. However, it has the limited ability to cope with large transformations because it uses a fixed template obtained in the first frame.

Maggio et al. (2009) proposed a template-based algorithm that effectively combines the CONDENSATION algorithm (Isard and Blake 1998) with the extended Mean-Shift algorithm (Comaniciu et al. 2003). It inherits from the CONDENSATION algorithm the ability to cope with occlusions and improves the performance of the Mean-Shift algorithm through the additional estimation of the target rotation and anisotropic scaling. The target object is approximated with an ellipse in which every pixel has weighted normalized color histogram. Additionally, similar to Schreiber's method (2008), multiple semi-overlapping histograms are introduced in order to complement the lack of spatial information. This method showed robustness to occlusion and the changes in the objects' planar rotation or scale. Nevertheless, since this method also uses the fixed template model, it is not capable of managing severe changes in the target object's pose. Marfil et al. (2007) proposed a template-based framework that has the ability to track an object showing relatively severe pose variation by updating the template at every frame. This method uses the bounded irregular pyramid (BIP) which represents a target object as a hierarchical model (Marfil et al. 2007). At every frame, this hierarchical template model

is rearranged and updated reflecting the previous frame's information. This hierarchical template-based model can lessen computational cost because it uses the higher level that contains fewer elements instead of using all pixels in the matching process. The nodes keep the information of hue, saturation, and brightness to make tracking insensitive to the sharp changes in illumination (Marfil et al. 2007). This framework can track non-rigid objects and does not require a priori knowledge of the object to track. Since the template model is adjusted according to the updated objects' appearance, it can successfully maintain tracking when the objects exhibit severe transformation.

Ross et al. (2008) proposed a framework that uses multi-view appearance models. The principal components of this framework are the subspace representation, particle filter, and online update (Ross et al. 2008). The model of objects is composed of a low dimensional subspace representation, the eigenbases which are obtained from a fixed number of previous frames. This model provides abundant information of the object regions. The online learning process updates the model and enables it to be adapted to the changes in appearance caused by the deformation of object or illumination conditions, etc. A particle filter estimates the motions of objects without optimization processes.

2.4.3 Point-Based Methods

In point-based trackers, some limited points are selected as features of an object and only these points are tracked in consecutive frames (Yilmaz 2006). Point-based tracking can be interpreted as the correspondence of identified objects represented by points across frames (Yilmaz 2006). Point-based methods are best for tracking extremely small objects which can be represented by a single point (Yilmaz 2006). However, the larger an object appears in image frames, the more points may be demanded for successful identification

of the object. In addition, when multiple points are employed, grouping of points that belong to the same object becomes an important problem (Yilmaz 2006). Also, it is important to select and keep appropriate feature points that can effectively represent the objects. Point-based methods using multiple points are beneficial to characterization of non-rigid objects, provided that the locations of the feature points are flexible.

Cox and Hingorani (1996) presented a method for finding point correspondences that can be used in point-based tracking method. They employed Reid's MHT algorithm (Reid 1979) which enabled occlusion handling, improving it in terms of computational complexity. In order to reduce the computational cost, they narrowed the search space in prediction by introducing the k-best hypothesis (Cox and Hingorani 1996). Shafique and Shah (2003) also proposed a point correspondence method. Unlike the MHT algorithm that uses multiple heuristics, this method employs a single non-iterative greedy algorithm that allows real time tracking and occlusion handling (Shafique and Shah 2003). In their experiments, they showed the application to the tracking of small object such as a flock of birds in the sky or particles in a cylindrical reservoir. First, one feature point is assigned to each object by the KLT method (Shi and Tomasi 1994) and then the objects are successfully tracked by their point correspondence method.

Arnaud et al. (2005) proposed a point-based method that tracks a textured object with multiple feature points. They combined a Rao-Blackwellized particle filter (Khan et al. 2004) with a model composed of a planar cloud of feature points. The concatenation of all the points' locations is an important factor in estimating the object's state. This method allows for not only occlusions but also deformation of the object. However, this

method considers only planar movement and thus, may fail when the object rotates out of plane and changes pose severely.

Mathes and Piater (2006) proposed a point-based method in which a point distribution model is used to represent objects. This model employs feature vectors of the interest points to describe objects. The feature vector, which consists of the local appearance and the spatial configurations of feature points, enables the tracker to differentiate two similar-looking objects even when they are occluded by each other (Mathes and Piater 2006). The feature points are located sparsely to describe non-rigid objects accurately and to keep the stability in case of partial occlusion. Most importantly, the feature points are updated for every frame to make the model adaptive to the change in appearance and illumination. It matches current feature points to the image points. Based on the stability of the matching, stable points are added to and unstable points are removed from the model. When an object is occluded by another object, the update process stops to avoid the addition of bad points (Mathes and Piater 2006). Accordingly, it is stable under the changes in scale, appearance and shape. The stability is maintained in the image when occlusion occurs.

2.5 Stereo View Geometry

As explained in Chapter 1.4, two cameras are employed for tracking construction resources in 3D. Vision-based tracking is not comparable with other 3D technologies which are described in Chapter 2.1 unless it can provide 3D information. In order to reconstruct the 3D position of an entity, several steps must be taken to determine the stereo view geometry (Hartley and Zisserman 2004). Heikkilä and Silvén (1997), Zhang (1999), and Bouguet (2004) presented and provided standard calibration tools. The

calibration tools reveal intrinsic camera parameters including the focal length, the principal point, radial distortions and tangential distortions. They use calibration objects which have specific patterns such as a checkerboard. Figure 5.1 In Zhang's calibration method, tangential distortion is not modeled. Heikkilä and Silvén's toolbox and Bouguet's toolbox use the same distortion model which takes into account both radial and tangential distortions. Therefore, both toolboxes generally result in almost equivalent calibration. Bouguet provides additional functions such as error analysis which is useful to re-calibrate with revised inputs.

After having calibrated each camera separately, the external camera system has to be determined (see Figure 2.2). For this purpose feature points are identified and matched within the two camera views. The most well-known and robust algorithms commonly used for this task are the SIFT (Scale-Invariant Feature Transform) (Lowe 2004) and SURF (Speeded Up Robust Features) (Bay et al. 2008). While SIFT uses Laplacian of Gaussian (LoG), Difference of Gaussian (DoG), and histograms of local oriented

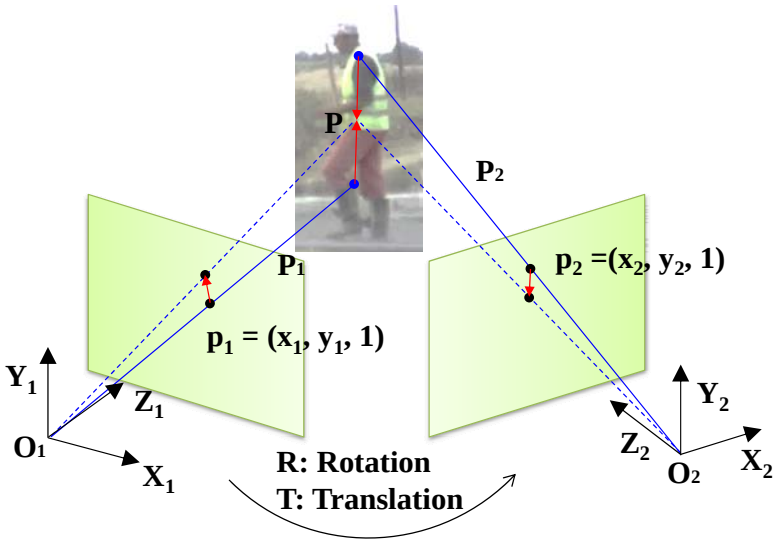


Figure 2.2: Epipolar geometry and centroid relocation

gradients, SURF relies on a Hessian matrix and the distribution of Haar-wavelet responses for feature point detection and matching, respectively. While SIFT turned out to be slightly better in terms of accuracy, SURF is computationally much more efficient (Bauer et al. 2007). SIFT and SURF provide point matches including extreme outliers (mismatches) that have to be removed. To achieve that, robust algorithms for managing the outliers were introduced. RANSAC (RANdom SAmple Consensus) (Hartley and Zisserman 2004), and MAPSAC (MAximum a Posteriori SAmple Consensus) (Torr 2002) are the representative robust methods. RANSAC minimizes the number of outliers by randomly selecting a small subset of the point matches and repeating the maximization process for different subsets until it reaches a desired confidence in the exclusion of outliers. One of its problems is poor estimates associated with a high threshold (Torr 2002). MAPSAC which works in a similar way resolved this problem by minimizing not only the number of outliers but also the error associated with the inliers.

The next step is the estimation of the essential matrix, E based on the identified point matches. In general, the normalized eight point (Hartley 1997), seven point (Hartley and Zisserman 2004), six point (Pizarro et al. 2003), and five point (Nistér 2004) algorithms are used. Eight, seven, six and five is the minimal number of points required to perform the estimation. Rashidi et al. (2011) compared the resulting accuracy of these algorithms in practical civil infrastructure environments, obtaining the five point algorithm to be the best. However, due to its simplicity and reasonable accuracy the normalized eight-point algorithm is still the most common one and the second best according to (Brückner et al. 2008). Based on the essential matrix, E the relative pose of two cameras (R, T in Figure 2.2) can be derived directly (Hartley and Zissermann 2004).

In the last step, triangulation is performed. Based on two corresponding pixels in the respective view, two lines of sight have to be intersected to find the 3D position (Figure 2.2). However, due to image noise and slightly incorrect point correspondences, the two rays may not intersect in space. To address this problem, Hartley-Sturm optimal triangulation (Hartley and Sturm 1997) and optimal correction (Kanatani et al. 2008) algorithms are currently used as standard methods for finding corrected correspondences. They both try to find the minimum displacement based on the geometric error minimization, correct the pixel coordinates accordingly and intersect the corrected rays to determine 3D coordinates. While the latter has a faster process, the former's results are more accurate (Fathi and Brilakis 2011).

Several researchers have introduced and applied stereo vision technologies to construction. Most applications presented so far are related to 3D modeling of structures for progress monitoring. Chae and Kano (2007) estimated spatial data for development of a project control system from stereo images. In another work, Son and Kim (2010) used a stereo vision system to acquire 3D data and to recognize 3D structural components. Golparvar-Fard et al. (2010) presented a sparse 3D representation of a site scene using daily progress photographs for use as an as-built model. While these previous works employed stereo vision to create 3D geometry models based on static feature points, this research applies stereo vision to locate moving entities in 3D across time.

2.6 Summary

Obviously, various tasks in construction management can benefit from tracking technologies which is capable of providing location of construction entities across time. GPS, RFID, and UWB are well-known technologies that are applied to track on-site

construction entities. While GPS has been used mostly for heavy equipment, RFID has been considered appropriate for construction materials such as pipes and concrete elements. UWB is also used for tracking materials, but mainly in indoor environment.

In spite of the successful performance of the radio frequency technologies, vision-based tracking draws interests because it is free of tag or sensors. Vision-based tracking tracks objects in a video based on their visual patterns and motion patterns. Vision-based tracking can track multiple objects with only one sensor, which is a camera, as long as the objects are present in the camera view. There are various types of tracking methods, and they can be classified into three categories – contour-based methods, template-based methods, and point-based methods – based on the way of representation of objects.

A few research works were performed to use vision-based tracking for productivity measurement and safety enhancement. However, the results of the employed tracking methods were limited to 2D information. 2D results are generally not enough to extract substantial information for most construction management tasks since it is unknown how far entities are located from the camera. Due to the lack of depth information, even approximate distance measurements between two entities are not reliable, but necessary for safety management. Therefore, additional procedures to obtain the depth information and calculate 3D spatial coordinates are required. Furthermore, entities to be tracked had to be manually initialized in the first frame in the previous works. In order to further automate the tracking process, methods to automatically recognize and localize construction entities are necessary.

CHAPTER 3

DETECTION OF CONSTRUCTION WORKER AND EQUIPMENT

In order to initiate a vision tracker, it is necessary to first determine which entities to track in each camera view. All entities that enter the camera views have to be captured and marked to trigger a vision tracker. Moreover, a vision tracker generally fails to track an entity when it is fully occluded. Therefore, when the entity gets free from the occlusion, it has to be initialized again. Given a pixel region corresponding to each entity, a vision tracker extracts the visual patterns from the region and starts tracking on the basis of the patterns. However, it is time consuming and error prone to manually mark all construction entities in multiple views, due to the large amount of entities to track in construction sites. Therefore, an automated way of detecting the entities is required.

3.1 Construction Worker Detection

This chapter deals with detection of construction workers for initializing 2D vision tracking. Specifically this chapter aims to 1) investigate image features that effectively characterize the appearance of construction workers who wear safety vests, 2) determine appropriate measures to evaluate the method regarding that the main role of the method is to accurately initialize 2D vision trackign, and 3) test the method for detecting construction workers in various illumination conditions and backgrounds.

3.1.1 Methodology

In this research, ‘construction worker’ is considered as a person wearing safety gear such as safety vests and hard hats. The proposed method takes advantage of three types of

image features to describe construction workers: motion, shape and color. These features are separately implemented in sequential steps. First, it detects foreground blobs where objects in motion are expected to exist. Given the fixed camera views, the difference of pixel values between a background model and the incoming frames is the main cue to recognize motions. Second, it identifies the regions corresponding to human bodies from the foreground blobs based on their patterns of HOG (Histogram of Oriented Gradients) features. Third, the detection regions that result from the second step are classified into construction workers and non-workers. It uses color histograms and a k-NN (k-Nearest Neighbors) (Cover and Hart 1967) classifier to characterize fluorescent colors of safety gear. Figure 3.1 illustrates the described framework. In short, the first and second steps detect people including workers based on their motion and shape features, then the third step sort out workers from the detected people by analyzing their color features.

Background subtraction, in the proposed method, reduces the candidate regions for worker detection. It is used to achieve higher computational efficiency and lower false alarm rates. The second step, which is the detection of people with HOG, is computationally far more expensive than the background subtraction. Background

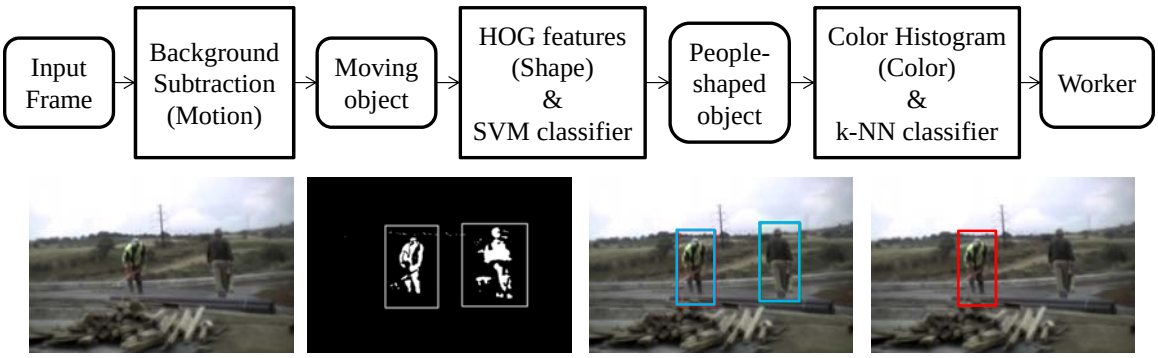


Figure 3.1: The framework of construction worker detection

subtraction restricts the search areas of the second step. It allows the step to scan only foreground blobs and prevents from unnecessarily scanning irrelevant areas. In addition, it prevents the detection of non-people objects in the background scene (i.e. a static area which has a similar shape to people) in the following step. It eventually leads to higher precision. Once foreground blobs are obtained, they are post-processed to remove small blobs and to merge adjacent blobs. For each foreground blob, the smallest rectangle that encloses the blob is found. Finally, each rectangle is inflated to allow for some margin around a moving object, which is necessary to calculate HOG shape features in the next step. Various methods of background modeling have been presented: approximate median filter (Macfarlane and Schofield 1995), MoG (Stauffer and Grimson 2000), color co-occurrence (Li et al. 2002), etc. In this research, the median filter method is employed since it is computationally the most efficient. Figure 3.2 exhibits the foreground blobs extracted with the three methods and the rectangular foreground regions after post processing. Even with the simplest process, the median filter method provides comparable results to the other methods.

Wearing hard hats or safety vests does not induce significant changes in human shapes. Figure 3.3 shows two gradient images obtained by averaging a number of gradient images of ordinary people and construction workers. Human can be intuitively inferred from both gradient images. Therefore, as a step to narrow down candidate regions to people, the second step employs HOG features which have proven to be robust in detecting human bodies. Detection of people in test images works by sliding a search window across the foreground blob regions. The HOG feature is calculated from each window and classified into either people or non-people. In order to embed various

Figure 3.2: The foreground regions which result from the approximate median filter (1st row), the MoG (2nd row), and the color co-occurrence (3rd row) methods

appearances of people into the HOG feature template, training of a classifier is necessary. The SVM is chosen for the training as in Dalal and Triggs' work [34]. On the basis of HOG features extracted from a large number of positive (images of people) and negative (images without any people) images, an SVM classifier learns to recognize the patterns of HOG features and to distinguish people from others. Training images have to be in the same size so that all their HOG features have the same vector size.

The results of the second step are the rectangular regions determined to include a person inside. These rectangles are further processed by another classifier which determines whether they are workers or not. In other words, this step filters out the non-workers out of the people regions. The HSV color histogram is selected for this purpose. ANSI (American National Standards Institute) stipulates that construction workers must

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

(e) (f) (g) (h)

Figure 3.3: (a)-(c): people images, (d): average gradients of 200 people images, (e)-(g): worker images, (h): average gradients of 200 worker images

use high-visibility safety apparel since they are regularly exposed to the hazards of low visibility (ANSI/ISEA 2010). It specifies three colors for the high-visibility apparel: fluorescent yellow-green, fluorescent orange-red, and fluorescent red. Since the colors are limited and all fluorescent colors, it is viable to characterize the patterns of the safety gear colors.

The RGB color space, which is an additive color model, is not effective for modeling the safety gear colors because of its sensitivity to illumination conditions. This fact is illustrated in Figures 3.4 and 3.5 which show the RGB color histograms (32 bins) of yellow-green and orange-red safety vests, respectively. In each figure, the two safety vests are the same type, but with different illumination conditions – one in a bright condition (1st row), and the other in a dark condition (2nd row). It is observed that

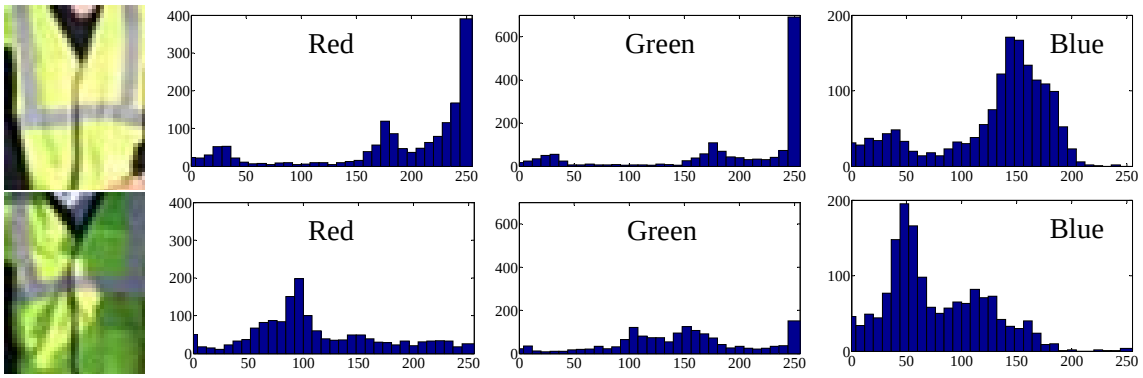


Figure 3.4: RGB color histograms of yellow-green safety vests in bright (1st row) and dark (2nd row) illumination conditions

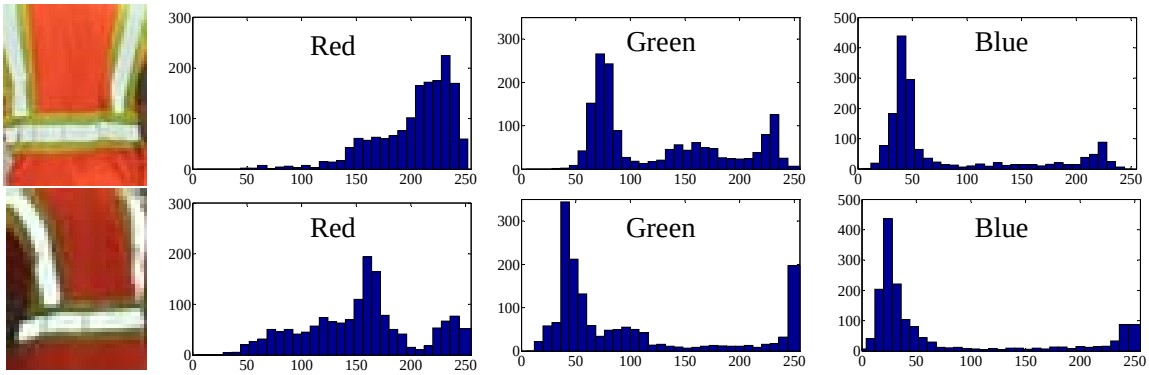


Figure 3.5: RGB color histograms of orange-red safety vests in bright (1st row) and dark (2nd row) illumination conditions

significant disparity between the color histograms in the 1st and 2nd row is caused by the slight change in illumination. Compared to the RGB color space, the HSV color space is in better accordance with the conceptualization of human eyes, thus, is more appropriate in categorizing objects by colors. The three components of hue, saturation, and value measure the actual color, purity, and intensity (brightness), respectively. Figures 3.6 and 3.7 show the HSV color histograms (32 bins) of the safety vests as in Figures 3.4 and 3.5. Hue and saturation histograms in the 1st and 2nd rows exhibit similar patterns in spite of the illumination change which is reflected in the value histogram. In addition, hue and saturation histograms in Figure 3.6 and 3.7 are distinct from those of ordinary vests on

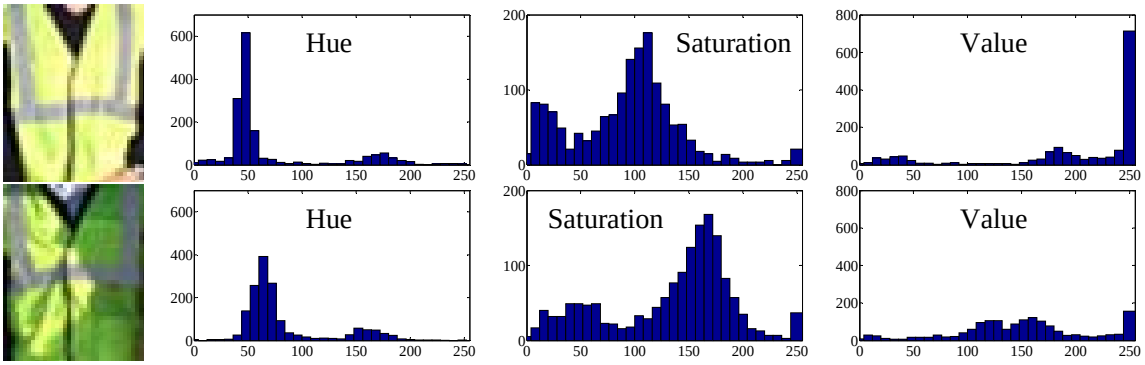


Figure 3.6: HSV color histograms of yellow-green safety vests in bright (1st row) and dark (2nd row) illumination conditions

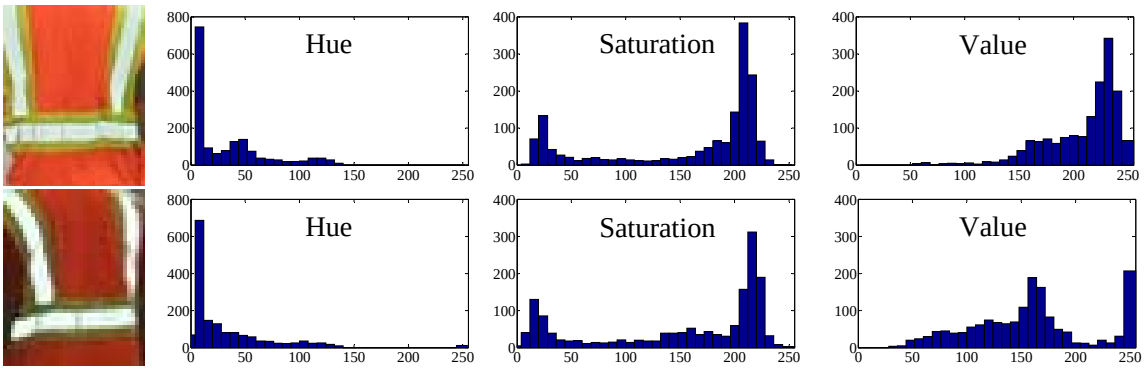


Figure 3.7: HSV color histograms of orange-red safety vests in bright (1st row) and dark (2nd row) illumination conditions

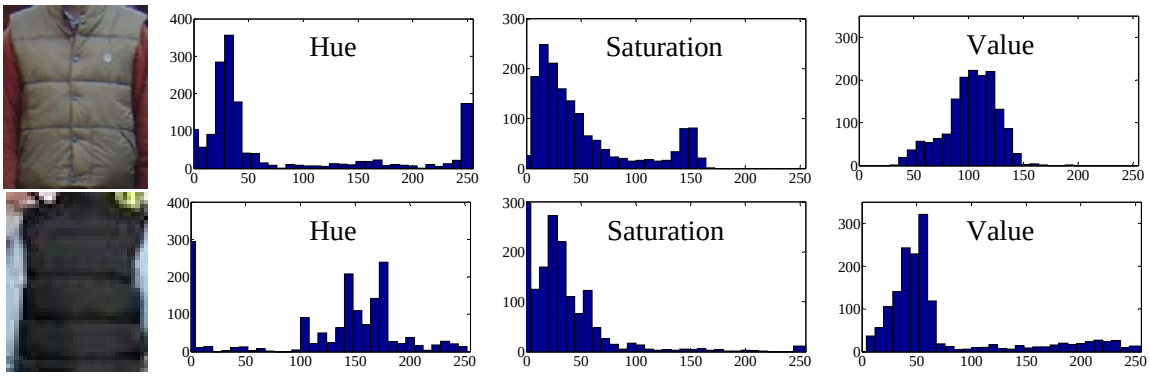


Figure 3.8: HSV color histograms of ordinary vests on pedestrians

pedestrians in Figure 3.8. Figure 3.9 shows saturation images of a pedestrian and construction workers. Unlike in the pedestrian image, the safety vest regions in construction worker images are highlighted by the saturation components regardless of their colors. These illustrations signify that the distinct characteristics of safety vests can be effectively modeled with hue and saturation histograms. Accordingly, the third step exploits hue and saturation histograms for characterizing the visual patterns of the safety gear.

As in the second step, this step also needs to go through a training process. Positive and negative data, in this step, are the images of construction workers and ordinary people, respectively. Focusing exclusively on safety vests and hard hats, this step takes into account only the upper half of the human body to construct color histograms. The obtained color histogram is normalized to minimize the effect of the image window sizes. Negative images contain a few images of people wearing jackets of which the colors are similar to safety vests. These images prevent SVM from constructing clear boundaries between the color histogram vectors of worker and non-worker. The k-NN is more suitable to this case because it performs classification simply

Figure 3.9: Saturation images of a pedestrian (a) and construction workers (b and c)

based on a majority vote of its k neighbor. Even though the main role of this third step is classifying a person into a worker or a non-worker, it also acts as a filter that eliminates false positive results of the second step (non-people regions).

3.1.2 Implementation

The whole framework as a single package is implemented using Microsoft Visual C# in .NET Framework 4.0 environment. The size of the HOG template used for people detection is 64×128. In order to detect people smaller than the template size, the foreground regions are scaled up. The HOG feature is calculated on the basis of two spatial layers (cells and blocks). The first layer is comprised of a grid of cells (8×8 pixels) where a local HOG is obtained. The second layer is comprised of a grid of blocks. A block is a collection of 2×2 cells. Four local HOG's of 2×2 cells in a block are aggregated and normalized together. The normalization allows for invariance to illumination changes. The histogram is constructed based on the angle and magnitude of gradients. It consists of 9 bins which represent evenly distributed angles ranging from 0 to 180 degrees. A gradient magnitude at each pixel is added to the corresponding bin which the angle falls into. A sequence of HOG features that are spatially distributed in a grid manner structures a final descriptor of the input image.

The k-NN classifiers associated with color histograms are trained with images of upper half bodies wearing both hard hats and safety vests. A majority vote of 'k' nearest neighbors determines the classification. Parameter studies are performed to determine the optimal number of histogram bins as well as the optimal number of nearest neighbors (k). In all tests, 500 positive images (upper half body with safety gear), including yellow-green, orange-red, and red safety vests, are collected for the training of color histograms.

3.1.3 Experiments and Results

To validate the implemented method, the method is tested on ten high-definition (HD) videos taken with a HD camcorder (Canon VISXIA HF S100). However, the videos are resized to 768×432 for the sake of faster process. 5 frames per second are processed in near real time. The videos are taken of 4-5 people wearing different combinations of safety gear (hard hats and safety vests) to investigate the effect of the combinations on the detection results. Table 3.1 illustrates the combinations involved in three of the videos. All the videos have different backgrounds and various illumination conditions.

Table 3.1: The combinations of safety gears

Person	Video 1		Video 2, 3	
	Safety vest	Hard hat	Safety vest	Hard hat
1	X	X	X	White
2	X	White	X	White
3	N/A		Orange-red	X
4	Yellow-green	White	Yellow-green	White
5	Orange-red	Yellow	Orange-red	Yellow

X: not wearing

3.1.3.1 Metrics for Performance Evaluation

Two metrics are chosen to measure the performance of the method. The metrics are selected from the perspective that the main purpose of the method is initializing a vision tracker. The first metric is precision which is popular in evaluating pattern recognition. To clarify the meaning of precision, several terms are defined as follows:

- TP (true positive): the number of correctly detected workers
- FP (false positive): the number of the detections which is not relevant to a worker
- FN (false negative): the number of missed workers (i.e. not detected)

Precision is defined as a ratio of TP to TP+FP and measures the reliability of the detection. The second metric is the time delay of detection. It measures how instantly it detects a worker after the worker enters the scene or gets out of an occlusion case. It is worthwhile to note that the recall is not an appropriate measure of a method for vision tracker initialization. The recall is the ratio of TP to TP+FN. The proposed method is not designed to detect a worker in every frame, for which a tracker is responsible. For instance, a method that detects a worker on every other frame resulting in just 50% recall (low recall) could perform well enough to initialize a tracker as long as it detects no other type than a worker (high precision).

3.1.3.2 The Definition of 'Construction Worker'

Although recall is not an appropriate metric overall, it is calculated for the analysis of the effect of safety gear combinations. Since the method is devised to detect workers that appear in the view, occluded workers are not the target of the method, hence are not counted as positive objects. Table 3.2 summarizes the recall of five people wearing different combinations of safety gear as shown in Table 3.1. There is clear distinction between recall values of Persons 1-2 and Persons 3-5. Person 1 and 2 who did not wear safety vests are barely detected. Person 3 who only wore a safety vest is detected far more frequently than Persons 1-2, and its detection rate is comparable to Persons 4-5's. It can be inferred from this fact that the safety vest has a more decisive effect on color histograms than the hard hat. As ANSI (2010) stipulates, safety vests have a more limited

number of colors, which contributes to the effective detection. According to these observations, a ‘construction worker’ is defined in this research as a person wearing at least a safety vest. Hence, Persons 3, 4 and 5 are defined as construction worker in the following experiments. Since the proposed method first detects people (the second step) and then sort out construction workers (the third step), it can eventually detect both workers and non-workers and classify them by comparing the results of the second step and the third step of the framework in Figure 3.1. Figure 3.10 exhibits two result frames from each video. Whereas the left images contain only correct detections, the right ones contain false results as well. Persons 4 and 3 are classified as non-worker in Video 1 and 3, respectively (FN). Person 2 is classified as construction worker in Videos 2 (FP). The third step which plays a core role in classifying construction worker and non-worker takes only 14 ms in average for each frame which is about 9 % of the total processing time. This classification will be useful to warn people who are not wearing safety vests on a construction site for safety issues.

Table 3.2: The detection rate (recall) of 5 people

Recall (%)	Video 1	Video 2	Video 3
Person 1	0.0	0.0	0.0
Person 2	0.0	1.8	6.3
Person 3	-	77.6	71.1
Person 4	88.7	96.9	80.9
Person 5	88.3	74.6	76.8

Figure 3.10: Examples for correct detections (left) and false results (right) (Videos 1-3 from top to bottom)

3.1.3.3 Results - Precision and Time Delay

Precision is calculated according to the definition of ‘construction worker’, and time delay is also measured for all ten videos. It should be noted that several videos include both shade and light where abrupt changes in illumination occur as workers cross the border between them (e.g. Videos 2 and 3 in Figure 3.10). Figures 11-13 show the results of parameter studies on the number of bins in saturation and hue histograms, and the number of nearest neighbor (k). The number of bins in saturation histograms is

determined to be ‘8’ because of its highest precision and the delay less than 1 s (Figure 3.11). ‘64’ is selected for hue histograms since it exhibits the minimal delay and comparable precision with other sizes (Figure 3.12). Accordingly, the total size of the color histogram is 72 (=64+8). Regarding the value of ‘k’ in the k-NN learning process, ‘11’ is found the best (the least delay in Figure 3.13).

Using the selected parameters, four tests are conducted for each video through the training of color histograms with 550, 1100, 1650, and 2200 negative images. The results are summarized in Tables 3.3 and 3.4. Results of step 1 (background subtraction) and step 2 (people detection based on HOG) remain the same in all four tests. In Table 3.3, precision increases as more negative images are used. It indicates that the use of more negative images helps reduce FP’s. However, on the contrary, recall works inversely to the number of negative images, which means TP’s are also reduced by using more

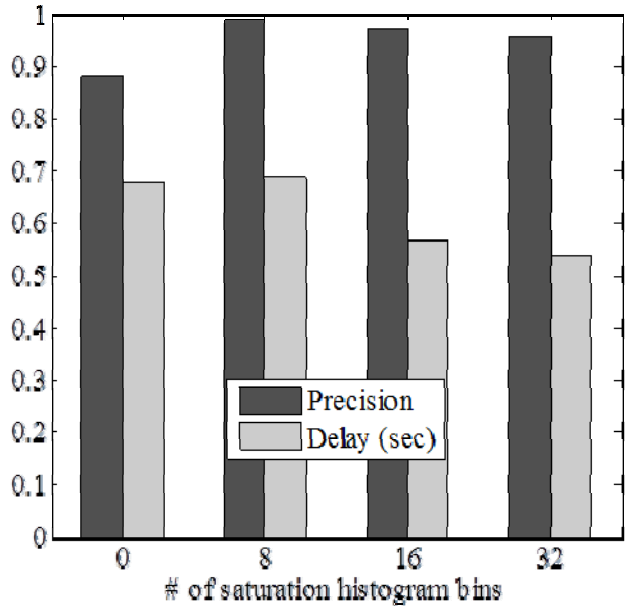


Figure 3.11: The performance variation of the method depending on the number of bins in a saturation histogram

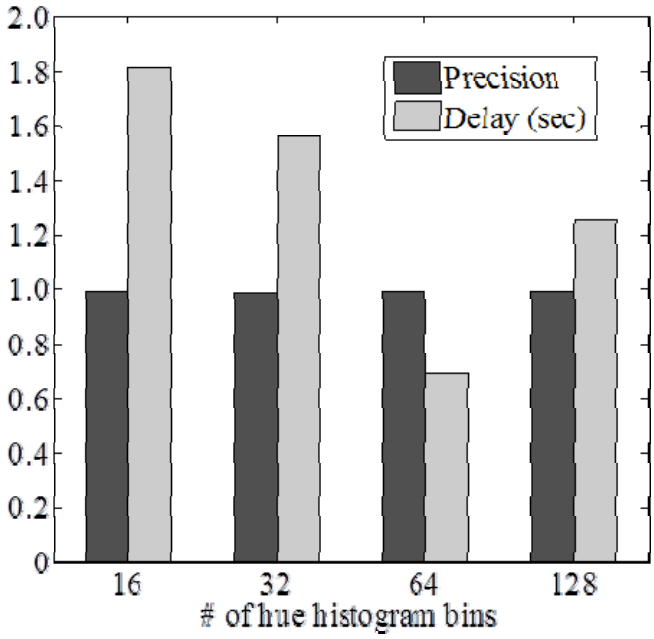


Figure 3.12: The performance variation of the method depending on the number of bins in a hue histogram

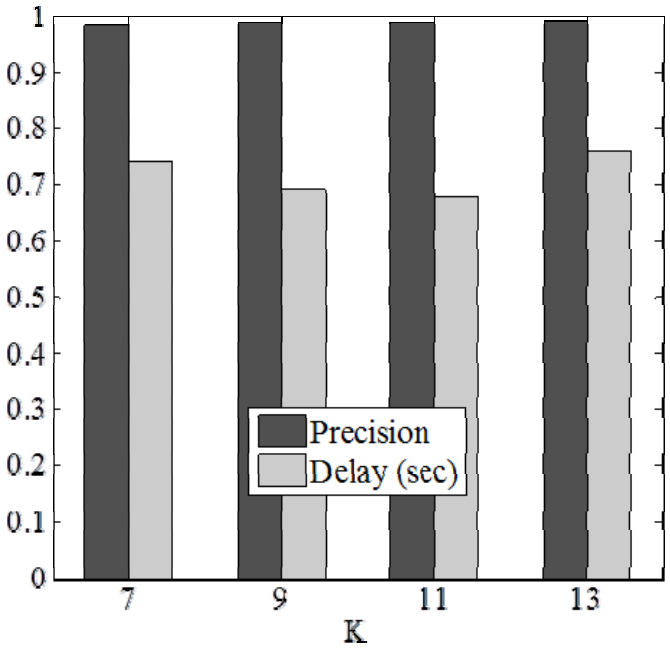


Figure 3.13: The performance variation of the method depending on the ‘k’ value

negative images. Figure 3.14 shows examples of these effects. The first and second rows illustrate the removal of FP's (non-person and Person 2) while the third row shows a missed TP (Person 5). Regarding precision results, the proposed method achieved the best performance when 2200 negative images are used in training the color histograms of upper half body. As explained in the previous chapter, time delays are considered to better reflect the performance of the initialization method. In total, 73 cases are observed in which a worker enters a view or gets free from occlusion. In Table 3.4, the use of 2200 negative images results in a maximum of 0.67 s delay which is only 0.11 s longer than using 550 negative images. The maximum is calculated with a 95% confidence interval for the standard normal distribution. The 99.0 % precision and the 0.67 s delay signify the potential of the proposed method for initializing a vision tracker.

Figure 3.14: Detection of construction workers using 1100 (left) and 2200 (right) negative images

Table 3.3: The precision of detection results

# of negative training images	550	1100	1650	2200
Precision (%)	90.1	97.2	97.7	99.0
Recall (%)	87.1	83.6	82.4	81.4

Table 3.4: The delay of detection

# of negative training images		Frames	Seconds
550	Average	1.5	0.30
	Standard Deviation	0.8	0.16
	Maximum	2.8	**0.56**
1100	Average	1.6	0.31
	Standard Deviation	1.0	0.20
	Maximum	3.2	**0.64**
1650	Average	1.6	0.33
	Standard Deviation	1.1	0.21
	Maximum	3.4	**0.68**
2200	Average	1.6	0.32
	Standard Deviation	1.1	0.21
	Maximum	3.4	**0.67**

Additionally, since the proposed method is composed of three sequential steps, the performance of each step is further investigated. For this purpose, the critical cause of the missed workers (FN) is analyzed (Table 3.5). The FN cases are classified into the three steps which cause them. Overall, the third step is discovered as a main step that

filters out FN's. In detail, it is also observed that Persons 3 and 5, who wear orange-red vests, are mainly discarded by the third step while Person 4, who wears a yellow-green vest and a white hard hat, is discarded in the second step. The former is attributed to the negative images of people with orange-red shirts. In fact, negative images of upper half body contain a considerable number of images with orange or red shirts which could be confused with orange-red safety vests. In addition, white is a dominant color of hard hats in the positive images which force to eliminate Person 3 (brown hair) and Person 5 (yellow hard hat). On the contrary, it is inferred that the latter results from the similar background color to Person 4. Similar colors of background make it difficult to extract a human body shape.

Table 3.5: The cause of missed detections

Step	Background Subtraction	HOG shape detection	Color histogram classification
Person 3	9.2 %	26.5 %	64.3 %
Person 4	0.9 %	72.7 %	26.4 %
Person 5	17.2 %	13.2 %	69.5 %
Total	10.5 %	33.8 %	55.8 %

3.2 Construction Equipment Detection

The method of construction equipment detection is similar to construction worker detection. It is also comprised of three sequential steps (Figure 3.15). The three steps exploit motion, shape, and color features, respectively. The first step is exactly same as in

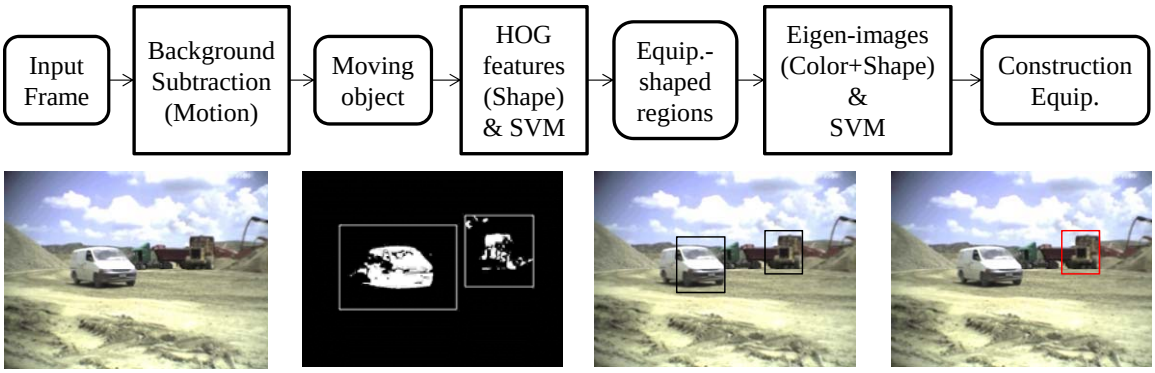


Figure 3.15: The framework of construction equipment detection

construction worker detection. It uses background subtraction to extract regions of moving objects. Cameras are fixed on construction sites so that they have static background scenes. The objects in motion (foreground blobs) are detected by comparing the current frame with the background scene. This step narrows candidate regions down to moving object regions.

The second step is to find the shape of construction equipment out of the candidate regions. As in construction worker detection, HOG features are used, and trained with SVM. However, different from construction worker detection, construction equipment detection requires several independent trainings for distinct views. For example, rear and side views of a wheel loader (Figure 3.16) should be trained independently because of the great difference in their appearances. In this research, four views (front, rear, left, and right) are trained independently. Therefore four separate models are created as equipment templates. Though training of more views such as rear-right and rear-left can lead to higher recall, it is not considered in this research since it will cost more processing time. Also, the gaps between the four trained views can be

(a) (b)

Figure 3.16: (a) Rear (b) and left views of a wheel loader: 4 principal components of eigen-images (right upper), and the reconstructed image (right lower)

effectively covered by the tracking process, which will be dealt with in Chapter 4. All four models are jointly used for searching every piece of construction equipment.

Potential chances of false detections are still remained in the motion- and shape-based detection. For example, a moving object that exhibits a similar shape of construction equipment can be detected through the first and second steps. The third step filters out these false detections based on eigen-images which contain both shape and color information. When exploiting color features, it is important to select color components that can best characterize colors of construction equipment. Even though construction equipment is generally yellow, there are other colors of equipment such as red and green. Also, RGB (red, green, and blue) values and gray-scale intensity vary depending on illumination conditions. Therefore, instead of RGB or gray-scale intensity, the proposed method employs the HSV (hue, saturation, and value) color space. Among the three components, it uses saturation which measures the purity of the color since most equipment has relatively pure colors. PCA reduces the dimension of eigen-images by removing the components corresponding to smaller eigenvalues. Similar to the second step, eigen-images are trained separately for each view of equipment. Training is

achieved through SVM. Figure 3.16 illustrates eigen-images for rear and left views of wheel loader. Figure 3.16 also shows the images reconstructed with 30 principal components of eigen-images. It should be noted that only lower half part (red) is used for side views because backgrounds (yellow) accounts for a substantial portion of the upper half (Figure 3.16).

3.3 Summary

The major hindrance to the use of general vision tracking is the lack of detection methods that are needed to automatically initiate the tracking. Even though several methods of detecting construction entities were proposed, no approach has been made to differentiate construction workers from pedestrians. Also, a robust method against illumination changes is missing. This chapter presented methods for automating the detection of construction workers, equipment, and vehicles in the video frames. The methods to detect construction workers or equipment exploit three types of features – motion, shape, and color. The background subtraction and HOG shape features trained with SVM are commonly used in both methods as motion and shape features, respectively. While the method for workers uses color histograms as a color feature, the one for equipment uses eigen-images which contain a certain extent of color and shape information.

Out of the three methods, the worker detection method has been tested on various illumination conditions and backgrounds. The proposed method is preliminarily tested for detecting five people wearing different combinations of safety gear. Based on the preliminary tests, ‘construction worker’ is defined as a person wearing a safety vest. According to the definition of ‘construction worker’, the experiments resulted in 99.0 %

precision and 0.67 second time lapse, which signifies that the proposed method can effectively initialize the tracking of construction workers.

CHAPTER 4

2D TRACKING OF CONSTRUCTION ENTITIES

Once interested objects are detected, the detected regions are fed to a 2D tracking algorithm so that they are tracked in the subsequent frames. As discussed in Chapter 3, detection methods provide independent results for each frame, relying on common features of an object type. On the contrary, tracking methods determines the location of each object in the current frame, making use of motion and visual patterns of the object that recognized in previous frames. This chapter deals with a comparative study of the existing 2D tracking methods to find the most appropriate one for tracking construction entities, and proposes a way of integrating detection and tracking methods to achieve more stable and reliable 2D localization processes.

4.1 Comparison of 2D Vision Tracking Methods for Tracking of Construction Entities

A comparative study of 2D vision tracking methods are performed for the purpose of identifying, in a scientific manner, the most effective tracker for construction resource tracking. To accomplish this objective, state-of-the-art trackers are classified into contour-based, template-based and point-based methods, as discussed in Chapter 2.4 and the performance of the three categories are compared. The domain specific challenges that might affect the tracking performance in construction sites are identified and used as test parameters for the comparison.

4.1.1 Independent Variables

Construction sites have some unique characteristics such as large-scale, outdoor and congested environments, where illumination conditions change frequently and the entities are likely to overlap with each other in the camera view. These characteristics directly affect the quality and outcome of vision tracking because they change the appearance of objects in the view of the cameras. Therefore, to find the most suitable 2D tracking method, one needs to know how well each of these factors can be handled. To satisfy this, a list of independent variables that are based on these characteristics is proposed here to evaluate and compare the 2D vision tracking methods. The variables should represent the general qualities common to the methods in the same category for the sake of the reasonable comparison of categories. These variables are based on the ability of the tracker to handle: 1) absolute value of illumination, 2) illumination variations, 3) occlusions, 4) scale variations, and also 5) various types of objects. One additional factor is the required cost. In terms of the equipment cost, there is no difference among the categories. The cameras, computers, and cables are the hardware required for every vision tracking method. The processing time is the factor where the difference exists among methods. However, it is not considered in the comparision because even the methods of the same category differ in processing time. Also, the processing time is dependent on the extent of the code optimization which is hard to explicitly quantify. Furthermore, because of the continuous advances of computer hardware capabilities, the difference of processing time may become slighter.

The absolute value of illumination is related to lighting conditions. When it gets extremely dark or bright, the information of objects' appearances that we can get from

videos is likely to be reduced. Accordingly, tracking might become more difficult. Working hours at construction sites can start from very early hours in the morning when it is still dark and go until very late hours of the day. Also, construction site locations vary geographically from very illuminated sites in deserts to light reflecting sites near oceans.

Also, illumination variations are important factor to consider. Illumination variations are related to changes that occur in lighting conditions. Illumination variations are common at open, large-scale and congested construction sites, where shadows and light reflections (from other equipment or adjacent buildings) are present. Such variations can affect tracking results because they can significantly alter the appearance of objects in the view of the cameras.

Occlusion refers to the state when an object is partially or fully blocked by itself or another object. In construction sites, there is constant movement of equipment, materials and personnel. As a result, it is extremely important to select a method that can continue to track objects even in the presence of occlusions. Self-occlusion is a common problem of contour-based methods especially when tracking non-rigid objects. It should be noted that one way to handle and control occlusions is by picking the right position to install cameras. By installing the cameras in higher levels, it is possible to minimize occlusion of objects. Also, the more cameras are employed in the sites, the easier it is to handle occlusion.

Scale variation indicates changes in the objects' size in the video images. The tracking model should be able to represent the objects regardless of their size. Since construction sites generally occupy an extensive area, project related objects are likely to

appear in different sizes as they move away from, or towards, the camera. The objects could occupy a large region of the image or only cover small number of pixels, depending on the distance from the camera. Therefore, it is important to select a method that can track small objects and also be able to handle changes in scale.

There are various kinds of objects in construction sites, which have different characteristics from each other. They have different appearances (colors and shapes) and different ways of movement (speeds and directions). In addition, the extent of deformation is another factor in relation to this metric. Rigid objects (e.g. wheel loaders, backhoes, steel beams) do not make severe changes in their poses. Therefore, their appearances are adequately predictable. The equipment with articulated movable parts can also be reliably tracked based on the rigid part (e.g. tracking a backhoe's body part). On the other hand, non-rigid objects (e.g. workers) change their poses dynamically. This makes their appearances in the subsequent frames unpredictable.

4.1.2 Dependent Variables

To evaluate the performance of tracking methods, the number of frames that are successfully tracked and the accuracy of centroid coordinates are used. The accuracy is measured by the errors determined as the Euclidean distance between the centroid of the tracked region and the ground-truth centroid (Figure 4.1). Accordingly, the unit of the error is a distance in pixels. The ground-truth is determined as the centroid of the rectangle that encloses the entity in each video frame and is found manually. The tracked region for the point-based method is determined as the smallest rectangle that encloses all points of the active shape model. The errors are obtained only with the successfully tracked frames and the average of the errors is finally calculated. This measurement is

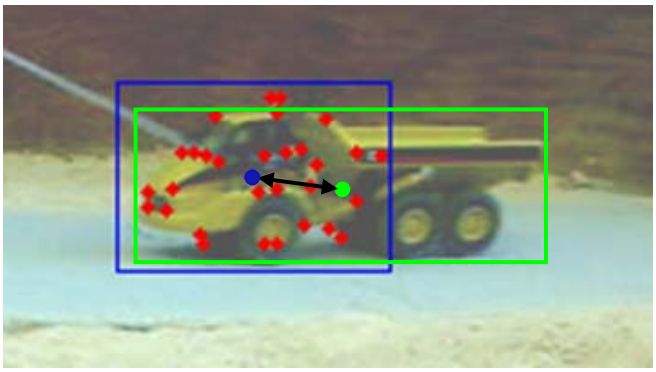


Figure 4.1: The error for accuracy measurement (The blue/green rectangle and dot represent the tracked/actual (ground-truth) region and centroid respectively. The arrow represents the error.)

valid for the evaluation since the purpose of this comparison is to find the appropriate methods for 3D vision tracking which requires corresponding centroid points from two different camera views.

4.1.3 Discussion on Contour-Based Methods

The main comparison experiments in Chapter 4.1.4 exclude the contour-based methods for the following three reasons. First, the main advantage of contour-based trackers is that they can detect the exact contour, which is useful in applications where exact posture of the target is needed. However, as mentioned in the previous section, the desirable results we want from 2D tracking are constant centroid points of the objects, and the flexibility of the contours actually degrades the accuracy of the calculated centroids. Second, most project related objects in construction sites are rigid objects that do not deform heavily, and even workers are commonly tracked relying on the unique colors of their hard hats and vests, to differentiate them from pedestrians. Consequently, the merits of contour-based methods cannot contribute to a 3D vision tracking framework for construction.

Above all, preliminary tests show that template-based methods perform better than contour-based methods (Makhmalbaf et al. 2010). For comparison purposes, one template-based method is selected that uses the Bhattcharyya coefficient as similarity measure and the mean-shift algorithm to carry out the optimization based on a method presented by Comaniciu et al. (2003). Also, a variational framework is selected as a contour-based tracker, which combines a Bayesian model to develop their variational method (Rousan and Deriche 2002) and a knowledge-based segmentation algorithm (Haker et al. 2001). These methods are implemented and tests are performed using real-site videos.

Some frames from preliminary tracking test results are shown in Figures 4.2, 4.3, and 4.4. In Figure 4.2, the worker (non-rigid object) is being tracked using the contour-

Figure 4.2: A worker tracked by a contour-based (upper row) method and a template-based method (lower row)

based method. This worker is tracked for 10 frames until he is occluded by another worker and then the tracking failed. On the other hand, the result from the template-based method shows that this method can track the worker in the equivalent frame and even the subsequent frames. In Figure 4.3, a bulldozer that is moving away from the camera is being tracked. The results show that the contour-based method loses the target in frame 2500 while the template-based tracker keeps tracking for another 6000 frames (until the object comes to the camera). Figure 4.4 shows the results of tracking of a backhoe's bucket. Complete deformation of the bucket in the view of the camera can be observed in these images. The results reveal that the template-based method can track the bucket

Figure 4.3: A roller tracked by a contour-based (upper row) method and a template-based method (lower row)

Figure 4.4: A backhoe bucket tracked by a contour-based (upper row) method and a template-based method (lower row)

for 723 frames while the contour-based lose the object in frame 101. The number of frames successfully tracked is taken into account as a performance measurement in this preliminary experiment, and the results are shown in Table 4.1. Based on the results, it is concluded that compared to template-based methods, contour-based methods are found weak to illumination variation, and are likely to lose the object which occupies a small area. Therefore, the contour-based methods are determined inappropriate for the purpose of 3D tracking of construction entities, and excluded from the experiments in the following chapter.

Table 4.1: Number of frames successfully tracked

Corresponding figures (object type)	Independent variables	Template-based	Contour-based
Figure 4.3 (worker)	Occlusion	115	102
Figure 4.4 (road roller)	Object scale	8564	2500
Figure 4.5 (bucket of a backhoe)	Illumination variation	723	101

4.1.4 Experiments on Template-Based and Point-Based Methods

The performance of the template-based and the point-based approaches is tested and evaluated based on the independent and dependent variables discussed in Chapters 4.1.1 and 4.1.2. Since it is impractical to compare a large number of existing tracking methods, one state-of-the-art method for each category which is referred to heavily by other methods in the same category is chosen as the representative of the category. The template-based method that incrementally learns the object appearance model of eigen-images (Ross et al. 2008), and a point-based method that uses the active shape model (Mathes and Piater 2006) are implemented. These two methods do not involve additional algorithms for the improvement of specific cases (e.g. occlusion), which may deviate the results from the general aspect of the category. Only the aspects related to the general characteristics of the category are investigated in the results. Tests are performed using two kinds of video sets - one with a highway construction site model of scale 1:87 (model videos) and the other taken at construction sites (site videos). The use of model videos

allows for sufficient data within a controlled environment. For example, to test the methods' ability to handle occlusion, the other factors such as illumination and the objects' size have to be kept constant, which is actually difficult to control with site videos. A two-tailed t-test is performed to compare, in a statistical way, the centroid error of the template-based method (E_k) and the point-based method (E_p). The p-values of (E_k-E_p) is calculated and presented together with the centroid errors and the number of successfully tracked frames.

4.1.4.1 Absolute Value of Illumination

In the tests of absolute illumination, six types of objects are used for model videos (brick, pipe, backhoe, car, crane, and truck) and five are used for site videos (concrete bucket, timber, dozer, wheel loader, and worker). There is no occlusion or severe changes in illumination/size of objects in all videos. For each video, five levels of illumination conditions are imposed by increasing or decreasing the intensity of the images. The default is made to have the average intensity of 128 (based on the 8 bits intensity images ranging 0-255). For two darker (brighter) levels, a constant intensity value (40% or 80% of 128) are added to (subtracted from) all pixels. The maximum and minimum values of the intensity are kept as 255 and 0, respectively. The loss of information occurs when the intensity value, which exceeds 255 or falls below 0 due to the addition or subtraction, becomes 255 or 0. Even though this artificial modification may not exactly replicate the phenomenon in real sites, it is possible to create the effect of illumination on the loss of information in the video frames. Also, consistent illumination conditions can be retained on all different videos. The video frames of the five illumination levels are shown in Figure 4.5, and the average error results of model videos and site videos are provided in

Tables 4.2 and 4.3, respectively. Another important measurement, the number of frames in which an object is successfully tracked, is also included in Table 4.3. Since both methods successfully track the object until the last frame of most model videos, the number of successfully tracked frames is not included in Table 4.2. Instead, the cases in which a method failed to track until the end of the video are indicated by the term 'Failed' in Table 4.2.

Table 4.2: The average errors (pixels) and p-values of error difference for the tests on illumination conditions (model videos)

Object (Total frame #) \ Illum. level		1	2	3	4	5
Brick (182)	Template-based	1.33	2.94	2.29	1.20	1.60
	Point-based	25.29	22.78	27.06	32.05	30.37
	p-value	0.0000	0.0000	0.0000	0.0000	0.0000
Pipe (128)	Template-based	2.27	1.87	2.03	2.03	1.86
	Point-based	44.80	42.46	43.16	42.65	38.85
	p-value	0.0000	0.0000	0.0000	0.0000	0.0000
Backhoe (159)	Template-based	2.33	3.10	1.83	1.83	2.36
	Point-based	Failed	14.18	10.46	17.18	19.31
	p-value	N/A	0.0000	0.0000	0.0000	0.0000
Car (144)	Template-based	2.59	2.15	1.31	1.90	1.72
	Point-based	6.05	5.00	4.43	5.27	7.07
	p-value	0.0002	0.0026	0.0000	0.0001	0.0037
Crane (104)	Template-based	2.28	1.41	1.55	1.88	1.62
	Point-based	5.26	4.41	4.55	4.49	3.93
	p-value	0.0063	0.0011	0.0028	0.0010	0.0080
Truck (151)	Template-based	3.99	6.77	3.17	3.26	2.33
	Point-based	Failed	14.43	12.21	13.08	12.06
	p-value	N/A	0.0005	0.0000	0.0000	0.0000

Table 4.3: The number of successfully tracked frames, the average errors (pixels), and p-values of error difference for the tests on illumination conditions (site videos)

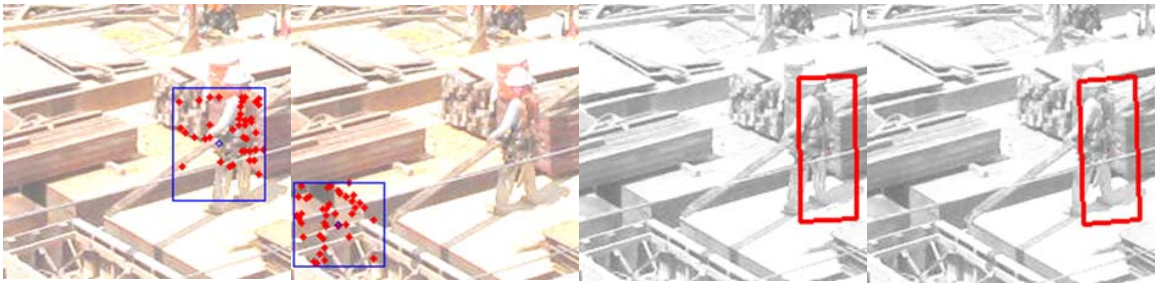
Object (Total frame #) \ Illum. level			1	2	3	4	5
Concrete Bucket (174)	Template-based	# of frames	174	174	174	174	174
		Avg. error	1.90	2.59	2.38	2.42	2.57
	Point-based		Failed	Failed	Failed	Failed	Failed
Timber (179)	Template-based	# of frames	86	88	179	179	179
		Avg. error	3.91	2.93	7.04	7.42	7.31
	Point-based		Failed	Failed	Failed	Failed	Failed
Dozer (277)	Template-based	# of frames	277	277	277	277	277
		Avg. error	14.56	15.95	16.40	16.60	17.64
	Point-based	# of frames	40	120	138	132	108
		Avg. error	19.08	26.53	13.12	25.21	25.19
	p-value for error diff.		0.5364	0.0203	0.3437	0.0516	0.0418
Wheel loader (294)	Template-based	# of frames	18	294	294	294	294
		Avg. error	10.75	24.21	10.73	11.14	11.05
	Point-based	# of frames	53	200	212	186	190
		Avg. error	56.41	40.58	43.40	52.01	46.40
	p-value for error diff.		0.0007	0.0005	0.0000	0.0001	0.0000
Worker (245)	Template-based	# of frames	140	140	140	140	140
		Avg. error	8.16	9.29	9.82	10.72	12.21
	Point-based	# of frames	108	122	122	116	84
		Avg. error	27.84	24.30	23.34	24.55	17.20
	p-value for error diff.		0.0002	0.0001	0.0007	0.0006	0.1830

The template-based method successfully tracks objects over all frames of model videos while the point-based method fails to track a backhoe and a truck in the darkest condition. The failure is attributable to the dark colors of the backhoe and the truck. The point-based

Figure 4.5: Five levels of illumination conditions (level 1 to 5 from the darkest to the brightest)

method extracts an extremely reduced number of feature points under darker illumination conditions. On the other hand, the template-based method maintains enough information to differentiate the object regions. Both methods have more difficulties in tracking objects in the site videos because the site videos have a less controlled environment and the objects are more distant from the camera. The overall results show that the template-based method outperforms the point-based method under the severely dark or bright condition (Figures 4.6 and 4.7).

The average errors in Tables 4.2 and 4.3 indicate the performance problem of the point-based method regardless of illumination conditions. The point-based method's errors are larger than the template-based method's in all cases. There are two primary reasons for this. First, depending on the spatial distribution of the detected feature points, the centroid of the tracked region in point-based method is prone to bias. In Figure 4.8(b), all feature points are extracted from the worker's back resulting in the centroid point

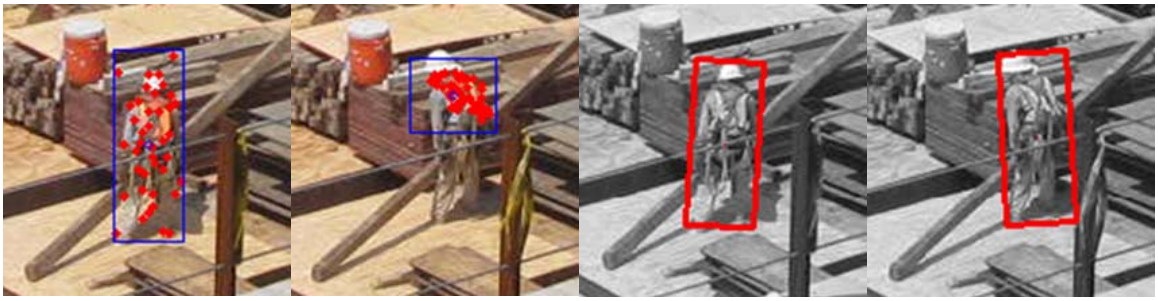


(a) (b) (c) (d)

Figure 4.6: The results of tracking a worker under level 5 illumination condition ((b) the frame at which the point-based method lost the object, (a) the frame previous to (b), (c) and (d) the template-based method's result corresponding to (a) and (b))

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

Figure 4.7: The results of tracking a dozer under level 1 illumination condition ((b) the frame at which the point-based method lost the object, (a) the frame previous to (b), (c) and (d) the template-based method's result corresponding to (a) and (b))



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

Figure 4.8: (a) and (b) the point-based method's results of the 10th and 50th frame, (c) and (d), the template-based method's results of the 10th and 50th frame (tracking a worker)

being biased towards the upper half. Second, in the case of tracking bricks or pipes carried by vehicles, the point-based method detects the points in the region of the vehicle and finally tracks the vehicle and the pipe together recognizing them as one object (Figure 4.9). This phenomenon also biases the centroid. Another notable fact is that the point-based method is not able to track a concrete bucket or timbers at all. The plain surface of the concrete bucket does not contain sufficient features and the point-based method is attracted more by the cluttered background region (Figure 4.10). Also, due to the small number of feature points extracted from the timbers, a small amount of error in matching corresponding points can corrupt the tracking.

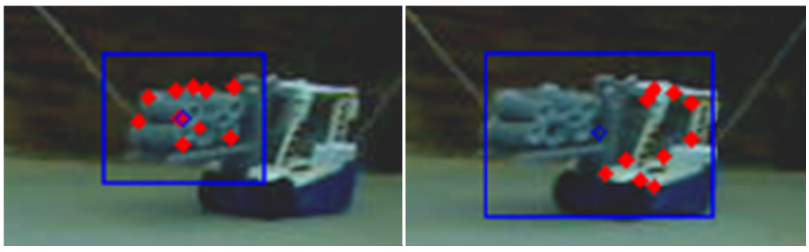


Figure 4.9: The point-based method's results of the 1st and 6th frame (tracking a pipe model)

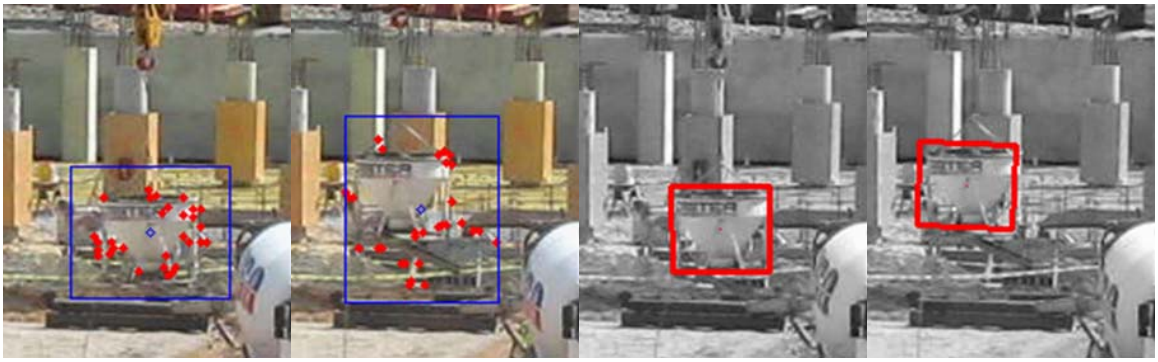


Figure 4.10: (a) and (b), the point-based method's results of the 2nd and 10th frame, (c) and (d), the template-based method's results of the 2nd and 10th frame (tracking a concrete bucket)

4.1.4.2 Illumination Variation

In this section, six kinds of objects for model videos (brick, pipe, backhoe, car, crane, and truck) and three for site videos (dozer, wheel loader, and worker) are used. Illumination variations are imposed on the videos by artificially manipulating the intensity of the images. Across the frames, the intensities are changed using a function defined as $I = I + 128 \times 0.6 \times \sin(2\pi \times f \times t)$, where I, f, and t are the intensity of a pixel, the frequency, and time in seconds, respectively. Three frequencies of 0.1 /s, 0.15 /s, and 0.2 /s are tested for the comparison. For each frequency, the gradient of the function (dI/dt) changes along the time, which creates diverse effects of illumination variation. Even though the illumination variation based on this equation may not exactly reflect the real sites' conditions, diverse types of variations can be created with the controlled frequency value. Because it is inferred from the previous section that the trackers may fail when 80% of the intensity value 128 is subtracted (illumination level 1), 60% of the intensity value 128 is selected as the height of the sine function. In this way, extremely dark or bright conditions are avoided in order to investigate only the effect of variations. The maximum and minimum values of the intensity are kept as 255 and 0, respectively. This modification generates controlled and consistent conditions for all videos. An example of the illumination variation that is artificially created is shown in Figure 4.11, and the resulting data are provided in Tables 4.4 and 4.5. The number of successfully tracked frames is not included in Table 4.4 since both methods successfully track the object until the last frame of all model videos.

It can be inferred from Table 3 and 4 that both methods are not heavily affected by the illumination variation itself. The numbers of frames that the template-based

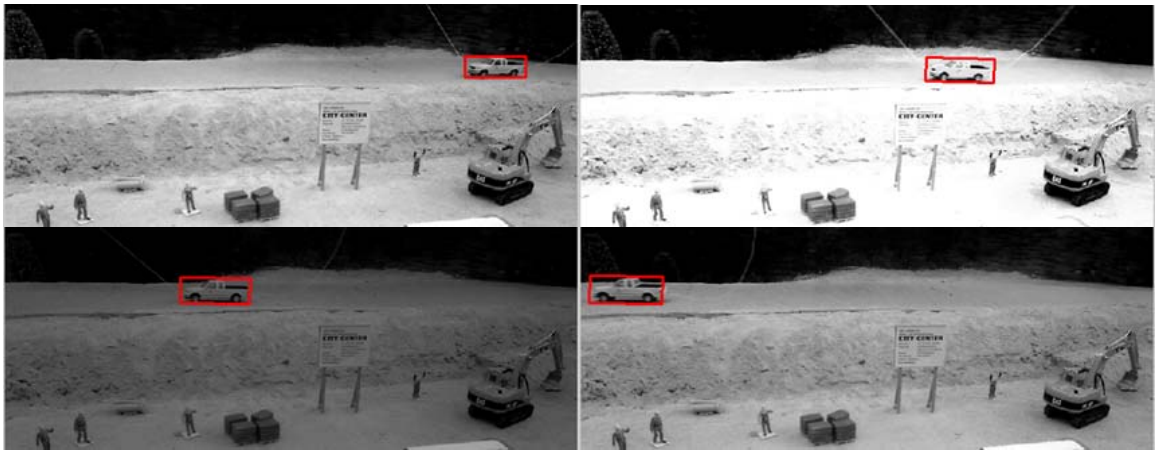


Figure 4.11: Illumination variation with 0.1 /s frequency imposed on the video of a car model

method successfully tracked are the same for all frequencies and the errors do not display any specific tendency. The point-based method also does not show any tendency in either the errors or the numbers of tracked frames. It is found that the variation causes higher chances for the entity to have low contrast to the background, regardless of the frequency. The more influential factor is when the variation causes the entity to look similar to the background at the moment, rather than the frequency itself. Also, if it becomes dark or bright when the part that has relatively less features is facing the camera, the tracking is likely to be more unstable. In Table 4.5, the errors and the numbers of frames of the point-based methods exhibit wider ranges than those of the template-based method, which indicates the point-based method is more sensitive to this factor. In most cases, the statistical analysis results in p-values lower than 0.01 which signify the superiority of the template-based method over the point-based method.

Table 4.4: The average errors (pixels), p-values of error difference for the tests on illumination conditions (model videos)

Object (Total frame #) \ Frequency		0.0 (level 1)	0.1 (level 2)	0.15 (level 3)	0.2 (level 4)
Brick (182)	Template-based	2.29	2.78	2.25	2.66
	Point-based	27.06	29.00	40.36	30.88
	p-value	0.0000	0.0000	0.0000	0.0000
Pipe (128)	Template-based	2.03	1.81	1.28	0.77
	Point-based	43.16	40.83	37.56	34.68
	p-value	0.0000	0.0000	0.0000	0.0000
Backhoe (159)	Template-based	1.83	2.65	2.95	3.12
	Point-based	10.46	35.30	26.61	34.80
	p-value	0.0000	0.0000	0.0000	0.0000
Car (144)	Template-based	1.31	2.29	2.21	2.29
	Point-based	4.43	6.32	5.66	5.25
	p-value	0.0000	0.0030	0.0005	0.0000
Crane (104)	Template-based	1.55	1.92	1.48	1.99
	Point-based	4.55	4.13	3.55	4.06
	p-value	0.0028	0.0108	0.0004	0.0087
Truck (151)	Template-based	3.17	3.23	2.80	2.21
	Point-based	12.21	9.19	9.64	8.93
	p-value	0.0000	0.0000	0.0000	0.0000

Table 4.5: The number of successfully tracked frames, the average errors (pixels), and p-values of error difference for the tests on illumination variations (site videos)

Object (Total frame #) \ Frequency			0.0 (level 1)	0.1 (level 2)	0.15 (level 3)	0.2 (level 4)
Dozer (277)	Template-based	# of frames	277	277	277	277
		Avg. error	6.86	7.30	7.55	7.77
	Point-based	# of frames	138	106	90	124
		Avg. error	13.12	14.79	15.77	19.76
	p-value		0.3437	0.0128	0.0004	0.0003
Wheel loader (294)	Template-based	# of frames	294	294	294	294
		Avg. error	11.18	10.45	11.40	9.09
	Point-based	# of frames	212	118	112	160
		Avg. error	43.40	43.18	34.07	42.56
	p-value		0.0000	0.0000	0.0002	0.0000
Worker (245)	Template-based	# of frames	140	140	140	140
		Avg. error	9.82	9.25	9.81	12.82
	Point-based	# of frames	122	196	220	104
		Avg. error	23.34	19.03	24.28	17.67
	p-value		0.0007	0.0014	0.0024	0.3894

4.1.4.3 Occlusion

For occlusion, six kinds of objects for model videos (brick, pipe, backhoe, car, crane, and truck) and one for site videos (dozer) are used. There are no severe changes in illumination and size of objects in all videos. For both model and site videos, occlusion is artificially created by drawing a black box at a fixed location on each video frame along

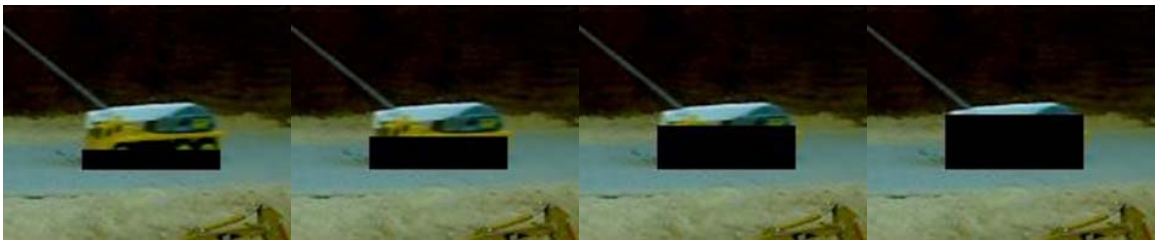


Figure 4.12: The four levels of occlusion (20%, 40%, 60%, and 80% - from left to right)

the object's travel path. Four levels of occlusion (20%, 40%, 60%, and 80%) are established by incrementally increasing the size of the box. The percentage of occlusion is determined by the ratio of the occluded area of the entity to the total area of the entity. In this way, consistent controlled occlusions are imposed on all videos. The examples of the four different levels are shown in Figure 4.12. The average error results of model videos and site videos are provided in Table 4.6. In the experiments for occlusion, it is important to compare whether or not a method lose the objects during the period of occlusion rather than the number of successfully tracked frames. Therefore, the number of successfully tracked frames is not included in Table 4.6, and the cases in which a method failed to track due to the occlusion are indicated by the term 'Failed'.

Brick, pipe and backhoe are disregarded for fair comparison in this section. When the pipes or bricks are occluded, the point-based method starts to extract the points from the vehicles that carry the pipes or bricks. That is, the point-based method tracks the vehicle and the pipes/bricks together as one entity relying on the points on the vehicle region which are not occluded. Also in the case of tracking the backhoe, although only the body part is selected to track in the first frame, the point-based method obtains the points on the arm and bucket regions by itself, and tracks relying on those points. Since the points also move in the same way, the method steadily transfers the interest of region from the

body part (pipes or bricks) to the arm and bucket (vehicles) which are not occluded (Figure 4.13). Even though this phenomenon increases the error due to the centroid's shift towards the region including the arm and bucket, it has a positive effect in that it avoids losing the entity completely.

Table 4.6: The average errors (pixels) and p-values of error difference for the tests on occlusions

Object (Total frame #) \ Occlusion level		20%	40%	60%	80%
Car (144)	Template-based	1.72	Failed	Failed	Failed
	Point-based	4.90	12.06	6.31	11.74
	p-value	0.0000	N/A	N/A	N/A
Crane (104)	Template-based	1.60	1.87	1.88	4.01
	Point-based	4.29	5.11	6.58	7.15
	p-value	0.0001	0.0008	0.0282	0.0585
Truck (151)	Template-based	2.74	2.80	5.55	Failed
	Point-based	9.78	17.80	15.80	20.99
	p-value	0.0000	0.0000	0.0016	N/A
Dozer (150)	Template-based	8.81	8.86	Failed	Failed
	Point-based	19.74	17.11	22.82	Failed

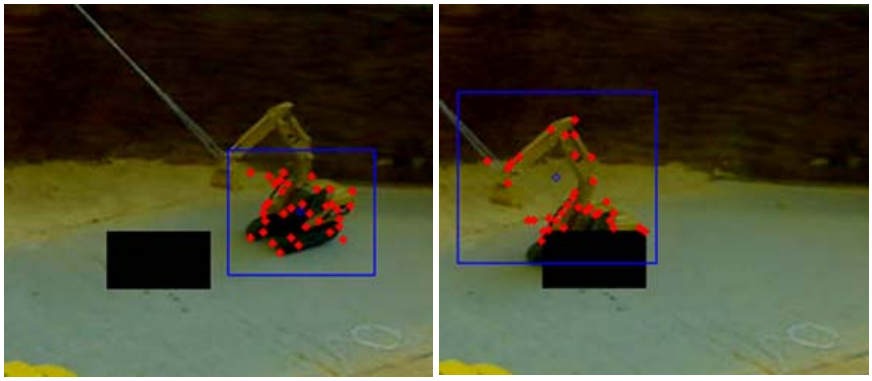


Figure 4.13: The 1st fame (left) and the 35th frame (right) of the tracking a backhoe with the point-based method

	p-value	0.0000	0.0159	N/A	N/A

In Table 4.6, it can be seen that the template-based method is more likely to fail in tracking under severe occlusion. The template-based method relies on the region of the object. Therefore, X % of occlusion provides X % reduced information of the object. On the contrary, since the point-based method relies on the points, it can conserve the amount of information by extracting the same number of points in the (100-X) % of the region. As long as the point-based method obtains the sufficient number of points from the object region, it could succeed in tracking.

However, in the cases that both methods successfully track the object, the template-based method shows much more accurate results than the point-based one. When occlusion occurrs, the template-based method tends to maintain the size of region,

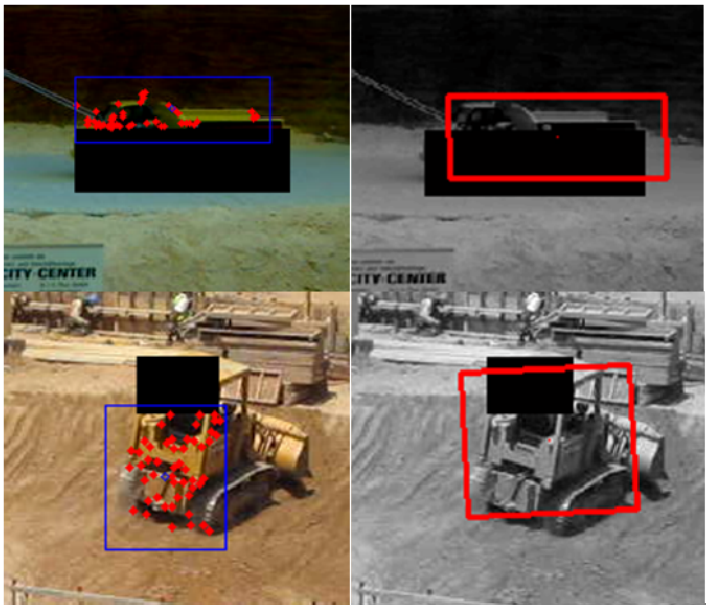


Figure 4.14: The 55th frame of tracking a 60% occluded truck model (the point-based method (upper left), the template-based method (upper right)), and the 46th frame of tracking a 40% occluded dozer in a real site (the point-based method (lower left) the template-based method (lower right))

while the point-based method reduces the size of region to include only the region that is not occluded (Figure 4.14).

4.1.4.4 Scale Variation

For scale variation, six objects are used for model videos (brick, pipe, backhoe, car, crane, and truck) and three are used for site videos (dozer, roller, and worker). In the videos, the object sizes change monotonically as they move away from the camera or towards the camera. The model dataset contains videos that show 75% decrease in the objects' scales, and the real-site dataset contains videos that show 81% decrease, 60% decrease and 37% increase of dozer, roller and worker, respectively. There are no occlusion and no severe changes in illumination. The results are provided in Tables 4.7 and 4.8. The number of successfully tracked frames is not included in Table 4.7 since both methods successfully track the object until the last frame of all model videos.

For the model videos, both methods successfully track the objects until the end of the videos. As it is shown in the previous sections, the template-based method shows higher accuracy than the point-based method. The results of the site videos exhibit more evidently the stability of the template-based method. The point-based method succeeds in tracking a worker who is walking toward the camera and getting larger in the video (Figure 4.15). However, the point-based method fails to track a dozer and a roller until the end of the videos in which the objects are getting smaller. When the object gets smaller with cluttered backgrounds, the point-based method extracts the feature points from the background region which is object region in the previous frame (Figure 4.16). This phenomenon exerts a harmful influence on the estimation of the movement.

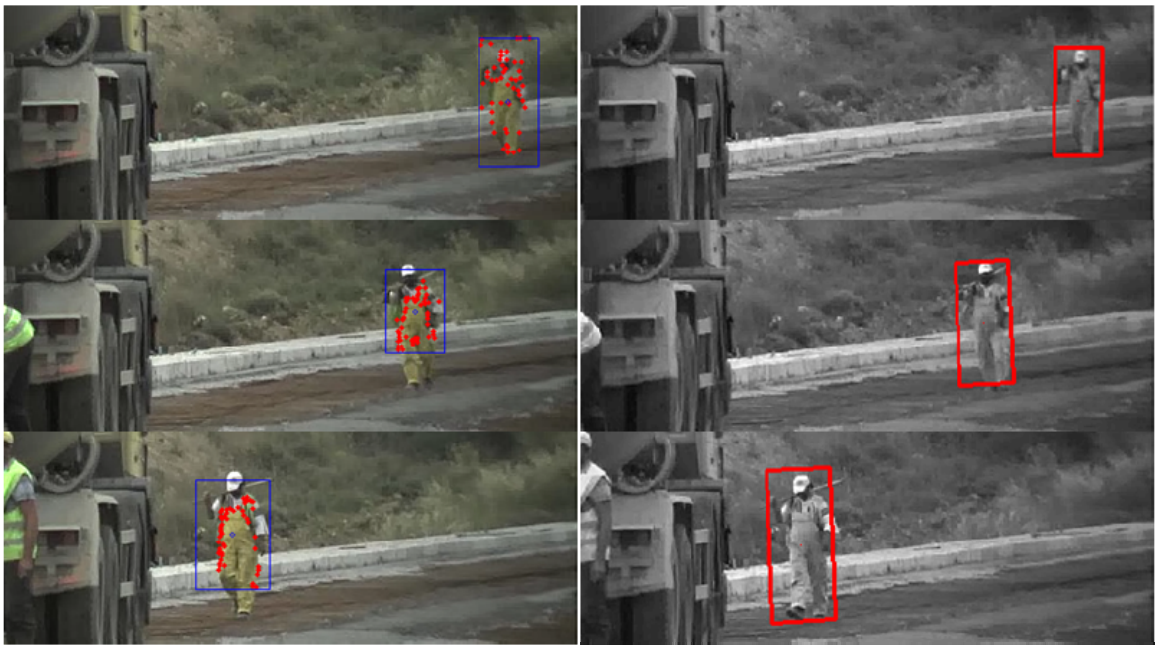


Figure 4.15: The results of tracking a worker with the point-based method (left column), the template-based method (right column) – the 1st, 42nd, and 84th (last) frames from top to bottom

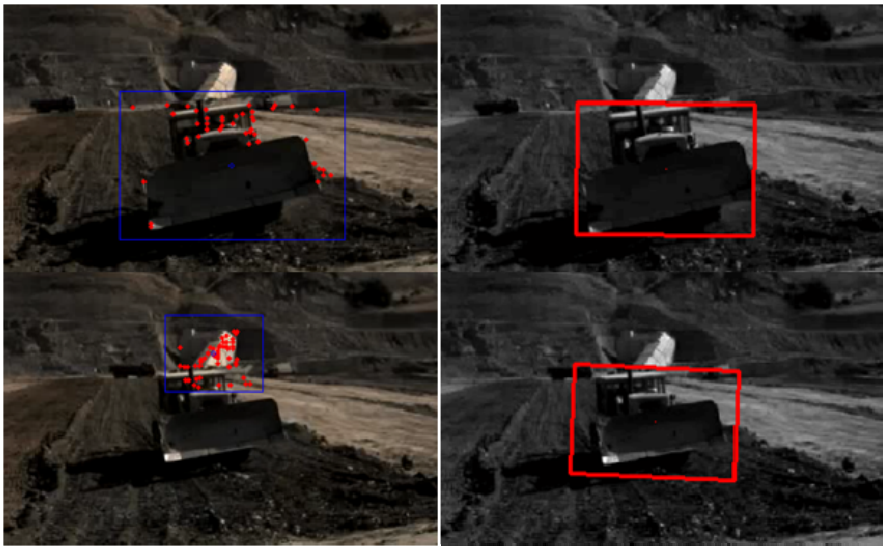


Figure 4.16: The results of tracking a dozer with the point-based method (left column), the template-based method (right column) – the 1st and 21st frames from top to bottom

Table 4.7: The average errors (pixels) and p-values of error difference for the tests on scale variations

Object (Total frame #)	Brick (326)	Pipe (256)	Backhoe (503)	Car (117)	Crane (264)	Truck (235)
Template-based	0.97	0.91	2.76	1.85	2.49	2.49
Point-based	8.57	3.78	14.46	2.41	4.52	8.66
p-value	0.0000	0.0000	0.0000	0.1213	0.0018	0.0000

Table 4.8: The number of successfully tracked frames, the average errors (pixels) and p-values of error difference for the tests on scale variations

Object (total frame #)		Dozer (266)	Roller (160)	Worker (163)
Template-based	# of frames	266	160	163
	Avg. error	12.88	5.70	2.77
Point-based	# of frames	20	92	163
	Avg. error	37.46	12.89	13.36
p-value		0.0049	0.0005	0.0000

4.1.4.5 Discussion

For thorough comparison of the template-based and the point-based methods, the characteristics of construction sites that can interfere with the performance of the vision tracking methods are recognized in order to identify metrics. Comparison tests are set up in such a way to measure the performance of tracking in relation with construction site requirements. Based on the tests, a number of experiments under different conditions of construction sites are performed. The template-based method turns out to be more stable and insensitive to illumination conditions, illumination variations, and scale variations

than the point-based method. Table 4.9 summarizes the comparison results. The point-based method shows its strength under occlusions. However, this strength is overshadowed by its inability to track the objects that do not have enough features on it. Overall, the template-based methods are determined as the most appropriate for tracking construction site resources.

Table 4.9: Determination of the best category between template-based and point-based methods

Variables	Absolute illumination	Illumination variation	Occlusion	Object scales
# of successfully tracked frames	Template-based	Template-based	Point-based	Template-based
Error of centroid	Template-based	Template-based	Template-based	Template-based

4.2 Combination of Detection and Tracking

Positioning of objects in a camera view is attainable through either object detection or object tracking. Object detection recognizes a certain type of objects (e.g. face and vehicle). For example, an algorithm of vehicle detection intends to recognize all vehicles in images. However, it does not provide the identification of each single vehicle. In other words, it cannot distinguish a vehicle from another. Also, the detection algorithms may fail to detect a positive object due to limitations on the training of all different views. On the contrary, object tracking finds a position of a unique object regardless of its object category, based on its previous location and appearance in video frames. Therefore, to initiate the tracking process, a prior position of an object has to be manually determined.

Accordingly, it is not feasible to achieve effective monitoring of construction sites by using either object detection or tracking because of their weaknesses such as false detections and manual initialization. However, their weaknesses can be complemented by each other's strengths. From this perspective of view, hybrid methods that combine the function of detection and tracking algorithms are presented in this chapter.

4.2.1 Methodology

Tracking always start with object detection. The object to be tracked has to be identified and its region needs to be specified. The tracking algorithm is triggered by taking the region of each object. The detection process is run for every frame not only to recognize new entities but also to adjust and improve tracking performance. In other words, detection and tracking are executed simultaneously. The proposed method basically relies on detection results for location. For each frame, the location results of tracking and detection are matched based on the distance between them. The tracking result is used only when there is no detection result matching with it. When detection results of consecutive frames show drastic changes in the object size or appearance, they are regarded as false positives and replaced with the corresponding tracking result. This reciprocal strategy enhances the stability of the processes. Furthermore, if detection results are missing for a certain length of time, the method regards an object disappeared (e.g. total occlusion or leaving the view) and stops processing. The identification of occlusion and automated termination prevent 2D tracking algorithms from failing and flying away from the object. Whenever the object gets free from the occlusion and appears in the view, the detection process newly initiates it again.

As described in Chapter 3.2 and 3.3, the detection method requires separate trainings of different views. Higher detection rate (recall) is achievable by increasing the number of views. However, it will demand considerably increased computational efforts. Even when taking a large number of views into account, it is hard to eradicate possibility of false detections due to illumination changes or signal noises. To resolve these problems, detection results are combined with tracking results. The combination allows higher detection rate with a smaller number of training views. Gaps between the trained views are filled with tracking results. In addition, tracking delivers identity of each object which is valuable when tracking multiple objects of the same type. Since separate templates of different views are used for detecting construction equipment, detection results can notify the changes of the view. Also, the detection process can alleviate instability of tracking algorithms caused by view changes (e.g. when equipment makes a turn).

4.2.2 Experiments and Results

The proposed method is implemented using Microsoft Visual C# in .NET Framework 4.0 environment. The implemented method is tested on two videos recorded with a wheel loader which is executing the loading/unloading process. They contain 959 (Video 1) and 773 (Video 2) frames, and they are resized to 800×600. The second video starts with a wheel loader's entering the view and includes two occlusion cases. The number of training images and template sizes involved in the detection is presented in Table 4.10. 30 principal eigen-images are used for each view. The detection algorithm is designed to maximize precision rather than recall, since low recall can be compensated by tracking results while low precision can cause tracking of false positive objects.

Table 4.10 The number of training images, and template sizes used for detection

Views	HOG			Eigen-images		
	# of training images		Template size	# of training images		Template size
	Positive	Negative		Positive	Negative	
Left & right	603	3000	144×96	603	4000	30×15*
Front & rear	412	3000	88×104	412	4000	20×20

*The template is constructed for lower half of the view

The experiment compares three methods: 1) detection-only, 2) tracking-only with manual initialization, and 3) detection combined with tracking (proposed method). Method 1 detected the wheel loader from 468 and 187 frames in Video 1 and 2, respectively. These values are significantly increased to 955 and 697 by using the proposed method, which result in 99.6% and 90.1% recall. Above all, no false positive was made on both test videos, which leads to 100% precision. The 76 (=773-697) frames of Video 2, in which the method fails to extract locations, are associated with total occlusions or the wheel loader's entering the view. Figure 4.17 shows examples of results which compare Methods 2 and 3 (the propose method). The 1st column of Figure 4.17 illustrates an occlusion case. The proposed method identified the disappearance of the wheel loader based on continuous absence of detection results, and the process stopped (no result in 3rd image). The process resumed when it appeared again (4th image). When the wheel loader made a turn, the proposed method appropriately switched over to the rear view, while Method 2 tended to keep its rectangle region (the 2nd column of Figure 4.17).

Figure 4.17: Results of the proposed method (red) and a tracking method (blue) under total occlusion (left column) and viewpoint change (right column)

4.3 Summary

2D vision tracking is the departure of this research. This research aims at automating initialization of 2D vision tracking and extracting 3D information by correlating multiple 2D tracking results or through coordinate conversion. In other words, 3D vision tracking

that calculates the depth information of objects depends on the results of 2D vision tracking. Selecting the best 2D vision tracking method for construction applications faces some challenges. First, there are a large number of 2D vision tracking methods with different capabilities and specifications. In addition, construction sites have unique tracking requirements, which must be considered in order to select a 2D tracking method that performs optimally under restrictions of construction sites. A systematic approach to compare different categories of 2D vision tracking methods is presented to find the most appropriate choice for 3D vision tracking purposes. The 2D vision tracking methods are categorized as contour-based, template-based and point-based methods.

To compare the three categories in a scientific way, independent variables and dependent variables of experiments are determined. Characteristics of construction sites that can affect the tracking performance are recognized and determined as independent variables – absolute value of illumination, illumination variations, occlusions, scale variations, and various types of objects. Two dependent variables – centroid error and the number of successfully tracked frames – measure the performance of methods. Methods of the three categories are tested on two data sets – model videos and site vidoes. Based on the results of executed tests, it is concluded that template-based methods are the best for tracking construction entities.

Both detection and 2D tracking methods can be used for locating construction entities in a camera view. A detection method locate all objects of a specific category in an image while a tracking method locate a unique object based on its location in previous frames. The combination of the two method leads to automated and stable 2D localization. The main role of detection is to initialize the tracking algorithm by

determining regions of objects to be tracked. Besides, there are additional benefits tracking process can take from detection results. First, intrinsic instability of tracking algorithms can be alleviated. When a tracked object experiences a drastic appearance change due to noise, illumination conditions, and making turns, the tracking may fail and lose the object. In this case, if a detection result is found near the tracked region, it can adjust the region to the correct position. Second, occlusion can be recognized, which automatically terminate the tracking process.

CHAPTER 5

CALCULATION OF 3D COORDINATES

2D pixel coordinates of object location determined by detection (Chapter 3) and 2D tracking (Chapter 4) need to be further processed to obtain 3D spatial coordinates. To achieve this, stereo camera calibration and triangulation are the required for a stereo camera system which is used for construction entities in this research. This research considers only fixed camera systems, and thus camera calibration is processed only once after the cameras are set up.

This chapter presents the method of combining 2D tracking results with stereo view geometry for the sake of accurate 3D trajectories of far-located construction entities. It should be noted that this research is aiming strictly for accurate localization of construction entities from multiple construction cameras at long camera-to-object distances, not real-time processing. Each single step involved in this method should be optimized to characteristics of the fixed camera system and construction sites such as various types of construction entities, the long baseline, and the long distance from cameras to an entity which is inevitable at large-scale construction sites.

5.1 Stereo Camera Calibration

To obtain depth information, this research adopts stereo camera systems. Camera parameters have to be discovered to establish geometry of the camera system. The camera system in this method is composed of two cameras located several meters apart from each other. The system is described by epipolar geometry as shown in Figure 2.2. The geometry consists of two types of parameters – intrinsic and extrinsic parameters. Intrinsic parameters are related to the system of each single camera while extrinsic

parameters are related to the relationship between two cameras. The following chapters will present methods of intrinsic and extrinsic camera calibration.

5.1.1 Intrinsic Calibration

Intrinsic parameters determine the linear system of projecting 3D points on the image plane (P_1 and P_2 in Figure 2.2). Bouguet's calibration toolbox (2004) is used to reveal the intrinsic parameters because of its accuracy, robust convergence, and convenience. Figure 5.1 shows several sample images of a checkerboard which are the input to the calibration toolbox. The image set should contain various angles of the checkerboard. For each input image, the toolbox also requires user input of information about the configuration of the checkerboard – the dimension and the number of the squares, the four extreme corners of the checkerboard pattern, and an initial guess of radial distortion. Then, all corner points of the squares are automatically extracted. Erroneous corner points can be refined by adjusting the initial guess of radial distortion. Using the user input, the toolbox calculates the focal length, the principal point, the skew coefficient, and the distortion coefficients. The toolbox provides an error analysis function through which corner points associated with large errors can be easily found and fixed.

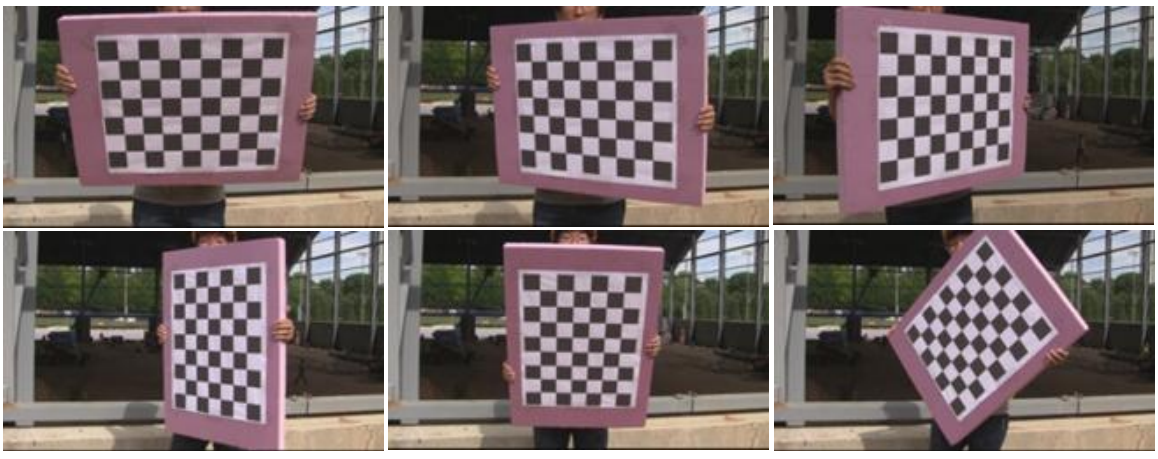


Figure 5.1: The example frames of a checkerboard video

5.1.2 Extrinsic Calibration

Once the linear systems of two cameras are revealed, the relation between the two cameras has to be estimated. In the stereo camera system, the focal point of the left camera becomes an origin of the coordinate. Extrinsic parameters represent the relative pose of the right camera to the left one (the rotation matrix R and the translation vector T in Figure 2.2). The estimation of R and T involves point matching between two views. Two combinations of algorithms are considered in this research. One is using SURF (Speeded Up Robust Features) (Bay 2008) and RANSAC (RANdom SAmple Consensus) (Hartley and Zisserman 2004) for the feature descriptor and outlier removal, respectively. This combination proved to be fast and accurate enough for point cloud generation of infrastructure (Fathi and Brilakis 2011). The other is using SIFT (Scale-Invariant Feature Transform) (Lowe 2004) and MAPSAC (MAximum a Posteriori SAmple Consensus) (Torr 2002), which is slower, but capable of acquiring more matches than the former combination. Even though the use of SIFT is slower than SURF, it is worth to consider this combination in our application because of the following reasons. First, cameras are fixed in our application, which requires the camera pose estimation only once at the initial stage of the framework. Therefore, the longer processing time of using SIFT can be ignored. Second, as a longer baseline line (distance between two cameras) is used, fewer point matches are obtained due to higher disparity between two camera views. In this case, SIFT and MAPSAC can be helpful to feed more inlier matches and less outlier matches to the next step.

The normalized 8-point algorithm (Hartley 1997) is selected to estimate the essential matrix based on intrinsic parameters and point matches. The selected method is the most widely used because of its simple implementation and reasonably accurate results. Although this method is less computational efficient and more sensitive to degeneracy problems compared to other methods (Nistér 2004, Li and Hartley 2006), it is still efficient and accurate enough to satisfy our needs with regard to fixed camera

positions, a long baseline, and the complexity of construction sites. Finally, extrinsic parameters (R and T) are recovered directly from the essential matrix (Hartley and Zisserman 2004). These parameters together with the intrinsic parameters are used for triangulating 2D tracking results.

5.2 Triangulation

The results obtained in two previous sections (epipolar geometry and two centroids) are fed into the triangulation step. Generally, the projections of two centroid coordinates determined from two views do not intersect each other due to camera lens distortions and errors caused by 2D tracking. Even if the 2D tracker correctly locates the entity on each frame, the disparity between two camera views causes mismatch of the centroids. In order to enhance the accuracy of the triangulation process, the two centroids had to be re-located so that their projections intersect (see Figure 2.2). For this purpose, Hartley and Sturm's algorithm (Hartley and Sturm 1997) is selected since the accuracy is more critical than the processing time in our application. Intersecting projections of the modified pair of centroids for each frame leads to the 3D coordinate of the tracked entity.

5.3 Summary

The calculation of 3D coordinates from 2D tracking results is covered in this chapter. For construction applications, two cameras are employed to obtain 3D spatial information by correlating two of 2D tracking results - one from each camera. Cameras calibration revealed intrinsic parameters of cameras by processing video frames of a checkerboard. The extrinsic parameters of two cameras were estimated through point matching between two views. The extrinsic calibration uses SIFT or SURF for feature point extraction, and RANSAC or MAPSAX for outlier removal. The discovered intrinsic and extrinsic camera parameters establish epipolar geometry. For every frame, two of 2D pixel

coordinates of an object, which are provided by a 2D tracking method applied to both cameras, are triangulated based on the constructed epipolar geometry, and 3D coordinates are calculated.

CHAPTER 6

VALIDATION

This chapter validates the results of the two frameworks (Figures 1.2 detailed in Chapters 3 through 5) through experiments and statistical analysis. The first framework of tracking construction entities are tested on site videos taken with a stereo camera system. Its performance is evaluated based on the accuracy of 3D location.

6.1 Experimental Design

The data for validation of 3D tracking are collected from a construction site at the Georgia Institute of Technology. This site is the construction of an indoor football practice facility managed by Barton Malow Company. The roof and columns of the steel-framed facility were already completed when the data were collected. The videos were taken with two HD camcorders (Canon VISXIA HF S100, 30 fps, 1920×1080 pixels) located about 4.5m above the ground on one side of the facility structure, where the ground area of the facility structure could be overlooked. Two lengths of the baseline (distance between two cameras) were tested – 3.8 m and 8.3 m. A total station (Sokkia SET 230RK3) was used to acquire comparable results of the entities' trajectories which are used to measure the accuracy of video-derived results.

Figures 6.1 shows total station data of the positions of the total station, the cameras, and pre-defined trajectories. The figure is a top view of the site layout in a total station coordinate system. In the total station coordinate system, x-axis, y-axis, and z-axis are along the horizontal, vertical, and depth directions, respectively. In Figure 6.1, the postions of right camera 1 and 2 are corresponding to the short and the long baseline system. Figure 6.2 shows the entities' trajectories from the views of the left camera and the right camera 2. In Figure 6.1 and 6.2, Trajectories 1 and 2 are composed of 10 and 8

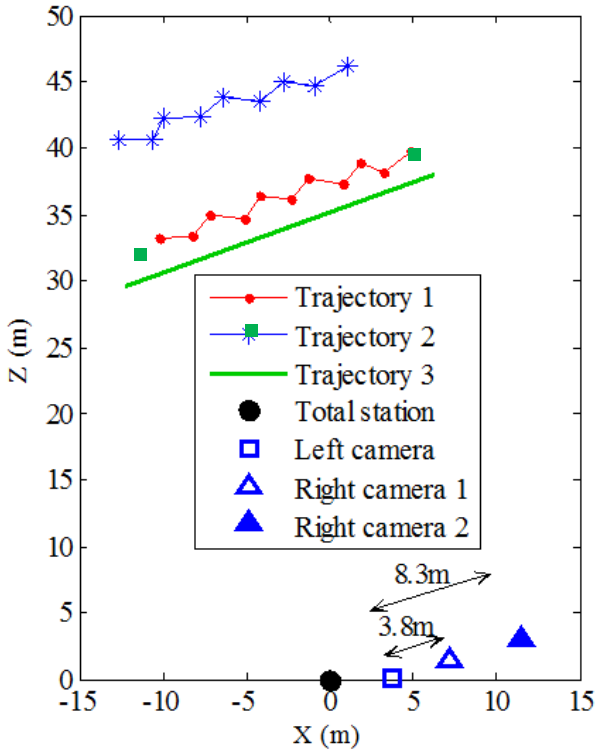


Figure 6.1: The layout of tests from a top view

segments of straight lines, containing 11 and 9 nodes, respectively. Trajectory 3 is a straight line. Trajectories 1, 2, and 3 are located approximately at a 39 m, 43 m, and 36 m distant from the left camera, respectively. The end points of all segments, i.e. 11, 9, and 2 nodes for Trajectories 1, 2, and 3 (●, *, and ■ in Figure 6.1), were pre-defined and clearly marked on the ground of the site. The position data of the nodes were acquired with the total station. At each node, a target mark of the total station was mounted on a tripod at approximate heights of the entities' centroids. Ground-truth trajectory data to be compared with vision tracking results are obtained by connecting the acquired position data of nodes with straight lines. The proposed methodology is tested on three types of entities (a worker, a steel plate carried by a worker, and a sport utility vehicle (SUV)). Trajectories 1 and 2 are those of a worker and a steel plate, and the trajectory 3 is of an

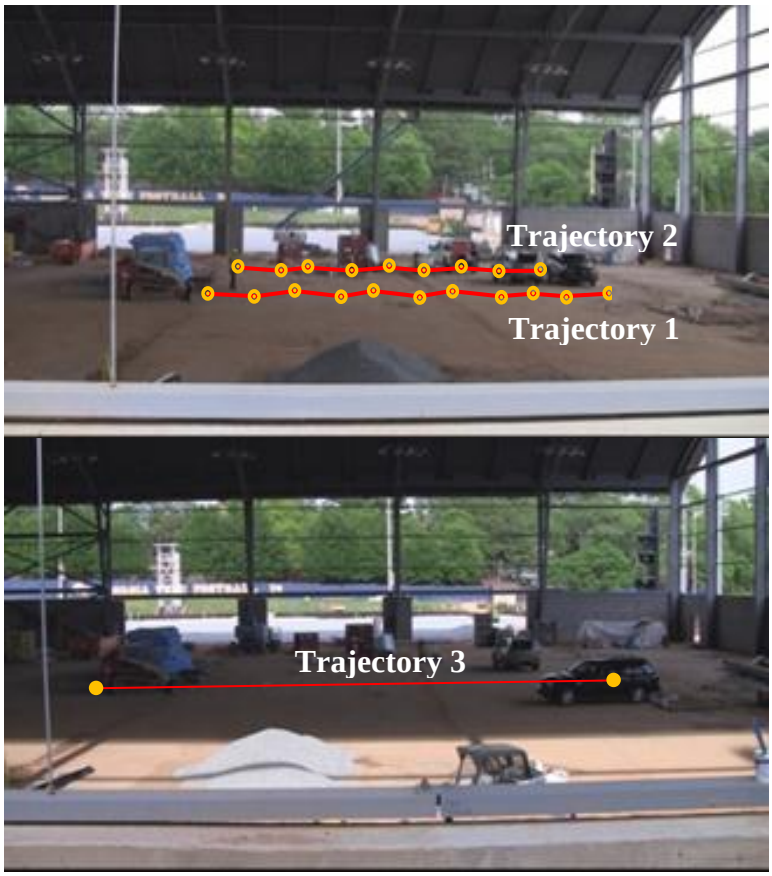


Figure 6.2: Trajectories 1 and 2 from right camera 1's view (top) and trajectory 3 from right camera 2's view (bottom)

SUV. The entities followed the marks on the ground (nodes of Trajectories 1, 2, and 3), moving straightly from node to node.

The accuracy of vision tracking is quantified by an absolute error that is defined as the distance between the tracked point and a line segment of ground-truth trajectory. For each frame j, the distance D_j is calculated by the following equation.

$$D_j \coloneqq \left|(\mathbf{Q}_{i+1} - \mathbf{Q}_i) \times (\mathbf{P}_j - \mathbf{Q}_i)\right| \div |(\mathbf{Q}_{i+1} - \mathbf{Q}_i)|$$

$$D_j\text{: distance between a point } \mathbf{P}_j \text{ and a line } \mathbf{L}_i = \overline{\mathbf{Q}_i\mathbf{Q}_{i+1}} = \mathbf{Q}_i + t(\mathbf{Q}_{i+1} - \mathbf{Q}_i)$$

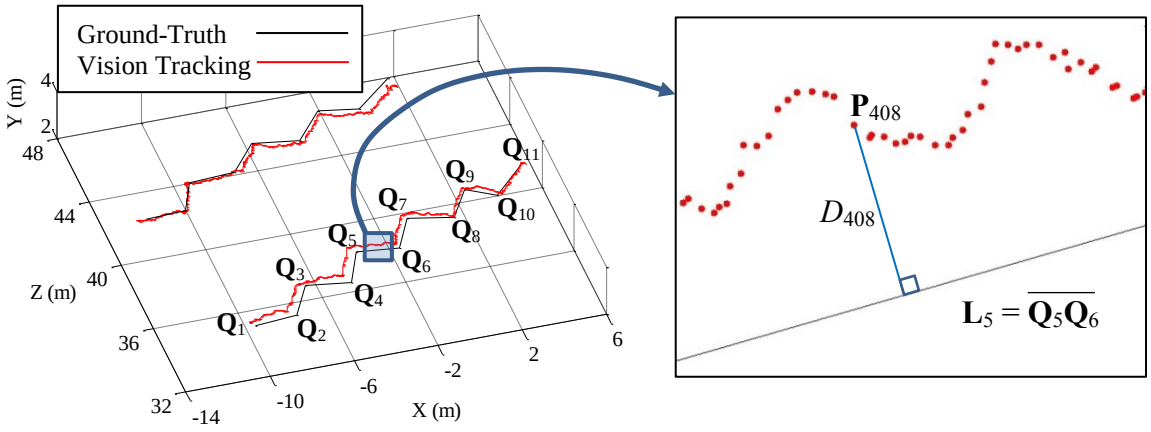


Figure 6.3: 3D tracking error calculation

where $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$ are end points of the i-th line segment $\mathbf{L}_i$ of ground-truth trajectories, on which the object in frame j lies. $\mathbf{P}_j$ is the j-th frame's tracking results, i.e. 3D points. Figure 6.3 illustrates the error calculation. The graph on the left of Figure 6.3 is an example of tracking results. As shown on the right of the figure, D_{408}, the error of the 408th tracked point, is determined as the length of the projection of $\mathbf{P}_{408}$ onto the line $\mathbf{L}_5$.

The main causes of error considered in this research can be classified into the 2D tracking error (Chapter 4) and the error of camera pose estimataion (Chapter 5.1.2). Also, the assumption that an entity moves exactly along the straight line is another miscellaneous cause of error.

6.2 Implementation

For the purpose of camera calibration, a video of a moving checkerboard (7 by 9 of 65mm×65mm squares) is recorded by each camera (Figure 5.1). 26 frames are selected appropriately to have various angles of view, and are fed into Bouguet's calibration toolbox (Bouguet 2004). Once the checkerboard videos are taken and the cameras are calibrated, all camera system settings are remained the same through experiments. All functions that may automatically cause a change in the camera intrinsic parameters, such

as autofocus and automated image stabilization, are disabled. Out of all video frames, a pair of corresponding frames of left and right cameras is used to obtain a large number of point matches. The point matches and calculated intrinsic parameters are used to estimate camera poses. Because the positions of the cameras are fixed in the proposed method, all calibration procedures are required only once as a pre-process.

For each calibrated camera view, construction entity is recognized and tracked across subsequent frames. While a worker and an SUV are detected by the proposed methods in Chapter 3.1 and 3.3, respectively, a steel plate is manually marked. Given the results of the comparative study (Chapter 4.1), a template-based 2D tracker based on Ross et al.'s method (2008) is used. The eigen-image is constructed selectively with gray scale values or saturation values depending on the tracked entity's color characteristics to enhance the accuracy. Also, in the particle filtering process, the position translation (delta-*x* and delta-*y* between consecutive frames) is considered instead of the entity location (*x* and *y* coordinates). This estimation strategy is beneficial to correctly locate the entity with fewer samples in particle filtering. The centroid coordinates are updated every frame by accumulating the estimated translation vector.

6.3 Experiments and Results

Experimental results of tracking a steel plate, a worker, and an SUV are presented in the following subsections.

6.3.1 Point Matching between Two Views

As described in Chapter 5.1.2, two point matching methods are tested – 1) SURF (Speeded Up Robust Features) with RANSAC (RANdom SAmple Consensus) and 2) SIFT (Scale-Invariant Feature Transform) with MAPSAC (MAximum a Posteriori SAmple Consensus). SURF is tested with two threshold values of distance ratio (DR), 0.8 and 0.6. Distance ratio is the distance of the closest neighbor to that of the second closest

neighbor (Lowe 2004). Discarding feature points that have distance ratios higher than the threshold, is an effective way of reducing false positive matches. In the case of DR=0.8, more point matches are obtained than if DR=0.6, but they contain apparent outliers as shown in Figure 6.4. In Figure 6.4, while most matching lines have similar slopes in left-

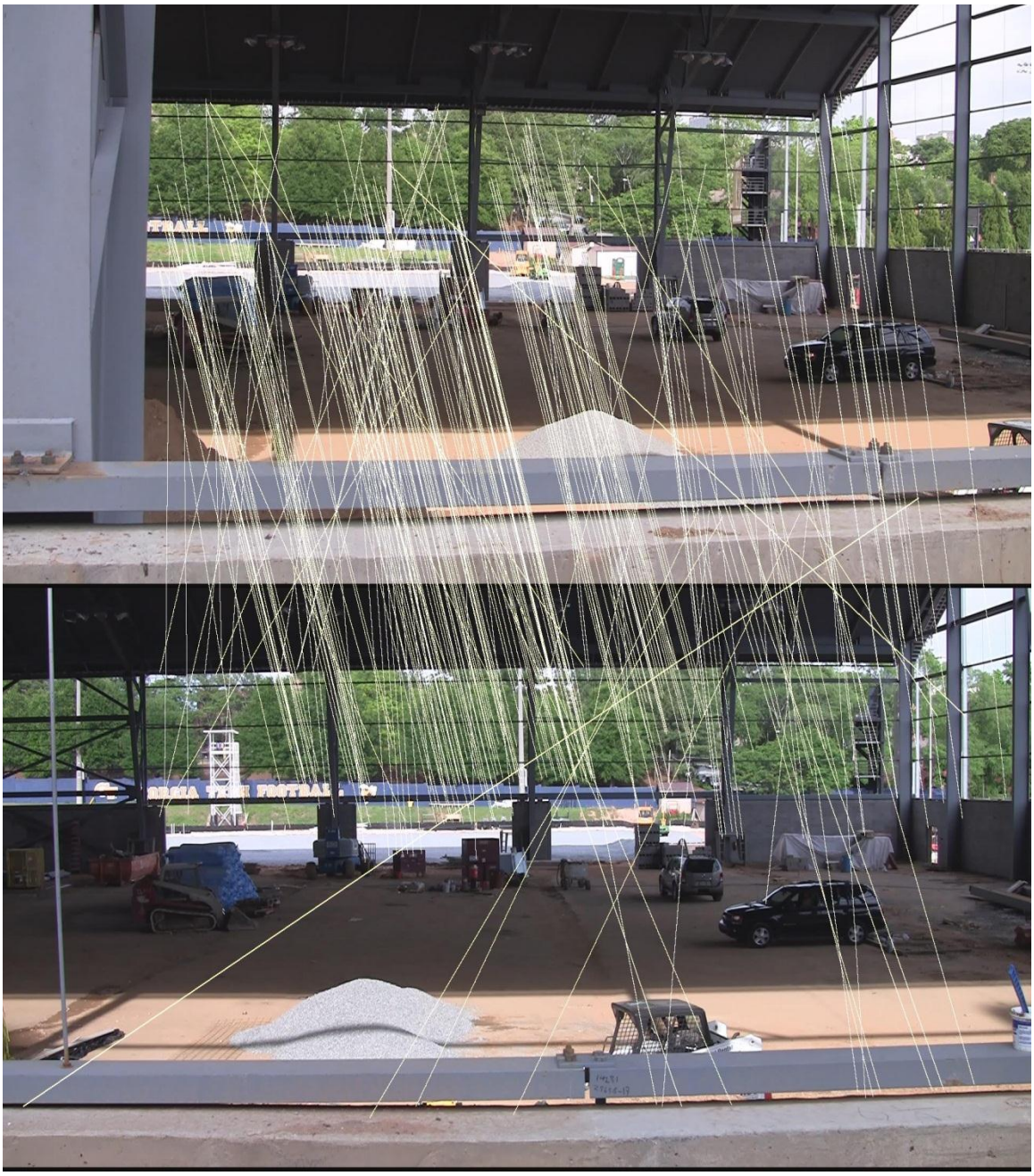


Figure 6.4: Point matches obtained by SURF+RANSAC (DR=0.8)

bottom direction (\), outliers direct in different way (/). The outliers have adverse effects on essential matrix estim ation, and the effect of outliers is reflected on the large error of tracking. On the other hand, in Figure 6.5 which shows point matching results of SIFT+MAPSAC, outliers are significantly reduced when compared to Figure 6.4.

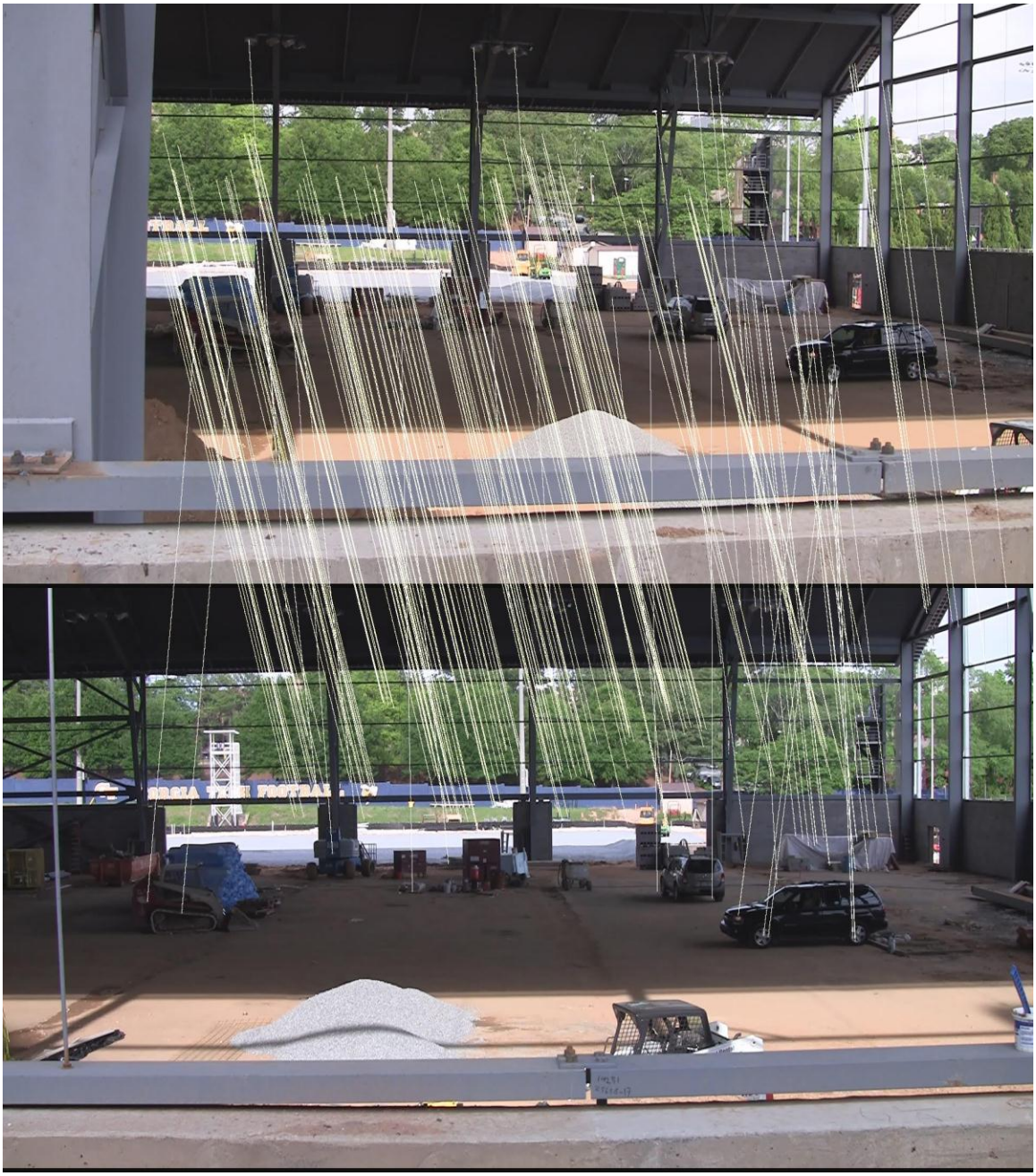


Figure 6.5: Point matches obtained by SIFT+MAPSAC (DR=0.6)

6.3.2 Tracking of a Steel Plate

A 0.6-m by 0.3-m steel plate is chosen as the first entity to track. The plate is carried by a worker walking along Trajectories 1 and 2. Tracking tests were performed separately for Trajectories 1 and 2. The video contains 1430 frames in total (790 and 640 frames for Trajectory 1 and 2, respectively), which means the results have 1430 tracked 3D coordinates. In this experiment, right camera 1 (Figure 6.1) is set to have a 3.8 m baseline. The template model for the 2D tracker is composed of gray pixel values. The tracker accurately fits the steel plate with an affine-transformed rectangle in most frames. Therefore, it can be inferred that the errors in this experiment mostly come from triangulation including camera pose estimation.

Figure 6.6 shows 3D tracking results. In Figure 6.6, black solid lines represent the ground-truth trajectories which are obtained by connecting total station data of nodes with straight lines. The black lines are same as Trajectories 1 and 2 in Figure 6.1. The other colors of lines are vision tracking results with different point matching methods. When using SURF and RANSAC with DR = 0.8, the results are far away from the ground-truth trajectories. The results of using DR = 0.6, on the other hand, are closely fitting the ground-truths. From the results of Trajectory 2, it can be observed that SURF+RANSAC (green) is a little closer to the ground-truths than SIFT+MAPSAC.

These observations are quantiried by the errors in Table 6.1. For every tracked position, the error, D_j, is calculated, thus, 1430 error data are obtained. The average and the standard deviation (STD) of the errors are computed. Maximum errors are calculated based on the average and STD with 95% confidence level. In Table 6.1, the errors are calculated for tracking along Trajectories 1 and 2, and also the total of them. Even though SURF with DR = 0.6 generates fewer point matches (271) than others (568 and 423), the method reduces outliers significantly and performs even better than SIFT+MAPSAC (DR=0.6), which provide about twice as many point matches. Assuming the error follows a normal distribution, it is concluded that the tracking error is less than 0.429 m with 95%

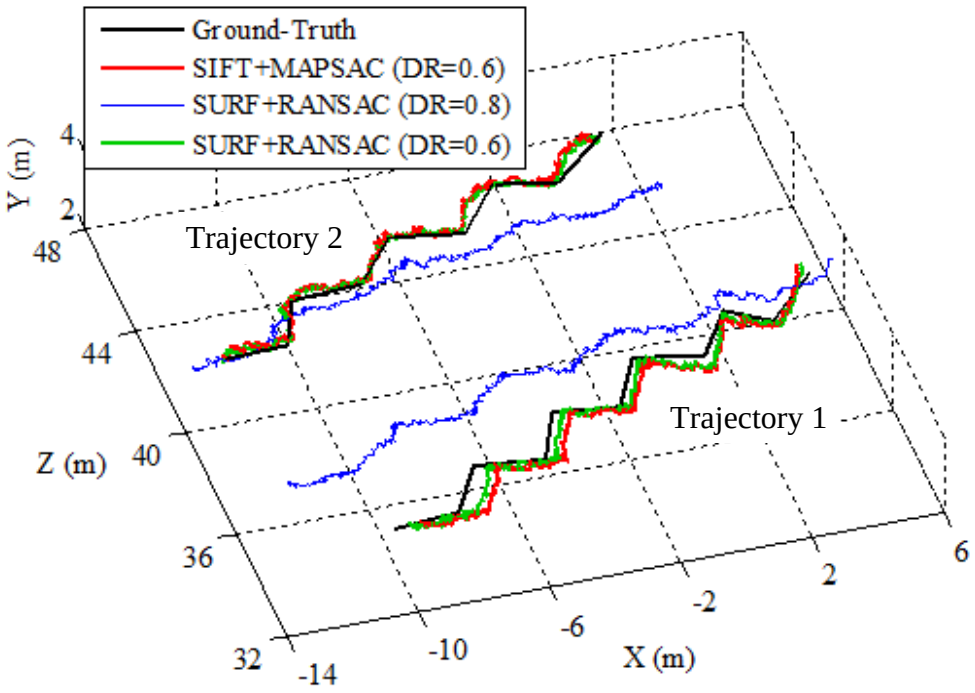


Figure 6.6: The tracking results of a steel plate

confidence. It is worthwhile to note that errors of Trajectory 1 is larger than those of Trajectory 2 even though Trajectory 1 is closer to the camera. It is inferred that the error disparity is related to the fact that most point matches are obtained from the area which are further than Trajectory 2 from cameras. That is, the points are closer to Trajectory 2 than Trajectory 1.

Table 6.1: Errors of tracking a steel plate

Method	DR	# of point matches	Error (m)								
			Total			Trajectory 1			Trajectory 2		
			Max	Avg.	STD	Max	Avg.	STD	Max	Avg.	STD
SIFT+ MAPSAC	0.6	568	**0.603**	0.252	0.179	**0.690**	0.314	0.192	**0.422**	0.177	0.125
SURF+ RANSAC	0.8	423	**3.005**	1.220	0.911	**3.463**	1.537	0.983	**2.043**	0.828	0.620
	0.6	271	**0.429**	0.180	0.127	**0.489**	0.222	0.136	**0.305**	0.127	0.091

6.3.3 Tracking of an SUV

The second experiment deals with the tracking of an SUV (2-m wide, 1.95-m high, and 5.13-m long). The SUV moved along Trajectory 3 forward and backward. The video contains a total of 1034 frames. A long baseline (8.3 m) is tested in this experiment placing a camera at 'right camera 2' in Figure 6.1. Gray pixel values are used for templates of the 2D tracker. Figure 6.7 displays obtained trajectories with ground-truths. Similar to the first experiment, it is observed that outliers finally result in inaccurate depth estimation (SURF+RANSAC with DR=0.8). It is worth to notice that tracking results of moving forward and backward are different from each other even though they were on the same trajectory. This disparity is caused mostly by the 2D tracking results. Figure 6.8 shows 2D tracking results in the right camera view. In the figure, a slight difference between the results of forward and backward movement is observed. This difference corresponds to the centroid error described in Chapter 4.1.2. The 2D tracking provides inconsistent centroids, which are linked to 3D tracking error.

The error results are presented in Table 6.2. Compared to the results in Table 6.1,

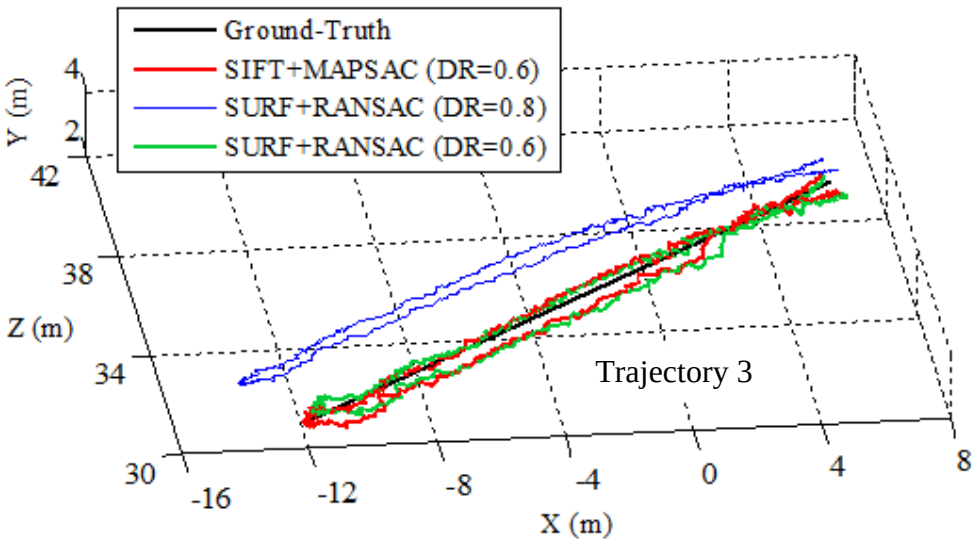


Figure 6.7: Tracking results of an SUV

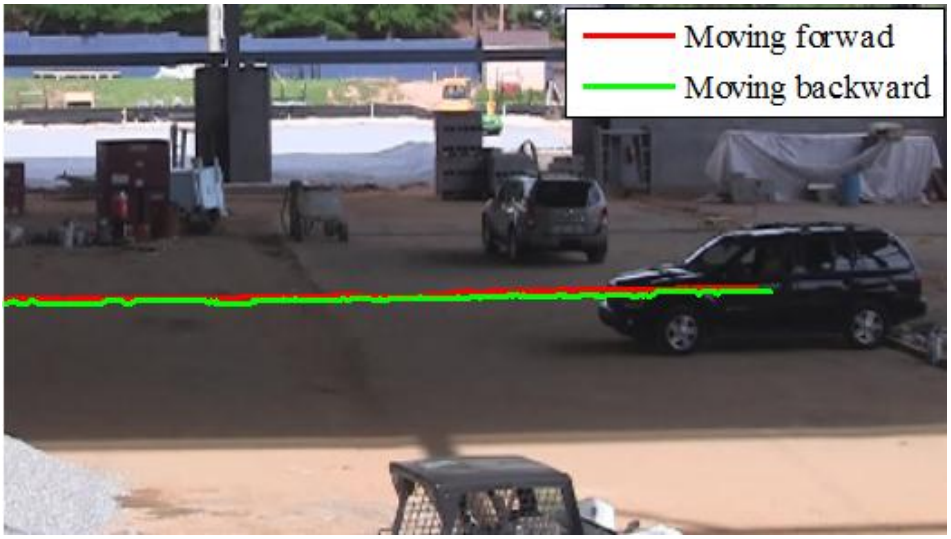


Figure 6.8: 2D tracking results in the right camera view

the number of point matches significantly reduced because the long baseline camera system is used. The small numbers of point matches are attributed to the greater difference between the left and right camera views. Different from the results of tracking a steel plate in the previous chapter (short baseline), SIFT+MAPSAC, which generated 26% more matches than SURF+RANSAC, performed better in this case (long baseline). It is inferred that though SURF+RANSAC (DR = 0.6) also contains few outliers, 183 point matches are not enough to accurately estimate extrinsic parameters. Assuming the error follows a normal distribution, it is concluded that the tracking error is less than 0.658 m with 95% confidence.

Table 6.2: Errors of tracking an SUV

Method	DR	# of point matches	Error - Trajectory 3 (m)		
			Max.	Avg.	STD
SIFT+MAPSAC	0.6	230	**0.658**	0.278	0.194
SURF+RANSAC	0.8	235	**1.068**	0.426	0.327
	0.6	183	**0.750**	0.289	0.235

6.3.4 Tracking of a worker

The third experiment is performed on a worker. The worker first walked along Trajectory 1, and then Trajectory 2. The worker moved straightly between the node markings on the ground. Two lengths of baseline (3.8 m and 8.3 m) are tested. The videos with a short and a long baseline contain 1435 and 1368 frames, respectively. In the 2D tracking process, the region of a worker's upper body, which can be well characterized by fluorescent colors of a hard hat and a safety vest, is tracked. Instead of gray pixel values, saturation values are used for composing the template model.

Figures 6.9 and 6.10 present the trajectory results of the short baseline system and the long baseline system, respectively. In both figures, it is observed that SIFT+MAPSAC (red) results are closer to the ground-truths than SURF+RANSAC (blue). More importantly, the results in Figure 6.10 (long baseline) show more stable and accurate tracking performance than the ones in Figure 6.9 (short baseline). The longer

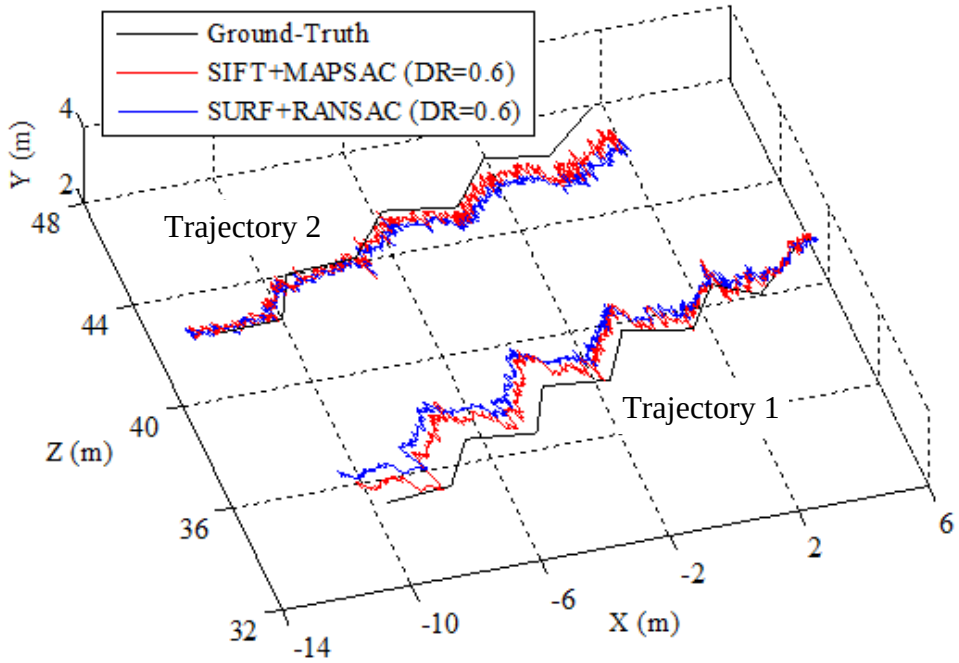


Figure 6.9: Tracking results of a worker with a short baseline

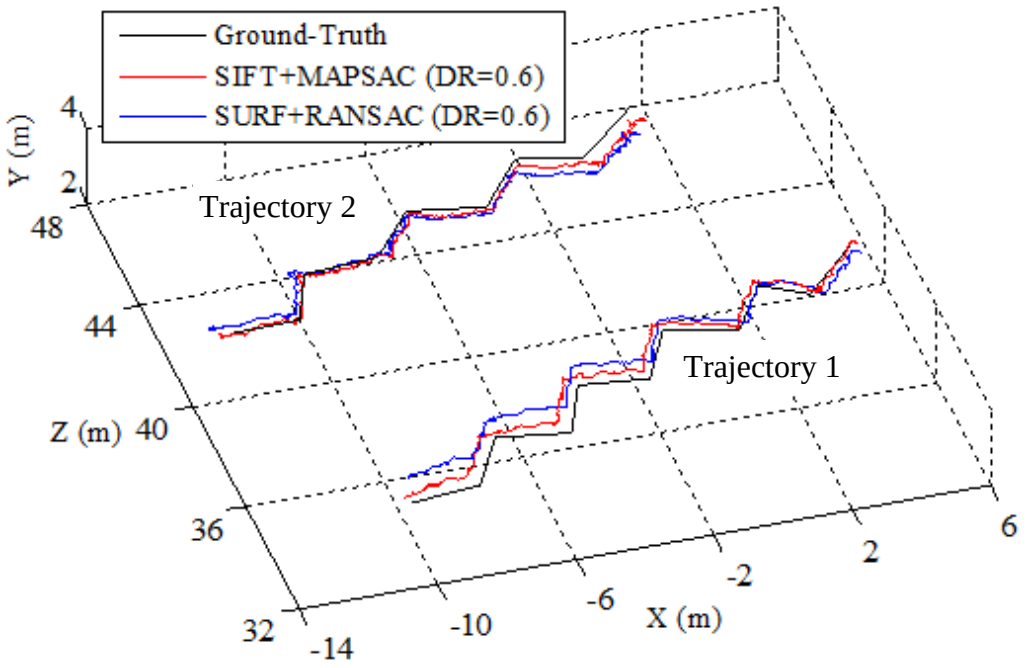


Figure 6.10: Tracking results of a worker with a long baseline

baseline forms a larger angle between two projections P_1 and P_2 in Figure 2.2, which results in higher accuracy. Table 6.3 summarizes tracking error results. As observed in Figures 6.9 and 6.10, SIFT+MAPSAC produces lower errors than SURF+RANSAC regardless of the baseline length. When a long baseline system is employed, SIFT+MAPSAC generates 215 point matches, which are 30% more than the matches generated from SURF+RANSAC. More accurate results are allowed by the larger number of point matches which leads to better estimation of the essential matrix (Chapter 5.1.2). In addition, when using SIFT+MAPSAC, errors of a long baseline system are significantly approximately half of those of a short baseline. Assuming the error follows a normal distribution, it is concluded that the tracking error is less than 0.636 m with 95% confidence.

Table 6.3: Errors of tracking a worker (DR=0.6)

Method	Base-line length	# of point matches	Error (m)								
			Total			Trajectory 1			Trajectory 2		
			Max.	Avg.	STD	Max.	Avg.	STD	Max.	Avg.	STD
1[i]	3.8 m	584	**1.223**	0.523	0.357	**1.338**	0.605	0.374	**1.032**	0.426	0.309
	8.3 m	215	**0.636**	0.258	0.193	**0.730**	0.317	0.211	**0.462**	0.187	0.140
2[ii]	3.8 m	503	**1.656**	0.714	0.481	**1.827**	0.841	0.503	**1.354**	0.562	0.404
	8.3 m	166	**1.010**	0.381	0.321	**1.188**	0.455	0.374	**0.708**	0.292	0.212

[i]Method 1: SIFT+MAPSAC
[ii]Method 2: SURF+RANSAC

It is worth to compare the short baseline results with the results of steel plate tracking. When SIFT+MAPSAC is used, the maximum errors of worker tracking and steel plate tracking are 1.223m (Table 6.3) and 0.603 m (Table 6.2). Because both results are from short baseline systems and the numbers of point matches (584 and 568) are only 0.2% different, it is inferred that the error difference is attributed to 2D tracking performance. 2D tracking errors can be divided into two elements. The first element is caused when the determined centroid in each view does not exactly match the real centroid, i.e. the total station target point. Tracking of a worker produces higher errors of the first element because the 2D tracker suffers severe variations of a worker's appearance whenever a worker changes one's direction. When compared with Figure 6.11(a), Figure 6.11(b) shows more substantial changes in the distribution of pixel values inside a rectangle. The second element is caused when two centroids from left and right cameras do not correspond to each other (Figure 6.12). 2D Tracking of a worker results in higher errors in both elements, leading to twice higher errors of 3D tracking than tracking of a steel plate.

(a) (b)

Figure 6.11: The appearance variations of (a) a steel plate and (b) a worker

(a) (b)

Figure 6.12: 2D tracking results: the 693rd frame of (a) the left and (b) the right camera

6.4 Summary

The first framework for tracking construction entities was evaluated based on the accuracy of determined 3D positions. The framework was implemented and tested on the videos recorded on a real construction site. A total station was used to acquire ground-truth data which were compared with vision tracking results. The tests involved 3 types of entities (a steel plate, a worker, and an SUV). A template-based 2D tracker was employed, and different methods of point match extraction were experimented to reveal the effect of errors caused by correlating multiple views. SIFT+MAPSAC provided a

larger number of point matches, which generally resulted in a good estimation of extrinsic parameters especially for long baselines. For tracking of a steel plate and an SUV, the maximum errors determined with 95% confidence were smaller than the entity's width. Various appearances of a worker (front, side, and, rear views) brought about larger errors of 2D tracking than tracking of a steel plate. However, it results in at most 0.658 m error with 95% confidence using a long baseline. The results validated that the vision based 3D tracking approach can effectively provide accurate localization of construction site entities with distance ranging around 40-50 m.

CHAPTER 7

CONCLUSION

This chapter first reviews the motivation and objectives of this research. Then, the brief descriptions of the methods created in this research are outlined. The conclusions, recommendations, possibilities of future research that grow out of this research are finally presented.

7.1 Review of Motivation and Objectives

This research proposes a framework of 3D vision-based tracking that promises to determine the spatial location of construcion entities without installation of any sensors. The main objective of this research is to test feasibility of the framework that can resolve two problems of general 2D vision tracking – the lack of automated initialization and depth information. Under this objective, the research effort in this study is focused on four parts: 1) detection of project related entities for automated initialization of 2D tracking, 2) comparison of 2D tracking methods to find the most appropriate ones for tracking construction entities, 3) combination of detection and tracking, and 4) stereo camera calibration for correlating 2D tracking results. The framework is expected to provide spatiotemporal information of construction entities in a large-scale congested construction site, such as construction equipment and personnel with comparable accuracy.

Vision-based tracking tracks moving objects by making inference on their location on the basis of visual and motion patterns. Vision-based tracking uses only camera videos, and is capable of tracking multiple entities concurrently. Wireless

surveillance cameras have been gradually applied to the construction site as cameras with higher quality and lower costs are available: hence, it is obviously beneficial to employ vision-based technology. It can be directly applied to video streams collected from the cameras, costing no additional equipment.

Vision-based tracking is frree of tags or sensors. It can track multiple entities without attaching any tags or sensors to the entities as long as they are present in camera views. On the contrary, radio frequency technologies such as RFID (Radio Frequency Identification), UWB (Ultra Wide Band), and GPS (Global Positioning System), which have been applied for tracking construction entities, need to install a sensor to each entity to be tracked. Though the radio frequency technologies work excellent for a certain type of materials and equipment for various construction scenarios, the drawbacks of each technology such as short ranges or the need for installation of sensors impose limits on their application to construction sites where a large number of entities exist. Therefore, vision-based tracking can be a promising alternative to the radio frequency technologies.

This research can contribute to the construction industry by establishing a basis of automated vision-based tracking. The automated tracking can be used for effective monitoring of the construction sites and project progress. More broadly, this research can help advance the level of automation in construction which is one of the greatest challenges of engineering noted by National Academy of Engineering.

7.2 Review of Methods

This research considers only fixed cameras, and the proposed tracking frameworks for construction and transportation applications are composed of four steps - 1) detection of

construction workers and equipment, 2) 2D tracking of detected entities, 3) integration of tracking and detection results, and 4) stereo or single camera calibration.

Detection processes of construction workers and equipment comprised of three sequential steps. First, a background subtraction method is used to reduce ROI (Region of Interest) of the second step to foreground blobs which are moving object regions. The second step searches shapes of workers or equipment from foreground blobs using HOG (Histogram of Oriented Gradients) features. The detected regions of the second step are further filtered based on color information (color histogram or eigen-images).

There are a great number of 2D vision-based trackers proposed in literatures which can be directly applied to construction entities. However, little is known about which one is the best for construction applications. To address this, a comparative study was undertaken. The methods are classified into the contour-based, template-based, and point-based methods, and the classes are compared regarding the construction sites' environments that can affect the tracking performance. Stability and accuracy are measured by the number of successfully tracked frames and the centroid error. Illumination conditions, the level of partial occlusions, and object scales are varied and controlled independently to investigate their effect on the tracking performance.

Proposed detection methods and the selected 2D tracking method are interated in such a way that detection initiates 2D tracking and increase the degree of stability of 2D tracking. Detection of construction equipment which involves four separate trainings for front, rear, left, and right, informs of view changes and adjusts 2D tracking results. Also, when no detection results are obtained around a tracking region for a certain amount of

time, the tracked object is considered to be occluded and the tracking process automatically terminates.

The combination of detection and tracking generates 2D pixel coordinates across frames. To obtain 3D coordinates, 2D coordinates from two views are correlated. First, cameras are calibrated to find their intrinsic parameters. Intrinsic parameters represent the linear system of projecting 3D points on the image plane. The second step is to estimate a relative pose (rotation and translation) of the calibrated cameras, which is called extrinsic parameters. Once the intrinsic and extrinsic parameters are known, 2D tracking results are triangulated based on the revealed parameters.

7.3 Discussion and Conclusions

All the methods proposed in this research study have been implemented in the Microsoft Visual Studio environment. Real videos of construction sites have been used to test these methods. The results from the methods are compared with ground-truth data retrieved manually or from other technologies to indicate the effectiveness of the methods. To validate the detection step, the proposed method of construction worker detection is tested. According to the results, the method can detect construction workers who wear safety vests with 99% precision within 1 s after they first appear in a camera view. The results indicate that the method can effectively initialize 2D tracking, which is the major role of the detection process.

According to the comparative study, template-based methods prove to be the best for tracking construction entities. Contour-based methods which have a merit of recognizing exact boundaries of objects do not fit to the role of 2D tracking in the 3D tracking framework. Experimental results show template-based methods are more stable

than the contour-based methods under partial occlusions and illumination changes. A thorough comparison of template-based methods and point-based methods shows the overall superiority of template-based methods over point-based methods. Point-based methods exhibit greater strength only in the stability under severe partial occlusions. Especially, the centroid errors of point-based methods are significantly larger than those of template-based methods for all different conditions in terms of illumination conditions, partial occlusions, and object scales.

The method of detecting construction equipment that uses four separate templates for different views is integrated with a selected template-based method. Tests of the integration show highly stable tracking results which are not achievable only with 2D tracking. In addition to the automated initialization, detection results are effectively used for adjusting and stabilizing 2D tracking results. It was also possible to automatically terminate the tracking process when the tracked object got occluded.

Finally, the 3D tracking framework is tested on real-site videos to validate the accuracy of 3D position data. The tests involve three types of entities: a steel plate, a worker, and an SUV (Sport Utility Vehicle). Various point matching methods and different baseline lengths are applied to identify their effects on accuracy. When the baseline is longer, higher accuracy is achievable since it reduces the error of triangulation. The experiments shows that the SIFT+MAPSAC method is more appropriate for the stereo camera system that has a long baseline since it generates more point matchings than SURF+RANSAC. The errors of 3D position are at maximum 0.658 m with 95% confidence. It validates the effectiveness, the accuracy and the applicability of the proposed vision based 3D tracking approach.

7.4 Contributions

This research will contribute to civil engineering community by providing an unobtrusive tracking method that can determine 3D positions of various types of objects across time. This research is expected to enable to improve the efficiency of monitoring both construction sites and traffic conditions. Furthermore, this research is also a step forward towards developing vision-based, automated, construction site model generation tools. The contributions of this research in tracking for construction are listed as follows.

1. This research automates the initialization of 2D tracking. Methods to detect construction entities (workers and equipment) are created, and used for initiating 2D tracking processes. The methods can trigger the 2D tracking process immediately once a construction entity newly appears in the camera view.
2. This research suggests the best methods to track on-site construction entities in 2D. The results of the comparative study can be referred to for any purpose that involves tracking construction entities. In addition, the experimental setup (independent and dependent variables, and video datasets) in the study can be used for further comparison of 2D trackers which are or will be proposed more recently or in the future.
3. This research investigates on extrinsic calibration methods for long baseline stereo camera systems. For monitoring large-scale construction sites, the baseline of stereo camera system should be long enough to improve the accuracy of triangulation. The experiments of this research provide findings on appropriate point matching algorithms that extract a sufficient number of point matches in long baseline systems.

7.5 Limitations and Recommendations for Future Work

This research investigates vision-based tracking which provides 4D coordinates (3D spatial position and time) of construction entities in camera views using appropriate image processing and machine vision techniques. This chapter will deal with limitations of this research and future work to enhance the proposed framework and overcome the limitations.

As presented in Chapter 6, experimental tests of the prosposed framework result in approximately 0.7 m error in maximum with 95% confidence level. The experiments were performed in a controlled condition (single object and no occlusion), and the level of error can be increased in a more complicated condition. The enhancement of 2D tracking and triangulation can reduce the error level.

Parts of 3D tracking error stem from 2D tracking error. Consistent centroids and pinpoint matching between two centroids from different camer views are the main targets to minimize the 2D tracking error. Combination of a point-based method and a template-baed method can be a promising way to achieve the targets. Also, an additional comparative study of new emerging 2D tracking algorithms will help find better algorithms and increase accuracy. Further categorization of template-based methods and their comparison based on the experimental design and dataset presented in this research will enable to find methods with better performance. Just by replacing the 2D tracking package with a new one, the accuracy and stability of 3D tracking can be enhanced.

The use of three or more cameras can improve the 3D tracking performance. First of all, chances of an object viewed in at least two cameras are increased. Also, the error of triangulation can be reduced. Furthermore, an investigation of optimal camera

positions can be an important study to improve the performance since the accuracy of triangulation is sensitive to the camera system.

Processing of the implemented framework is not real-time. Therefore, code optimization and employment of GPU (Graphics Processing Unit) are required to achieve real-time processing. Real-time processing will allow wider applications of vision tracking in construction project monitoring. For example, real-time information of on-site entities can be used to identify proximity of worker to heavy equipment or restricted zones.

Experiments in this research are limited yet to validate the possibility of practical monitoring applications. Experiments on more complicated scenes and scenarios are required to find and resolve problems of vision-based tracking implicated in tracking on-site entities. Various parameters such as site scale, crowdness, object speed, and activity type, etc. can be considered in the experiment design. Furthermore, benchmarking studies of tracking technologies (vision-based tracking and radio frequency technologies) can provide suggestions on the best tracking technologies for each single combination of the parameters.

The ultimate future goal of extending this research is practical application of 3D vision tracking to real tasks of site monitoring. As discussed in Chapter 2.2, 2D vision tracking has been already applied to simple scenarios for productivity measurement (Gong and Caldas 2010) and safety management (Teizer and Vela 2009) with a limited level of automation. Extended 3D vision tracking methods will broaden applications of vision tracking, and contribute to the construction industry by automating the acquisition of important information from videos for more complex tasks of project monitoring. The

3D vision tracking can enhance quality of construction projects by providing an efficient site monitoring system which contractors can easily adopt and apply to their on-site construction camera systems.

APPENDIX A

CODE FOR DETECTION

This appendix presents the main part in the prototype code for detecting construction entities. It includes the background subtraction, color histograms and HOG.

```
namespace ObjectTracking
{
   public class Detector: IDisposable
   {
      private bool isTrained = false;
      private List<Rectangle> detected;

      private HOGDescriptor hog;
      public Size winSize;
      private Size winStride, cellSize;
      private int nHOGbins;

      private double[] aspect;

      private BlobCounter blobCounter;
      private List<Rectangle> blobs = new List<Rectangle>();

      private bool _disposed = false;

      public Detector()
      {
         detected = new List<Rectangle>();
          winSize = new Size(64, 128);
          winStride = new Size(16, 16);
          cellSize = new Size(8, 8);
          nHOGbins = 9;
          hog = new HOGDescriptor(winSize, new Size(16, 16), cellSize, cellSize,
                                                  nHOGbins, 0, -1, 0.2, true);
      }

      public void FGBlobExtraction(List<Rectangle> entry, List<Image<Gray, byte>> fg,
                                   Option opt)
      {
         blobs.Clear();
```

```
for (int i = 0; i < entry.Count; i++)
{
   blobCounter = new BlobCounter(fg[i].Bitmap);
   Rectangle[] blobRect = blobCounter.GetObjectsRectangles();

   for (int j = 0; j < blobRect.Length; j++)
   {
      if (blobRect[j].Width >= opt.th_Blobs && blobRect[j].Height >=
                                                                  opt.th_Blobs)
      {
         blobRect[j].Offset(entry[i].Location);
         blobs.Add(blobRect[j]);
      }
   }
}

bool needToMerge = true;
while (needToMerge)
{
   int count = blobs.Count;
   needToMerge = false;
   for (int i = 0; i < count; i++)
   {
      for (int j = i + 1; j < count; j++)
      {
         if (blobs[i].Contains(blobs[j]))
         {
            blobs.RemoveAt(j);
            needToMerge = true;
            count = blobs.Count;
         }
         else if (blobs[j].Contains(blobs[i]))
         {
            blobs.RemoveAt(i);
            needToMerge = true;
            count = blobs.Count;
            break;
         }
         else if (blobs[i].IntersectsWith(blobs[j]))
         {
            int x = Math.Min(blobs[i].X, blobs[j].X);
            int y = Math.Min(blobs[i].Y, blobs[j].Y);
            int w = Math.Max(blobs[i].Right, blobs[j].Right) - x;
            int h = Math.Max(blobs[i].Bottom, blobs[j].Bottom) - y;
```

```
                blobs.RemoveAt(j);
                blobs.RemoveAt(i);
                blobs.Insert(i, new Rectangle(x, y, w, h));
                needToMerge = true;
                count = blobs.Count;
                i--;
                break;
            }
        }
    }
}

public void TrainHOG()
{
    hog.SetSVMDetector(GetHOGData());
    isTrained = true;
}

public float[] GetHOGData()
{
    float[] sv;
    String str_size = "(" + winSize.Width + "x" + winSize.Height + ")";
    System.IO.StreamWriter sw;
    System.IO.StreamReader sr;

    String[] files =
            System.IO.Directory.GetFiles(System.IO.Directory.GetCurrentDirectory(),
                                                "sv" + str_size + ".txt");
    if (files.Length != 0)
    {
        sr = new System.IO.StreamReader(files[0]);
        String[] str = sr.ReadToEnd().Split(new String[] { "\n" },
                                        System.StringSplitOptions.RemoveEmptyEntries);
        sv = new float[str.Length];
        for (int i = 0; i < str.Length; i++)
        {
            sv[i] = float.Parse(str[i]);
        }
        sr.Close();
        return sv;
    }

    files = System.IO.Directory.GetFiles(System.IO.Directory.GetCurrentDirectory(),
            "model_HOG_reg" + str_size + ".txt");
```

```
if (files.Length == 0)
{
   files = System.IO.Directory.GetFiles(
       System.IO.Directory.GetCurrentDirectory(), "trData_HOG"+str_size+".txt");
   if (files.Length == 0)
   {
      String[] str_p = System.IO.Directory.GetFiles(
                        @"D:\\[Detection]\\ObjectType\\Positives" + str_size);
      String[] str_n = System.IO.Directory.GetFiles(
                        @"D:\\[Detection]\\ \ObjectType\\Negatives");

      int lp = str_p.Length;
      int ln = Math.Min(lp * 3, str_n.Length);

      Image<Bgr, byte> images;

      int[] X = new int[lp + ln];
      float[][] des = new float[lp + ln][];

      ContinuousUniform distribution  = new ContinuousUniform();

      distribution.RandNumGen = new Random.MersenneTwister();

      for (int i = 0; i < lp + ln; i++)
      {
         String str = (i < lp) ? str_p[i] : str_n[i - lp];
         int ist = str.LastIndexOf('\\') + 1;
         int iend = str.LastIndexOf('.');
         String newfile = str.Substring(ist, iend - ist);

         images = new Image<Bgr, byte>(str);

         if (i < lp)
         {
            des[i] = HOG.Compute(images, winStride, new Size(0, 0), null);
         }
         else
         {
             int wrange = images.Width - winSize.Width;
             int hrange = images.Height - winSize.Height;

             int x = (int)(distribution.Sample() * wrange);
             int y = (int)(distribution.Sample() * hrange);

             Rectangle rect = new Rectangle(x, y, winSize.Width,  winSize.Height);
             Image<Bgr, byte> randImg = images.Copy(rect);
```

```
                des[i] = HOG.Compute(randImg, winStride, new Size(0, 0), null);
                randImg.Save("rand_" + newfile + ".png");
            }

            X[i] = (i < lp) ? 1 : 0;
        }

        sw = new System.IO.StreamWriter("trData_HOG" + str_size + ".txt");
        for (int i = 0; i < X.Length; i++)
        {
            if (des[i] != null)
            {
                String data = X[i].ToString() + " ";
                for (int j = 0; j < des[i].Length; j++)
                {
                    data += (j + 1).ToString() + ":" + des[i][j].ToString() + " ";
                }
                sw.WriteLine(data);
            }
        }
        sw.Close();
    }

    System.Diagnostics.ProcessStartInfo info
                = new System.Diagnostics.ProcessStartInfo(
                        "svm_learn", "-z c –t 0 -a alpha.txt trData_HOG" +
                        str_size + ".txt model_HOG_reg" + str_size + ".txt");

    info.RedirectStandardOutput = false;
    info.UseShellExecute = false;
    info.CreateNoWindow = false;

    try
    {
        using (System.Diagnostics.Process proc =
                    System.Diagnostics.Process.Start(info))
        {
            proc.WaitForExit();
        }
    }
    catch
    {
    }
}
```

```
    sr = new System.IO.StreamReader("model_HOG_reg" + str_size + ".txt");

    sr.ReadLine();
    sr.ReadLine();
    sr.ReadLine();
    sr.ReadLine();
    sr.ReadLine();
    sr.ReadLine();
    sr.ReadLine();

    String[] strs = sr.ReadLine().Split(new String[] { " " },
                                        StringSplitOptions.RemoveEmptyEntries);
    int hogSize = int.Parse(strs[0]);
    sv = new float[hogSize + 1];

    sr.ReadLine();
    strs = sr.ReadLine().Split(new String[] { " " },
                               StringSplitOptions.RemoveEmptyEntries);
    int nVectors = int.Parse(strs[0]);
    strs = sr.ReadLine().Split(new String[] { " " },
                               StringSplitOptions.RemoveEmptyEntries);
    sv[hogSize] = float.Parse(strs[0]);

    for (int i = 0; i < nVectors - 1; i++)
    {
        strs = sr.ReadLine().Split(new String[] { " " },
                                   StringSplitOptions.RemoveEmptyEntries);
        float alpha_y = float.Parse(strs[0]);
        for (int j = 0; j < hogSize; j++)
        {
            float scalar = float.Parse(strs[j + 1].Substring(strs[j + 1].IndexOf(':') + 1));
            sv[j] += alpha_y * scalar;
        }
    }
    sr.Close();

    sw = new System.IO.StreamWriter("sv" + str_size + ".txt");
    for (int j = 0; j < hogSize + 1; j++)
    {
        sw.WriteLine(sv[j].ToString());
    }
    sw.Close();

    return sv;
}
```

```
        public void WriteResults(Image<Bgr, byte> im, Option opt, long timestamp)
        {
            System.IO.StreamWriter sw = new System.IO.StreamWriter("Detection.txt", true,
                                                    System.Text.Encoding.UTF8);
            for (int i = 0; i < detected.Count; i++)
            {
                im.Draw(detected[i], TemplateTracker.colorValue(opt.rect_color),
                            opt.rect_thickness);
                sw.WriteLine(timestamp.ToString() + " " + detected[i].X.ToString() + " " +
                                detected[i].Y.ToString() + " "
                                 + detected[i].Width.ToString() + " "
                                 + detected[i].Height.ToString());
            }
            sw.Close();
        }
}
```

APPENDIX B

CODE FOR 2D TRACKING

This appendix presents the main part in the prototype code for 2D tracking of construction entities. It includes particle filtering (condensation), affine transformation, and eigen-image analysis, etc.

```
namespace ObjectTracking
{
   public partial class TemplateTracker : IDisposable
   {
      private String index;
      private Parameter param;
      private double cx, cy;
      private Matrix<double> wimgs;

      private Matrix<double> mean;
      private Matrix<double> eigval;
      private Matrix<double> basis;
      private int numsample;         /

      private long cur;
      private long start;
      private int batch;
      private bool _disposed = false;
      private double maxVal;

      private Matrix<double> corners;

      private String eqType;
      private String view;
      private int[] votes;
      public long SinceWhenNotDetected = 0;

      public TemplateTracker(string id, String etype, String view)
      {
         cur = 0;
         start = 0;
         batch = 0;
         numsample = 0;
```

```
    index = id;
    cx = 0;
    cy = 0;
    corners = new Matrix<double>(2, 5);
    eqType = etype;
    this.view = view;
    votes = new int[3];
}

public void Initiate(Image<Bgr, byte> im, Matrix<double> initParam, Option opt,
                     long timestamp, bool isNew)
{
    cx = initParam[0, 0];
    cy = initParam[1, 0];

    if (opt.x_or_dx == "dx/dy")
    {
        initParam[0, 0] = 0;
        initParam[1, 0] = 0;
    }

    param = new Parameter(initParam, opt);
    wimgs = new Matrix<double>(opt.tw * opt.th, opt.batchsize);

    param.Est = affparam2mat(param.Est);

    param.Wimg = warpimg(im, param.Est, cx, cy, param.TW, param.TH, param.C,
                                    opt);

    Image<Gray, double> item = new Image<Gray, double>(param.TW, param.TH);
    for (int i = 0; i < param.TW; i++)
    {
        for (int k = 0; k < param.TH; k++)
        {
            item.Data[param.TH - k - 1, param.TW - i - 1, 0] = param.Wimg.Data[i *
                        param.TH + k, 0] * 256.0;
        }
    }
    item.Save("wimg" + batch.ToString() + ".gif");

    mean = param.Wimg.Clone();

    double a = opt.tw / 2.0;
    double b = opt.th / 2.0;

    Matrix<double> _corners = new Matrix<double>(
```

```
        new double[3, 5] { { 1, 1, 1, 1, 1 }, { -a, a, a, -a, -a }, { -b, -b, b, b, -b } });

    Matrix<double> M = new Matrix<double>(new double[2, 3] {
                                { cx, param.Est.Data[2, 0], param.Est.Data[3, 0] },
                                { cy, param.Est.Data[4, 0], param.Est.Data[5, 0] } });

    CvInvoke.cvGEMM(M.Ptr, _corners.Ptr, 1.0, IntPtr.Zero, 0.0, corners.Ptr,
                               Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_DEFAULT);

    cur = timestamp;
    if (isNew) start = timestamp;
}

public void Track(Image<Bgr, byte> im, Option opt, long timestamp)
{
    if (opt.x_or_dx == "dx/dy")
    {
        if (cur == start)
        {
            opt.affsig[0] *= (10.0 * opt.n_th);
            opt.affsig[1] *= (10.0 * opt.n_th);
        }
    }

    maxVal = estwarp_condense(im, basis, mean, ref param, cx, cy, opt);

    if (opt.x_or_dx == "dx/dy")
    {
        cx += param.Est.Data[0, 0];
        cy += param.Est.Data[1, 0];
    }
    else
    {
        cx = param.Est.Data[0, 0];
        cy = param.Est.Data[1, 0];
    }

    for(int i = 0; i < wimgs.Rows; i++)
        wimgs.Data[i, batch] = param.Wimg.Data[i, 0];

    batch++;
    if (batch >= opt.batchsize)
    {
        sklm(wimgs, batch, ref basis, ref eigval, ref mean, ref numsample, opt.ff);
        wimgs.SetZero();
```

```
            if (basis.Cols > opt.maxbasis)
            {
                basis = GetSubRect(basis, 0, 0, basis.Rows, opt.maxbasis);
                eigval = GetSubRect(eigval, 0, 0, opt.maxbasis, 1);
            }
            batch = 0;
        }

        if (opt.x_or_dx == "dx/dy")
        {
            if (cur == start)
            {
                opt.affsig[0] /= (10.0 * opt.n_th);
                opt.affsig[1] /= (10.0 * opt.n_th);
            }
        }

        double a = opt.tw / 2.0;
        double b = opt.th / 2.0;

        Matrix<double> _corners = new Matrix<double>(
            new double[3, 5] { { 1, 1, 1, 1, 1 }, { -a, a, a, -a, -a }, { -b, -b, b, b, -b } });

        Matrix<double> M = new Matrix<double>(new double[2, 3] {
                                    { cx, param.Est.Data[2, 0], param.Est.Data[3, 0] },
                                    { cy, param.Est.Data[4, 0], param.Est.Data[5, 0] } });

        CvInvoke.cvGEMM(M.Ptr, _corners.Ptr, 1.0, IntPtr.Zero, 0.0, corners.Ptr,
                                Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_DEFAULT);

        cur = timestamp;
    }

    public void DrawResults(Image<Bgr, byte> im, Option opt, long timestamp,
                                Matrix<double> T)
    {
        DrawBox(im, param.Est, cx, cy, opt);

        if (opt.displayIndex)
        {
            MCvFont _font = new MCvFont(
                        Emgu.CV.CvEnum.FONT.CV_FONT_HERSHEY_PLAIN, 0.8, 1.0);
            _font.thickness = 2;
            im.Draw(index, ref _font, new Point((int)cx, (int)cy),
                        colorValue(opt.index_color));
        }
```

```
    Matrix<double> p = interestPointCoord(opt.interestPoint, corners, cx, cy);

    if (opt.displayPoint)
    {
        System.Drawing.Point centroid =
                new System.Drawing.Point((int)p.Data[0, 0], (int)p.Data[1, 0]);
        im.Draw(new Cross2DF(centroid, 3, 3), colorValue(opt.point_color), 1);
    }

    p = T * p;

    if (opt.saveCenter) WriteCentroid(opt, timestamp, p.Data[0, 0] / p.Data[2, 0],
                                                p.Data[1, 0] / p.Data[2, 0]);
}

public static Matrix<double> DrawBox(Image<Bgr, byte> im, Matrix<double> p,
                                                double cx, double cy, Option opt)
{
    LineSegment2D line = new LineSegment2D();
    Matrix<double> affCorners;

    if (p.Cols == 1)
    {
        double a = opt.tw / 2.0;
        double b = opt.th / 2.0;

        affCorners = new Matrix<double>(2, 5);
        Matrix<double> corners = new Matrix<double>(
            new double[3, 5] { { 1, 1, 1, 1, 1 }, { -a, a, a, -a, -a }, { -b, -b, b, b, -b } });

        Matrix<double> M = new Matrix<double>(new double[2, 3] {
                                            { cx, p.Data[2, 0], p.Data[3, 0] },
                                            { cy, p.Data[4, 0], p.Data[5, 0] } });

        CvInvoke.cvGEMM(M.Ptr, corners.Ptr, 1.0, IntPtr.Zero, 0.0, affCorners.Ptr,
                            Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_DEFAULT);
    }
    else
    {
        affCorners = p;
    }

    if (im != null)
    {
        for (int i = 0; i < 4; i++)
```

```
        {
            line.P1 = new Point((int)affCorners.Data[0,i],(int)affCorners.Data[1,i]);
            line.P2 = new Point((int)affCorners.Data[0,i+1],(int)affCorners.Data[1,i+1]);
            im.Draw(line, colorValue(opt.rect_color), opt.rect_thickness);
        }
    }

    return affCorners;
}

public static void DrawResults(Image<Bgr, byte> im, Matrix<double> p, double cx,
                               double cy, String name, String[] color, int thickness,
                               int tw, int th, bool showCentroid, bool showIndex,
                               String interestP)
{
    LineSegment2D line = new LineSegment2D();

    double a = tw / 2.0;
    double b = th / 2.0;

    Matrix<double> affCorners = new Matrix<double>(2, 5);
    Matrix<double> corners = new Matrix<double>(
        new double[3, 5] { { 1, 1, 1, 1, 1 }, { -a, a, a, -a, -a }, { -b, -b, b, b, -b } } );

    Matrix<double> M = new Matrix<double>(new double[2, 3] {
                { cx, p.Data[2, 0], p.Data[3, 0] }, { cy, p.Data[4, 0], p.Data[5, 0] } });

    CvInvoke.cvGEMM(M.Ptr, corners.Ptr, 1.0, IntPtr.Zero, 0.0, affCorners.Ptr,
                        Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_DEFAULT);

    for (int i = 0; i < 4; i++)
    {
        line.P1 = new Point((int)affCorners.Data[0, i], (int)affCorners.Data[1, i]);
        line.P2 = new Point((int)affCorners.Data[0, i+1], (int)affCorners.Data[1, i+1]);
        im.Draw(line, colorValue(color[0]), thickness);
    }

    Matrix<double> point = interestPointCoord(interestP, affCorners, cx, cy);

    if (showCentroid)
    {
        System.Drawing.Point centroid =
                new System.Drawing.Point((int)point.Data[0, 0], (int)point.Data[1, 0]);
        im.Draw(new Cross2DF(centroid, 3, 3), colorValue(color[1]), 1);
    }
```

```
    if (showIndex)
    {
        System.Drawing.Point centroid = new System.Drawing.Point((int)cx, (int)cy);
        MCvFont _font = new MCvFont(
                    Emgu.CV.CvEnum.FONT.CV_FONT_HERSHEY_PLAIN, 0.8, 1.0);
        _font.thickness = 1;
        im.Draw(name, ref _font, centroid, colorValue(color[2]));
    }

    return;
}

public void WriteCentroid(Option opt, long timestamp, double px, double py)
{
    System.IO.StreamWriter sw1;
    System.IO.StreamWriter sw2;

    if (cur == start)
    {
        sw1 = new System.IO.StreamWriter(opt.textFileName + index + ".txt", false,
                                         System.Text.Encoding.UTF8);
        sw1.WriteLine("tw: " + opt.tw.ToString());
        sw1.WriteLine("th: " + opt.th.ToString());
        sw2 = new System.IO.StreamWriter("rw_" + opt.textFileName + index + ".txt",
                                         false, System.Text.Encoding.UTF8);
    }
    else
    {
        sw1 = new System.IO.StreamWriter(opt.textFileName + index + ".txt", true,
                                         System.Text.Encoding.UTF8);
        sw2 = new System.IO.StreamWriter("rw_" + opt.textFileName + index + ".txt",
                                         true, System.Text.Encoding.UTF8);
    }

    sw1.WriteLine(timestamp.ToString() + " "
      + cx.ToString() + " " + cy.ToString() + " "
      + param.Est.Data[0, 0].ToString() + " " + param.Est.Data[1, 0].ToString() + " "
      + param.Est.Data[2, 0].ToString() + " " + param.Est.Data[3, 0].ToString() + " "
      + param.Est.Data[4, 0].ToString() + " " + param.Est.Data[5, 0].ToString() + " ");

    sw2.WriteLine(timestamp.ToString() + " " + (px).ToString() + " " +
                  py.ToString());

    sw1.Close();
    sw2.Close();
```

```
    return;
}

public bool Inside(double[] p, Option opt)
{
    int tw = opt.tw;
    int th = opt.th;

    double x = cx;
    double y = cy;
    Matrix<double> M = new Matrix<double>(new double[2, 2]{
                                {param.Est.Data[2, 0], param.Est.Data[3, 0]},
                                {param.Est.Data[4, 0], param.Est.Data[5, 0]} });

    Matrix<double> _p = new Matrix<double>(2, 1);
    Matrix<double> d = new Matrix<double>(new double[2, 1]{
                                                    {p[0] - x}, {p[1] - y} });

    CvInvoke.cvInvert(M.Ptr, M.Ptr
                            Emgu.CV.CvEnum.INVERT_METHOD.CV_LU);

    _p = M * d;
    double px = _p[0, 0] + opt.tw / 2;
    double py = _p[1, 0] + opt.th / 2;

    return (px >= 0 && px < opt.tw && py >= 0 && py < opt.th);
}

public void WriteCompleted(Option opt, long timestamp)
{
    System.IO.StreamWriter sw = new System.IO.StreamWriter(opt.textFileName +
                                        index + ".txt", true, System.Text.Encoding.UTF8);

    sw.WriteLine("completed_at " + timestamp.ToString());

    sw.Close();

    return;
}

public void UpdateVotes(String v)
{
    if (v == "r")
    {
        votes[0]++;
        votes[1]--;
```

```
            votes[2]--;
        }
        else if (v == "l")
        {
            votes[1]++;
            votes[0]--;
            votes[2]--;
        }
        else if (v == "b")
        {
            votes[2]++;
            votes[0]--;
            votes[1]--;
        }
        votes[0] = Math.Min(Math.Max(votes[0], 0), 3);
        votes[1] = Math.Min(Math.Max(votes[1], 0), 3);
        votes[2] = Math.Min(Math.Max(votes[2], 0), 3);

        view = "";
        if (votes.Max() == votes[0])
            view = "r";
        else if (votes.Max() == votes[1])
            view += "l";
        else if (votes.Max() == votes[2])
            view += "b";
    }

    public double estwarp_condense(Image<Bgr, byte> image, Matrix<double> basis,
                                   Matrix<double> mean, ref Parameter param,
                                   double cx, double cy, Option opt)
    {
        int n = param.N;
        int w = param.TW;
        int h = param.TH;
        int sz = w*h;

        if(param.Param == null)
        {
            param.Param = new Matrix<double>(6, n);
            CvInvoke.cvRepeat(affparam2geom(param.Est).Ptr, param.Param.Ptr);
        }
        else
        {
            Matrix<double> cumconf = new Matrix<double>(n, 1);
            cumconf.Data[0, 0] = param.Conf.Data[0, 0];
            for (int i = 1; i < n; i++)
```

```
        cumconf.Data[i, 0] = cumconf.Data[i - 1, 0] + param.Conf.Data[i, 0];

    Matrix<double> A = new Matrix<double>(1, n);
    A.SetRandUniform(new Emgu.CV.Structure.MCvScalar(0.0), new
                            Emgu.CV.Structure.MCvScalar(1.0));
    Matrix<int> idx = new Matrix<int>(n, 1);
    for (int i = 0; i < n; i++)
    {
        int id = 0;
        double temp = A.Data[0, i];
        for (int j = 0; j < n; j++)
        {
            if (temp > cumconf.Data[j, 0]) id++;
        }
        idx.Data[i, 0] = id;
    }
    cumconf.Dispose();
    A.Dispose();
    Matrix<double> par = new Matrix<double>(6, n);
    for (int j = 0; j < n; j++)
    {
        int id = idx.Data[j, 0];
        for (int i = 0; i < 6; i++)
            par.Data[i, j] = param.Param.Data[i, id];
    }
    param.Param.Data = par.Data;
    par.Dispose();
}

Matrix<double> ran = new Matrix<double>(6, n);
ran.SetRandNormal(new Emgu.CV.Structure.MCvScalar(0.0), new
                        Emgu.CV.Structure.MCvScalar(1.0));
for (int i = 0; i < 6; i++)
{
    double aff = opt.affsig[i];
    for (int j = 0; j < n; j++)
        param.Param.Data[i, j] += (ran.Data[i, j] * aff);
}
ran.Dispose();

Matrix<double> affmat = affparam2mat(param.Param);
Matrix<double> wimgs = warpimg(image, affmat, cx, cy, w, h, param.C, opt);

Matrix<double> diff = new Matrix<double>(wimgs.Size);
Matrix<double> t = new Matrix<double>(1, diff.Cols);
t.SetValue(1);
```

```
    CvInvoke.cvGEMM(mean.Ptr, t.Ptr, 1.0, wimgs.Ptr, -1.0, diff.Ptr,
                        Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_DEFAULT);
    t.Dispose();

    if (basis != null)
    {
       Matrix<double> coef = new Matrix<double>(basis.Cols, n);
      CvInvoke.cvGEMM(basis.Ptr, diff.Ptr, 1.0, IntPtr.Zero, 0.0, coef.Ptr,
                          Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_A_T);
       CvInvoke.cvGEMM(basis.Ptr, coef.Ptr, -1.0, diff.Ptr, 1.0, diff.Ptr,
                          Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_DEFAULT);
    }

    param.CalConf1(diff, opt.condensig);

    Point minidx, maxidx;
    double minVal, maxVal;
    param.Conf.MinMax(out minVal, out maxVal, out minidx, out maxidx);

    affmat.GetCol(maxidx.Y).CopyTo(param.Est);
    wimgs.GetCol(maxidx.Y).CopyTo(param.Wimg);

    return maxVal;
}

  public static Matrix<double> affparam2mat(Matrix<double> p)
{
    Matrix<double> q = new Matrix<double>(p.Size);

    for (int i = 0; i < p.Cols; i++)
    {
       double s = p.Data[2, i];
       double th = p.Data[3, i];
       double r = p.Data[4, i];
       double ph = p.Data[5, i];

       double cth = Math.Cos(th);
       double sth = Math.Sin(th);
       double cph = Math.Cos(ph);
       double sph = Math.Sin(ph);

       double ccc = cth * cph * cph;
       double ccs = cth * cph * sph;
       double css = cth * sph * sph;
       double scc = sth * cph * cph;
       double scs = sth * cph * sph;
```

```
      double sss = sth * sph * sph;

      q.Data[0, i] = p.Data[0, i];
      q.Data[1, i] = p.Data[1, i];
      q.Data[2, i] = s * (ccc + scs + r * (css - scs));
      q.Data[3, i] = s * (-ccs - sss + r * (ccs - scc));
      q.Data[4, i] = s * (scc + -ccs + r * (ccs + sss));
      q.Data[5, i] = s * (-scs + css + r * (ccc + scs));
   }

   return q;
}

public static Matrix<double> affparam2geom(Matrix<double> p)
{
   Matrix<double> q = new Matrix<double>(6, 1);

   Matrix<double> A = new Matrix<double>(new double[2, 2] {
                              { p[2, 0], p[3, 0] }, { p[4, 0], p[5, 0] } });
   Matrix<double> U = new Matrix<double>(2, 2);
   Matrix<double> S = new Matrix<double>(2, 2);
   Matrix<double> V = new Matrix<double>(2, 2);

   CvInvoke.cvSVD(A.Ptr, S.Ptr, U.Ptr, V.Ptr,
                        Emgu.CV.CvEnum.SVD_TYPE.CV_SVD_DEFAULT);

   if(U.Det < 0)
   {
      Matrix<double> Temp = U.Clone();
      U.Data[0, 0] = Temp.Data[0, 1];
      U.Data[1, 0] = Temp.Data[1, 1];
      U.Data[0, 1] = Temp.Data[0, 0];
      U.Data[1, 1] = Temp.Data[1, 0];

      Temp = V.Clone();
      V.Data[0, 0] = Temp.Data[0, 1];
      V.Data[1, 0] = Temp.Data[1, 1];
      V.Data[0, 1] = Temp.Data[0, 0];
      V.Data[1, 1] = Temp.Data[1, 0];

      double temp = S.Data[0, 0];
      S.Data[0, 0] = S.Data[1, 1];
      S.Data[1, 1] = temp;
   }

   q.Data[0, 0] = p.Data[0, 0];
```

```
    q.Data[1, 0] = p.Data[1, 0];
    q.Data[3, 0] = Math.Atan2(U.Data[1, 0]*V.Data[0, 0]+U.Data[1, 1]*V.Data[0, 1],
                              U.Data[0, 0] * V.Data[0, 0] + U.Data[0, 1] * V.Data[0, 1]);

    double phi = Math.Atan2(V[0, 1], V[0, 0]);
    double c, s;
    Matrix<double> R;

    if (phi <= -PI / 2)
    {
        c = Math.Cos(-PI / 2);  s = Math.Sin(-PI / 2);
        R = new Matrix<double>(new double[2, 2] { { c, -s }, { s, c } });
        V = V.Mul(R);
        S = S.Mul(R);
        CvInvoke.cvGEMM(R.Ptr, S.Ptr, 1.0, IntPtr.Zero, 0.0, S.Ptr,
                                    Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_A_T);
    }
    else if(phi >= PI / 2)
    {
        c = Math.Cos(PI / 2);  s = Math.Sin(PI / 2);
        R = new Matrix<double>(new double[2, 2] { { c, -s }, { s, c } });
        V = V.Mul(R);
        S = S.Mul(R);
        CvInvoke.cvGEMM(R.Ptr, S.Ptr, 1.0, IntPtr.Zero, 0.0, S.Ptr,
                                   Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_A_T);
    }

    q.Data[2, 0] = S.Data[0, 0];
    q.Data[4, 0] = S.Data[1, 1] / S.Data[0, 0];
    q.Data[5, 0] = Math.Atan2(V.Data[0, 1], V.Data[0, 0]);

    return q;
}

unsafe public Matrix<double> warpimg(Image<Bgr, byte> img, Matrix<double> p,
                                     double cx, double cy,
                                     int tw, int th, String color, Option opt)
{
    int l = tw * th;
    int nsample = p.Cols;

    double[,] _p = p.Data;
    double[] x = new double[l];
    double[] y = new double[l];
    Matrix<double> mean = new Matrix<double>(l, nsample);
    Image<Gray, double> temp = new Image<Gray, double>(img.Size);
```

```
if (color == "Gray")
{
   temp = img.Convert<Gray, byte>().ConvertScale<double>(1.0 / 256.0, 0.0);
}
else if (color == "Blue" || color == "Green" || color == "Red")
{
   Image<Bgr, double> image = img.ConvertScale<double>(1.0 / 256.0, 0.0);
   if (color == "Blue")
      temp = image[0];
   else if (opt.cspace_t == "Green")
      temp = image[1];
   else if (opt.cspace_t == "Red")
      temp = image[2];
}
else
{
   Image<Hsv, double> image = (img.Convert<Hsv, byte>() as Image<Hsv,
                                        byte>).ConvertScale<double>(1.0 / 256.0, 0.0);
   if (opt.cspace_t == "Hue")
      temp = image[0];
   else if (opt.cspace_t == "Saturation")
      temp = image[1];
   else if (opt.cspace_t == "Value")
      temp = image[2];
}

int height = temp.Height;
int width = temp.Width;

for (int isample = 0; isample < nsample; isample++)
 {
    double p0 = _p[0, isample] + (opt.x_or_dx == "dx/dy" ? cx : 0);
    double p1 = _p[1, isample] + (opt.x_or_dx == "dx/dy" ? cy : 0);
    double p2 = _p[2, isample];
    double p3 = _p[3, isample];
    double p4 = _p[4, isample];
    double p5 = _p[5, isample];

    for (int i = 0; i < tw; i++)
    {
       int ix = i - tw / 2 + 1;
       for (int j = 0; j < th; j++)
       {
          x[i * th + j] = p0 + ix * p2 + (j - th / 2 + 1) * p3;
          y[i * th + j] = p1 + ix * p4 + (j - th / 2 + 1) * p5;
```

```
            }
        }

        int x0, x1, y0, y1;
        double xx, yy, rx, ry;
        double a, b, c, d;
        bool x00, x01, x10, x11;
        bool y00, y01, y10, y11;

        for (int i = 0; i < l; i++)
        {
            xx = x[i];
            yy = y[i];

            x0 = (int)xx; x1 = x0 + 1;
            y0 = (int)yy; y1 = y0 + 1;

            rx = xx - x0;
            ry = yy - y0;

            y00 = (y0 <= 0);
            y01 = (y0 > height);
            y10 = (y1 <= 0);
            y11 = (y1 > height);
            x00 = (x0 <= 0);
            x01 = (x0 > width);
            x10 = (x1 <= 0);
            x11 = (x1 > width);

            a = (y00 || y01 || x00 || x01) ? 0 : temp [(y0 - 1) * width + (x0 - 1)];
            b = (y00 || y01 || x10 || x11) ? 0 : temp [(y0 - 1) * width + (x1 - 1)];
            c = (y10 || y11 || x00 || x01) ? 0 : temp [(y1 - 1) * width + (x0 - 1)];
            d = (y10 || y11 || x10 || x11) ? 0 : temp [(y1 - 1) * width + (x1 - 1)];

            mean.Data[i, isample] = ((1-rx)*a+rx*b)*(1-ry)+((1-rx)*c+rx*d)*ry;
        }
    }
    return mean;
}

public void sklm(Matrix<double> data, int bs, ref Matrix<double> U, ref
                 Matrix<double> D, ref Matrix<double> mu, ref int n0, double ff)
{
    int N = data.Rows;
    int n = data.Cols;
```

```
if (U == null)
{
   if (bs == 1)
   {
      data.GetCol(0).CopyTo(mu);
      U = new Matrix<double>(N, n);
      D = new Matrix<double>(1, 1);
   }
   else
   {
      CvInvoke.cvReduce(data.Ptr, mu.Ptr,
                  Emgu.CV.CvEnum.REDUCE_DIMENSION.SINGLE_COL,
                  Emgu.CV.CvEnum.REDUCE_TYPE.CV_REDUCE_AVG);

      Matrix<double> _mu = new Matrix<double>(data.Size);
      CvInvoke.cvRepeat(mu.Ptr, _mu.Ptr);
      CvInvoke.cvSub(data.Ptr, _mu.Ptr, data.Ptr, IntPtr.Zero);

      int dimension = Math.Min(data.Rows, data.Cols);
      U = new Matrix<double>(data.Rows, dimension);
      Matrix<double> S = new Matrix<double>(dimension, dimension);
      CvInvoke.cvSVD(data.Ptr, S.Ptr, U.Ptr, IntPtr.Zero,
                        Emgu.CV.CvEnum.SVD_TYPE.CV_SVD_DEFAULT);

      D = new Matrix<double>(dimension, 1);
      Matrix<double> diag = new Matrix<double>(dimension, 1);
      CvInvoke.cvGetDiag(S.Ptr, diag.Ptr, 0);
      CvInvoke.cvCopy(diag.Ptr, D.Ptr, IntPtr.Zero);
   }
}
else
{
   if(mu != null)
   {
      Matrix<double> mu1 = new Matrix<double>(data.Rows, 1);
      CvInvoke.cvReduce(data.Ptr, mu1.Ptr,
                        Emgu.CV.CvEnum.REDUCE_DIMENSION.SINGLE_COL,
                        Emgu.CV.CvEnum.REDUCE_TYPE.CV_REDUCE_AVG);

      Matrix<double> t = new Matrix<double>(1, data.Cols);
      t.SetValue(1);
      CvInvoke.cvGEMM(mu1.Ptr, t.Ptr, -1.0, data.Ptr, 1.0, data.Ptr,
                        Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_DEFAULT);
      t.Dispose();

      double temp = Math.Sqrt(n * n0 / (double)(n + n0));
```

```
    Matrix<double> _mu = new Matrix<double>(mu.Size);
    CvInvoke.cvSub(mu.Ptr, mu1.Ptr, _mu.Ptr, IntPtr.Zero);
    data = data.ConcateHorizontal(temp * _mu);
    mu1 = (ff * n0 * mu + n * mu1) / (n + ff * n0);
    n = (int)(n + ff * n0);
    _mu.Dispose();
}

Matrix<double> data_proj = new Matrix<double>(U.Cols, data.Cols);
Matrix<double> data_res = new Matrix<double>(data.Size);
CvInvoke.cvGEMM(U.Ptr, data.Ptr, 1.0, IntPtr.Zero, 0.0, data_proj.Ptr,
                        Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_A_T);
CvInvoke.cvGEMM(U.Ptr, data_proj.Ptr, -1.0, data.Ptr, 1.0, data_res.Ptr,
                    Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_DEFAULT);

int ndata = data.Cols;
int nbasis = D.Rows;

dnAnalytics.LinearAlgebra.Decomposition.GramSchmidt qr1
    = new dnAnalytics.LinearAlgebra.Decomposition.GramSchmidt(
                                            new DenseMatrix(data_res.Data));
Matrix<double> q = new Matrix<double>(qr1.Q().ToArray());
Matrix<double> Q = U.ConcateHorizontal(q);
Matrix<double> DD = new Matrix<double>(nbasis, nbasis);

for(int i = 0; i < nbasis; i++)
    DD.Data[i, i] = D.Data[i, 0];

Matrix<double> R = new Matrix<double>(q.Cols, data_res.Cols);
CvInvoke.cvGEMM(q.Ptr, data_res.Ptr, 1.0, IntPtr.Zero, 0.0, R.Ptr,
                        Emgu.CV.CvEnum.GEMM_TYPE.CV_GEMM_A_T);
R = data_proj.ConcateVertical(R);
R = (ff * DD).ConcateVertical(new Matrix<double>(
                                            ndata, nbasis)).ConcateHorizontal(R);

Matrix<double> S = new Matrix<double>(R.Size);
U = new Matrix<double>(R.Size);
CvInvoke.cvSVD(R.Ptr, S.Ptr, U.Ptr, IntPtr.Zero,
                    Emgu.CV.CvEnum.SVD_TYPE.CV_SVD_DEFAULT);

D = new Matrix<double>(S.Rows, 1);
Matrix<double> diag = new Matrix<double>(S.Rows, 1);
CvInvoke.cvGetDiag(S.Ptr, diag.Ptr, 0);
CvInvoke.cvCopy(diag.Ptr, D.Ptr, IntPtr.Zero);

Matrix<double> Temp = D.Clone();
```

```
            Temp._Mul(Temp);

            double cutoff = Temp.Sum * 1e-6;
            int index = D.Rows - 1;
            for (int i = D.Rows - 1; i >= 0; i--)
            {
               if (D.Data[i, 0] > cutoff) { index = i; break; }
            }
            D = GetSubRect(D, 0, 0, index + 1, 1);
            U = GetSubRect(U, 0, 0, U.Rows, index + 1);
            U = Q * U;
         }
      }
   }

}
```

APPENDIX C

CODE FOR 3D COORDINATE CALCULATION

This appendix presents the main part in the prototype code for stereo camera calibration.

```
public class Processing
{
  private MainWindow main;
  private Picture stream;
  private String test_name;
  private String test_acronym;

  public Processing(MainWindow main, Picture stream)
  {
    this.main = main;
    if (stream is Camera)
    {
      this.stream = new Camera(main, "Output");
      this.stream.Set_isOutput(true);
      this.stream.Show(true);
      ((Camera)(this.stream)).Play();
    }
    else
    {
      this.stream = new Picture(main, "Output.jpg");
      this.stream.Set_isOutput(true);
      this.stream.Show(true);
    }

    String[] str = main.Get_streamList()[0].Get_name().Split(new char[]{'_'});
    test_acronym = str[0].ElementAt(0).ToString() + str[1].ElementAt(0).ToString();
    test_name = str[0] + "_" + str[1];
  }

  public void Process_Frame(Frame frame)
  {
    Image<Bgr, byte> img1 = (Image<Bgr, byte>)frame.Get_frame();
    Image<Bgr, byte> img2 = (Image<Bgr,byte>)main.Get_streamList_stream(1)
                                          .Get_Buffer().ElementAt(0).Get_frame();

    List<double[]> inlier_left = new List<double[]>();
    List<double[]> inlier_right = new List<double[]>();
```

```
List<double[]> track_left1 = new List<double[]>();
List<double[]> track_left2 = new List<double[]>();
List<double[]> track_left3 = new List<double[]>();

List<double[]> track_right1 = new List<double[]>();
List<double[]> track_right2 = new List<double[]>();
List<double[]> track_right3 = new List<double[]>();

MatlabMatrixReader dmr;
Matrix matrix;

int method = 1;
double distratio = 0.8;

if (method == 1)
{
   Image<Gray, byte> gray1 = img1.Convert<Gray, byte>();
   Image<Gray, byte> gray2 = img2.Convert<Gray, byte>();

   MCvSURFParams surfParam = new MCvSURFParams(500, false);
   SURFFeature[] features1 = gray1.ExtractSURF(ref surfParam);
   SURFFeature[] features2 = gray2.ExtractSURF(ref surfParam);

   SURFTracker tracker = new SURFTracker(features1);
   SURFTracker.MatchedSURFFeature[] matchedFeatures =
                                        tracker.MatchFeature(features2, 2, 20);
   matchedFeatures = SURFTracker.VoteForUniqueness(matchedFeatures,
                                                        distratio);
   matchedFeatures = SURFTracker.VoteForSizeAndOrientation(
                                                    matchedFeatures, 1.5, 20);

   int n_matches = matchedFeatures.Length;
   Matrix<double> p1 = new Matrix<double>(n_matches, 2);
   Matrix<double> p2 = new Matrix<double>(n_matches, 2);

   for (int j = 0; j < n_matches; j++)
   {
      p1.Data[j, 0] = matchedFeatures[j].SimilarFeatures[0].Feature.Point.pt.X;
      p1.Data[j, 1] = matchedFeatures[j].SimilarFeatures[0].Feature.Point.pt.Y;

      p2.Data[j, 0] = matchedFeatures[j].ObservedFeature.Point.pt.X;
      p2.Data[j, 1] = matchedFeatures[j].ObservedFeature.Point.pt.Y;
   }

   Matrix<double> F = new Matrix<double>(3, 3);
```

```
    Matrix<sbyte> status = new Matrix<sbyte>(1, n_matches);
    CvInvoke.cvFindFundamentalMat(p1.Ptr, p2.Ptr, F.Ptr,
                                  Emgu.CV.CvEnum.CV_FM.CV_FM_RANSAC_ONLY,
                                  2.0, 0.99, status.Ptr);

    for (int i = 0; i < status.Cols; i++)
    {
       if (status[0, i] == 1)
       {
          inlier_left.Add(new double[] { p1.Data[i, 0], p1.Data[i, 1] });
          inlier_right.Add(new double[] { p2.Data[i, 0], p2.Data[i, 1] });
       }
    }
}
else
{
    dmr = new MatlabMatrixReader(test_name + "_inlier.mat");
    matrix = dmr.ReadMatrix(StorageType.Dense);
    Matrix<double> inlier_pl = new Matrix<double>(matrix.Rows, 2);
    Matrix<double> inlier_pr = new Matrix<double>(matrix.Rows, 2);
    for (int i = 0; i < matrix.Rows; i++)
    {
       inlier_pl.Data[i, 0] = matrix[i, 0];
       inlier_pl.Data[i, 1] = matrix[i, 1];
       inlier_pr.Data[i, 0] = matrix[i, 2];
       inlier_pr.Data[i, 1] = matrix[i, 3];
    }
    for (int i = 0; i < inlier_pl.Rows; i++)
    {
       inlier_left.Add(new double[2] { inlier_pl.Data[i, 0], inlier_pl.Data[i, 1] });
       inlier_right.Add(new double[2] { inlier_pr.Data[i, 0], inlier_pr.Data[i, 1] });
    }
    #endregion
}

String methods;
if (method == 1)
    methods = distratio.ToString();
else
    methods = "siftmapsac";

#region draw lines between the matched features
Image<Bgr, Byte> res = img1.ConcateVertical(img2);
PointF p = new PointF();
for (int i = 0; i < inlier_left.Count; i++)
{
```

```
    p.X = (float)inlier_right[i][0];
    p.Y = (float)inlier_right[i][1];
    p.Y += img1.Height;
    res.Draw(new LineSegment2DF(new PointF((float)inlier_left[i][0],
                                    (float)inlier_left[i][1]), p), new Bgr(0, 0, 0), 2);
}
res.Save("matching_" + methods +".jpg");

dmr = new MatlabMatrixReader(test_acronym + "l1.mat");
matrix = dmr.ReadMatrix(StorageType.Dense);
for (int i = 0; i < matrix.Rows; i++)
{
    track_left1.Add(new double[2] { matrix[i, 1], matrix[i, 2] });
}
dmr = new MatlabMatrixReader(test_acronym + "r1.mat");
matrix = dmr.ReadMatrix(StorageType.Dense);
for (int i = 0; i < matrix.Rows; i++)
{
    track_right1.Add(new double[2] { matrix[i, 1], matrix[i, 2] });
}

Main_Algorithms_CSharp.InternalParameters calib1 = new
                                    Main_Algorithms_CSharp.InternalParameters();
dmr = new MatlabMatrixReader("cc_left.mat");
matrix = dmr.ReadMatrix(StorageType.Dense);
calib1.cc = new double[2] { matrix[0, 0], matrix[1, 0] };
dmr = new MatlabMatrixReader("fc_left.mat");
matrix = dmr.ReadMatrix(StorageType.Dense);
calib1.fc = new double[2] { matrix[0, 0], matrix[1, 0] };
dmr = new MatlabMatrixReader("kc_left.mat");
matrix = dmr.ReadMatrix(StorageType.Dense);
calib1.kc = new double[5] { matrix[0, 0], matrix[1, 0], matrix[2, 0], matrix[3, 0],
                                    matrix[4, 0] };

Main_Algorithms_CSharp.InternalParameters calib2 = new
                                    Main_Algorithms_CSharp.InternalParameters();
dmr = new MatlabMatrixReader("cc_right.mat");
matrix = dmr.ReadMatrix(StorageType.Dense);
calib2.cc = new double[2] { matrix[0, 0], matrix[1, 0] };
dmr = new MatlabMatrixReader("fc_right.mat");
matrix = dmr.ReadMatrix(StorageType.Dense);
calib2.fc = new double[2] { matrix[0, 0], matrix[1, 0] };
dmr = new MatlabMatrixReader("kc_right.mat");
matrix = dmr.ReadMatrix(StorageType.Dense);
calib2.kc = new double[5] { matrix[0, 0], matrix[1, 0], matrix[2, 0], matrix[3, 0],
                                    matrix[4, 0] };
```

```
track_left1.Clear();
track_right1.Clear();
track_left1.Add(new double[2] { 775, 602 });
track_right1.Add(new double[2] { 736, 562 });

#region Obtaining Extrinsic Parameteres (R and t) and Triangulation (Abbas')
Main_Algorithms_CSharp.ReconstructionData extrinsic
        = Main_Algorithms_CSharp.Reconstruct(inlier_left, inlier_right,
          track_left1, track_right1, calib1, calib2, "output1_" + methods +".txt");

Matrix<double> xL1 = new Matrix<double>(2, track_left1.Count);
Matrix<double> xR1 = new Matrix<double>(2, track_right1.Count);
for(int i = 0; i < track_left1.Count; i++)
{
   xL1[0, i] = track_left1[i][0];
   xL1[1, i] = track_left1[i][1];
   xR1[0, i] = track_right1[i][0];
   xR1[1, i] = track_right1[i][1];
}

Matrix<double> R = new Matrix<double>(3, 3);
Matrix<double> t = new Matrix<double>(3, 1);
for (int i = 0; i < 3; i++)
{
   for (int j = 0; j < 3; j++)
   {
      R.Data[i, j] = extrinsic.RpT[i][j];
   }
   t.Data[i, 0] = extrinsic.tpT[0][i];
}

Matrix<double> fc_left = new Matrix<double>(2, 1);
Matrix<double> cc_left = new Matrix<double>(2, 1);
Matrix<double> kc_left = new Matrix<double>(5, 1);
fc_left.Data[0, 0] = calib1.fc[0];
fc_left.Data[1, 0] = calib1.fc[1];
cc_left.Data[0, 0] = calib1.cc[0];
cc_left.Data[1, 0] = calib1.cc[1];
kc_left.Data[0, 0] = calib1.kc[0];
kc_left.Data[1, 0] = calib1.kc[1];
kc_left.Data[2, 0] = calib1.kc[2];
kc_left.Data[3, 0] = calib1.kc[3];
kc_left.Data[4, 0] = calib1.kc[4];

Matrix<double> fc_right = new Matrix<double>(2, 1);
Matrix<double> cc_right = new Matrix<double>(2, 1);
```

```
        Matrix<double> kc_right = new Matrix<double>(5, 1);
        fc_right.Data[0, 0] = calib2.fc[0];
        fc_right.Data[1, 0] = calib2.fc[1];
        cc_right.Data[0, 0] = calib2.cc[0];
        cc_right.Data[1, 0] = calib2.cc[1];
        kc_right.Data[0, 0] = calib2.kc[0];
        kc_right.Data[1, 0] = calib2.kc[1];
        kc_right.Data[2, 0] = calib2.kc[2];
        kc_right.Data[3, 0] = calib2.kc[3];
        kc_right.Data[4, 0] = calib2.kc[4];

        Matrix<double> pl1 = StereoTriangulation.Triangulate(xL1, xR1, R, t, fc_left,
                        cc_left, kc_left, 0.0,
                        fc_right, cc_right, kc_right, 0.0);

        System.IO.StreamWriter sw = new System.IO.StreamWriter(methods + ".txt");
        for (int i = 0; i < pl1.Cols; i++)
        {
           sw.WriteLine(pl1.Data[0, i].ToString() + " " + pl1.Data[1, i].ToString() + " " +
                           pl1.Data[2, i].ToString());
        }
        sw.Close();

        frame.Set_frame(res);
        if(stream is Camera)
           stream.Add_Frame_To_Buffer(frame);
        else
        {
           stream.Add_Frame_To_Buffer(frame);
           stream.Show(false);
        }
     }
  }
```

REFERENCES

ANSI/ISEA (2010). "ANSI/ISEA 107-2010: American national standard for high-visibility safety apparel and headwear." International Safety Equipment Association (ISEA).

Anumba, C. J. (1998) "Industry Uptake of Construction IT Innovations – Key Elements of a Proactive Strategy." *The Life-cycle of Construction IT Innovations: Technology Transfer from Research to Practice*, Bjork B-C & Jagbeck A. (Eds), CIB Working Commission W78 Conference, KTH Stockholm, 3-5 June, 77-83.

Arnaud, E. and Memin, E. (2005). "An efficient Rao-Blackwellized particle filter for object tracking." *Proceedings of ICIP 2005, IEEE,* Genoa, Italy, 2, 426-429.

Bauer, J., Sünderhauf, N., Protzel, P. (2007). "Comparing several implementations of two recently published feature detectors." *Proceedings of Int. Conf. on Intelligent and Autonomous Systems*, Toulouse, France.

Bay, H., Tuytelaars, T. and Gool, L.V. (2008). "SURF: Speeded Up Robust Features." Computer Vision and Image Understanding, 110(3), 346-359.

Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection." *IEEE T Pattern Anal*, 19(7), 711-720.

Bernold L. E. (2007). "Control schemes for tele-robotic pipe installation." *Automation in Construction*, Elsevier, 16(4), 518-524.

Bock, T. (2008). "Digital design and robotic production 3-D shaped precast components." *Proceedings of The 25th International Symposium on Automation in Construction*, Zavadskas, E.K., Kaklauskas, A., Skibniewski M.J. (Eds.), Technika, Vilnius, 11–21.

Bohn, G. and Teizer, J. (2009). "Benefits and barriers of monitoring construction activities using hi-resolution automated cameras." *Proceedings of Construction Research Congress*, ASCE, 21-30.

Bouguet, J.Y. (2004). "Camera calibration toolbox for Matlab." Intel Corp., <http://www.vision.caltech.edu/bouguetj/calib_doc> (Accessed June 4, 2012)

Bradski, G., and Kaehler, A., (2008). “Learning OpenCV: Computer Vision with the OpenCV.” ISBN-13: 978-0596516130

Brilakis, I. K., and Soibelman, L.(2008). “Shape-based retrieval of construction site photographs.” *J Comput Civil Eng*, 22(1), 14-20.

Bruckner, M, Bajramovic, F., and Denzler, J. (2008). “Experimental evaluation of relative pose estimation algorithms.” *Proceedings of the 3rd International Conference on Computer Vision Theory and Applications*, 2, 431-438.

Cable, L. (2010) “Construction equipment monitoring system.” *Government Product News*, Penton Media, December 2010. <http://govpro.com/products/fleets/management/construction-equipment-monitoring-201012/> (Accessed June 4, 2012)

Caldas, C. H., Torrent, D. G., and Haas, C. T. (2004) "Integration of automated data collection technologies for real-time field materials management." *Proceedings of the 21st International Symposium on Automation and Robotics in Construction*, IAARC.

Chae, S. and Kano, N. (2007). “Application of location information by stereo camera images to project progress monitoring.” *Proceedings of the 24th International Symposium on Automation and Robotics in Construction*, Kochi, Kerala, India, 89-92.

Chae, S., and Yoshida, T. (2010). "Application of RFID technology to prevention of collision accident with heavy equipment." *Automat Constr*, 19(3), 368-374.

Chi, S. and Caldas, C.H. (2011). “Automated object identification using optical video cameras on construction sites.” *Computer-Aided Civil and Infrastructure Engineering*, 26(5), 368-380.

Comaniciu, D., Ramesh, B., and Meer, P. (2003). “Kernel-based object tracking.” *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5), 564-577.

Construction Equipment (2010) “DPL America, Hyundai selects DPL’s asset monitoring solution.” Construction Equipment, December 2010. <http://www.constructionequipment.com/hyundai-selects-dpl%E2%80%99s-asset-monitoring-solution> (Accessed June 4, 2012)

Cover, T. M., and Hart, P. E. (1967). "Nearest Neighbor Pattern Classification." *IEEE T Inform Theory*, 13(1), 21-27.

Cox, I.J. and Hingorani, S.L. (1996). "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking." *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(2), 138-150.

Dalal, N. and Triggs, B. (2005). "Histograms of oriented gradients for human detection." *Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 886-893.

Eecke, R. (2010) "Greening (and saving) with GPS fleet tracking." *Construction News Tracker*, Cygnus Business Media. <http://www.forconstructionpros.com/article/10288916/greening-and-saving-with-gps-fleet-tracking> (Accessed June 4, 2012)

Elghamrawy, T., and Boukamp, F. (2010). "Managing construction information using RFID-based semantic contexts." *Automat Constr*, 19(8), 1056-1066.Engineering News-Record (2008a) "Equipment tracking: wireless monitoring." The MacGraw-Hill Companies, December 2008. <http://enr.construction.com/products/product_snapshot/2008/1203-productsnapshot.asp> (Accessed June 4, 2012)

Emgu CV, (2010), "Emgu CV: OpenCV in .NET (C#, VB, C++ and More)." <http://www.emgu.com/wiki/index.php/Main_Page> (Accessed June 4, 2012)

Engineering News-Record (2008a) "Equipment tracking: wireless monitoring." The MacGraw-Hill Companies, December 2008 <http://enr.construction.com/products/product_snapshot/2008/1203-productsnapshot.asp> (Accessed June 4, 2012)

Engineering News-Record (2008b) "Equipment monitor: wireless locator." The MacGraw-Hill Companies, July 2008. <http://enr.construction.com/products/newProducts/archives/080723.asp> (Accessed June 4, 2012)

Engineering News-Record (2008c) "Behind a murky name lurks an amazing little tag." The MacGraw-Hill Companies, April 2008. <http://enr.construction.com/opinions/editorials/archives/080423.asp> (Accessed June 4, 2012)

Engineering News-Record (2008d). “Solor-powered webcam: remotely monitor job sites.” The MacGraw-Hill Companies <http://enr.construction.com/products/newProducts/archives/080730.asp> (Accessed June 4, 2012)

Engineering News-Record (2008e). “Jobsite monitoring: remote camera keeps an eye on projects.” The MacGraw-Hill Companies <http://enr.construction.com/equipment/features/archives/080301-26-2.asp> (Accessed June 4, 2012)

Ergen, E., Akinci, B., and Sacks, R. (2007). "Tracking and locating components in a precast storage yard utilizing radio frequency identification technology and GPS." *Automat Constr*, 16(3), 354-367.

Fathi, H. and Brilakis, I. (2011). “Automated sparse 3D point cloud generation of infrastructure using its distinctive visual features.” *Advanced Engineering Informatics*, 25(4), 760-770.

Fontana, R. J., Richley, E., and Barney, J. A. (2003). "Commercialization of an Ultra Wideband Precision Asset Location system." *2003 IEEE Conference on Ultra Wideband Systems and Technologies, Conference Proceedings*, 369-373.

Fontana, R. J. (2004). "Recent system applications of short-pulse ultra-wideband (UWB) technology." *IEEE T Microw Theory*, 52(9), 2087-2104.

ForConstructionPros.Com (2006). “GPS Tracking and fleet management software system pays big dividends.” Cygnus Business Media, November 2006. <http://www.forconstructionpros.com/article/10299515/gps-tracking-and-fleet-management-software-system-pays-big-dividends> (Accessed June 4, 2012)

ForConstructionPros.Com (2011). “Study shows interference with GPS poses $96 billion threat to US economy.” Cygnus Business Media, June 2011. <http://www.forconstructionpros.com/press_release/10366178/study-shows-interference-with-gps-poses-96-billion-threat-to-us-economy> (Accessed June 4, 2012)

Freedman, D., and Zhang, T. (2004). "Active contours for tracking distributions." *Ieee T Image Process*, 13(4), 518-526.

Freund, Y. and Schapire, R.E. (1997). "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of Computer and System Sciences,* 55(1), 119-139.

Fuchs, S. (2010), "Multipaths interference compensation in time-of-flight camera image." *Proceedings of the 20th Int. Conf.on Pattern Recognition*, Istanbul, 3583-3586.

Gächter, S., Nguyen, V., Siegwart, R. (2006). "Results on range image segmentation for service robots", *Proceedings of IEEE International Conference on Computer Vision Systems*, Lausanne, Switzerland.

Golparvar-Fard, M., Peña-Mora, and Savarese, S. (2010). "D4AR- A 4-Dimensional augmented reality model for automating construction progress data collection, processing and communication." *Journal of Information Technology in Construction (ITcon)*, 14, 129-153.

Gomez, K. (2007a) "Cost-effective anti-collision." Construction Contractor, June 2007, 32.

Gomez, K. (2007b) "Fall protection enters the information age." Construction Contractor, June 2007, 40.

Gomez, K. (2008). "Mobile security + remote monitoring." Construction Contractor, November 2008.

Gong, J., and Caldas, C. H. (2008). "Data processing for real-time construction site spatial modeling." *Automat Constr*, 17(5), 526-535.

Gong, J., and Caldas, C. H. (2010). "Computer Vision-Based Video Interpretation Model for Automated Productivity Analysis of Construction Operations." *J Comput Civil Eng*, 24(3), 252-263.

Gruen, A. (1997). "Fundamentals of videogrammetry – A review", Human Movement Science Journal, 16, 155-187.

Haker, S., Tannebaum, A., Sapiro, G., and Washburn, D. (2001), "Missile tracking using knowledge-based adaptive thresholding: Tracking of high speed projectiles." *Proceedings of ICIP*, 786-789.

Hartley, R. (1997). “In defense of the eight-point algorithm.” IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(6), 580-593.

Hartley, R. and Sturm, P. (1997). “Triangulation.” Journal of Computer Vision and Image Understanding, 68(2), 146-157.

Hartley, R. and Zisserman, A. (2004). “Multiple view geometry in computer vision.” Cambridge University Press.

Heikkilä, J. and Silvén, O. (1997). “A four-step camera calibration procedure with implicit image correction.” Proc., IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 1106-1112.

Henderson, H. (2008). “Benefits of machine control.” *Construction Contractor*, December 2008.

Huang, C.B., Yu, S.S., Zhou, J.L., and Lu, H.W. (2004). “Image retrieval using both color and local spatial feature histograms.” Proceedings of 2004 International Conference on Communication, Circuits, and Systems, 2, 927-931.

Isard, M. and Blake, A. (1998). “CONDENSATION-conditional density propagation for visual tracking.” *International Journal on Computer Vision*, 29(1), 5-28.

Joachims, T. (1999). “Making large-scale SVM learning practical.” *Advances in Kernel Methods – Support Vector Learning*, MIT Press, Cambridge, MA, USA, 169-184.

Jog, G., Park, M.W., and Brilakis, I. (2011). “Truck face recognition using semantic texton forests.” *Proceedings of the ASCE Construction Research Congress*, Ontario, Canada.

Kanatani, K., Sugaya, Y., Niitsuma, H. (2008). “Triangulation from two views revisited: Hartley-Sturm vs. optimal correction.” *Proceedings of the 19th British Machine Vision Conference*, Leeds, UK, 173-182.

Khan, Z., Balch, T., and Dellaert, F. (2004). “A Rao-Blackwellized particle filter for eigentracking.” *Proceedings of CVPR 2004, IEEE*, Washington DC, 2, 980-986.

Ko, C. H. (2009). “RFID-based building maintenance system.” *Automation in Construction*, 18(3), Elsevier, 275-284.

Ko, C. H. (2010). "RFID 3D location sensing algorithms." *Automat Constr*, 19(5), 588-595.

Li, L.Y., Huang, W.M., Gu, I.Y.H., and Tian, Q. (2002). “Foreground object detection in changing background based on color co-occurrence statistics.” *Proceedings of the 6th IEEE Workshop on Applications of Computer Vision*, 269-274.

Lienhart, R. and Maydt, J. (2002). “An extended set of haar-like features for rapid object detection.” *Proceedings of 2002 International Conference on Image Processing*, 1, 900-903.

Lowe, D.G. (2004). “Distinctive image features from scale-invariant keypoints.” *Int. Journal of Computer Vision*, 60(2), 91-110.

Lu, M., Chen, W., Shen, X.S., Lam, H.C., and Liu, J.Y. (2007) “Positioning and tracking construction vehicles in highly dense urban areas and building construction sites.” *Automation in Construction*, 16(5), 647-656.

Lu, W.L., Okuma, K., and Little, J.J. (2009). “Tracking and recognizing actions of multiple hockey players using the boosted particle filter.” *Image and Vision Computing*, 27(1-2), 189-205.

Lucas, B. and Kanade, T. (1981). “An image registration technique with an application to stereo vision.” *Proceedings of the International Joint Conference on Artificial Intelligent*, William Kaufmann, Vancouver, 674-679.

Maggio, E. and Cavallaro, A. (2009). “Accurate appearance-based Bayesian tracking for maneuvering targets.” *Computer Vision and Image Understanding*, 113, 544-555.

Makhmalbaf, A., Park, M.-W., Yang, J., Brilakis, I., and Vela, A.V. (2010). “2D vision tracking methods’ performance comparison for 3D tracking of construction resources.” Construction Research Congress, Banff, Canada, 459-469.

Marfil, R., Molina-Tanco, L., Rodriguez, J.A., and Sandoval, F. (2007). “Real-time object tracking using bounded irregular pyramids.” *Pattern Recognition Letters*, 28, 985-1001.

Mathes, T. and Piater, J.H. (2006). "Robust non-rigid object tracking using point distribution manifolds." *Proceedings of the 28th Annual Symposium of the German Association for Pattern Recognition (DAGM)*, Springer, Berlin, 4174, 515-524.

Mcfarlane, N.J.B. and Schofield, C.P. (1995). "Segmentation and tracking of piglets in images." *Machine Vision and Applications*, 8(3), 187-193.

Mimbela, L.E.Y., Klein L.A. (2000). "Summary of vehicle detection and surveillance technologies used in intelligent transportation systems." Federal Highway Administration, Intelligent Transportation Systems Joint Program Office.

National Academy of Engineering (2008). "Restore and improve urban infrastructure", Grand Challenges for Engineering, <http://www.engineeringchallenges.org/cms/8996/9136.aspx> (Accessed June 4, 2012).

Nguyen, H. T., Worring, M., van den Boomgaard, R., and Smeulders, A. W. M. (2002). "Tracking nonparameterized object contours in video." *IEEE T Image Process*, 11(9), 1081-1091.

Nistér, D. (2004). "An efficient solution to the five-point relative pose problem." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6), 756-770.

Oloufa, A.A., Ikeda, M., and Oda, H. (2003). "Situational awareness of construction equipment using GPS, wireless and web technologies." *Automation in Construction*, 12(6), 737-748.

Osman, H. M., Georgy, M. E., and Ibrahim, M. E. (2003). "A hybrid CAD-based construction site layout planning system using genetic algorithms." *Automation in Construction*, Elsevier, 12(6), 749-764.

Park, M., Nepal, M. P., and Dulaimi, M. F. (2004) "Dynamic Modeling for Construction Innovation." *Journal of Management in Engineering*, ASCE, 20(4), 170-177.

Pizarro, O., Eustice, R., and Singh, H. (2003). "Relative pose estimation for instrumented, calibrated platforms." *Proceedings of the 7th Digital Image Computing: Techniques and Applications*, 601-612

Rashidi, A., Dai, F., Brilakis, I., and Vela, P. (2011). “Comparison of camera motion estimation methods for 3D reconstruction of infrastructure.” *ASCE International Workshop on Computing in Civil Engineering, Miami*, FL, USA.

Reid, D.B. (1979). “An algorithm for tracking multiple targets.” *IEEE Trans. Automatic Control*, 24(6), 843-854.

Rodriguez, T., and Garcia, N. (2010). "An adaptive, real-time, traffic monitoring system." *Mach Vision Appl*, 21(4), 555-576.

Ross, D., Lim, J., Lin, R.-S., and Yang, M.-H. (2008). “Incremental learning for robust visual tracking.” *International Journal of Computer Vision*, 77(1), 125-141.

Rousson, M. and Deriche, R. (2002). “A variational framework for active and adaptive segmentation of vector valued images.” *Proceedings of IEEE Workshop on Motion and Video Computing*, 56-61.

Sawyer, T. (2008a) “$1-billion jigsaw puzzle has builder modeling supply chains.” *Engineering News-Record*, The MacGraw-Hill Companies, April 2008.

Sawyer, T. (2008b) “South Korean research in electronic tagging is forging ahead.” *Engineering News-Record*, The MacGraw-Hill Companies, April 2008.

Schreiber, D. (2008). “Generalizing the Lucas-Kanade algorithm for histogram-based tracking.” *Pattern Recognition Letters*, 29, 852-861.

Shafique, K. and Shah, M. (2003). “A non-iterative greedy algorithm for multi-frame point correspondence.” *Proceedings of ICCV 2003, IEEE*, Nice, France, 110-115.

Sharpira, A. and Rosenfeld, Y. (2011). “Achieving construction innovation through academia-industry cooperation-keys to success.” *Journal of Professional Issues in Engineering Education and Practice*, ASCE, 137(4), 223-231.

Shi, J. and Tomasi, C. (1994). “Good features to track.” *Proceedings of CVPR 1994, IEEE*, Seattle, WA, 593-600.

Shotton, J., Johnson, M., and Cipolla, R. (2008). “Semantic texton forests for image categorization and segmentation.” *Proceedings of 2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1-8.

Son, H. and Kim, C. (2010). “3D structural component recognition and modeling method using color and 3D data for construction progress monitoring.” *Automation in Construction*, 19(7), 844-854.

Stauffer, C.and Grimson, W.E.L. (2000). “Learning patterns of activity using real-time tracking.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 747-757.

Sugano, H. and Miyamoto, R. (2009). “Parallel implementation of pedestrian tracking using multiple cues on GPGPU.” *Proceedings of IEEE 12th International Conference on Computer Vision Workshops*, 900-906.

Swain, M.J. and Ballard, D.H. (1991). “Color Indexing.” *International Journal of Computer Vision*, 7(1), 11-32.

Teizer, J., Lao, D., and Sofer, M. (2007a). “Rapid automated monitoring of construction site activities using ultra-wideband.” *Proceedings of 24th Int. Symp. on Automation and Robotics in Construction*, Construction Automation Group, Kerala, India, 23-28.

Teizer, J., Caldas, C. H., and Haas, C. T. (2007b). "Real-time three-dimensional occupancy grid Modeling for the detection and tracking of construction resources." *J Constr Eng M ASCE*, 133(11), 880-888.

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.

Teizer, J., and Vela, P. A. (2009). "Personnel tracking on construction sites using video cameras." *Adv Eng Inform*, 23(4), 452-462.

The New Georgia Encyclopedia (2008) “Georgia Department of Transportation”, <http://www.georgiaencyclopedia.org/nge/Article.jsp?id=h-2444> (Accessed June 4, 2012)

Torr, P.H.S (2002). “Bayesian model estimation and selection for epipolar geometry and generic manifold fitting.” *Int. Journal of Computer Vision*, 50(1), 35-61.

Tsechpenakis, G., Tsapatsoulis, N., and Kollias, S. (2004). “Probabilistic boundary-based contour tracking with snakes in natural cluttered video sequences.” *International Journal of Image and Graphics*,4(3), 469-498.

Turk, M., and Pentland, A. (1991). "Eigenfaces for Recognition." *J Cognitive Neurosci*, 3(1), 71-86.

Veeramani, D., Tserng, H. P., and Russell, J. S. (1998). “Computer-integrated collaborative design and operation in the construction industy.” *Automation in Construction*, Elsevier, 7(6), 485-492.

Victores, J. G., Martínez, S., Jardón, A., and Balaguer C. (2011). “Robot-aided tunnel inspection and maintenance system by vision and proximity sensor integration.” *Automation in Construction*, 20(5), Elsevier, 629-636.

Viola, P. and Jones, M. (2001). “Rapid object detection using a boosted cascade of simple features.” *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 511-518.

Wang, L. C. (2008). "Enhancing construction quality inspection and management using RFID technology." *Automat Constr*, 17(4), 467-479.

Yang, J., Arif, O., Vela, P. A., Teizer, J., and Shi, Z. K. (2010). "Tracking multiple workers on construction sites using video cameras." *Adv Eng Inform*, 24(4), 428-434.

Yilmaz, A., Javed, O., and Shah, M. (2006). "Object tracking: A survey." *Acm Comput Surv*, 38(4).

Yoders, J. (2008) “Integrated project delivery using BIM.” *Building Design & Construction* 49.5, April 2008, 30-44.

Yokoyama, M. and Poggio, T. (2005). “A contour-based moving object detection and tracking.” *Proceedings of ICCCN 2005, IEEE,* San Diego, CA, 271-276.

Zhang, Z. (1999). “Flexible camera calibration by viewing a plane from unknown orientations.” *Proceedings of the 7th IEEE Int. Conf. on Computer Vision, IEEE*, Kerkyra, Greece, 1, 666-673.

Zhang, Z.H. Li, W.H., and Li, B. (2009). “An improving technique of color histogram in segmentation-based image retrieval.” *Proceedings of the 5th International Conference on Information Assurance and Security*, 2, 381-384.

Zhu, A., Yeh, M.C., Cheng, K.T., and Avidan, S. (2006). “Fast human detection using a cascade of histograms of oriented gradients.” *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1491-1498.

Zhu, Z. H., and Brilakis, I. (2010). "Concrete column recognition in images and videos." *J Comput Civil Eng*, 24(6), 478-487.

VITA

MAN-WOO PARK

Man-Woo Park was born in Seoul, South Korea. He holds a B.S. in Engineering and an M.S. in civil, urban, and geo-system engineering. In 2008, He started to pursue a doctorate in civil engineering with a specialization in construction engineering and management. Man-Woo Park is an active member in several academic and professional organizations, and officially serves as a reviewer for ASCE journals. His research interests include image pattern recognition and filtering techniques for tracking and monitoring construction entities and highway vehicles. In 2012, he received 2012 Fiatech CETI (Celebration of Engineering and Technology Innovation) Award in the category of Outstanding Student Research Project.