Machine Vision Techniques for Crane Workspace Mapping

A Thesis

Presented to the

Graduate Faculty of the

University of Louisiana at Lafayette

In Partial Fulfillment of the

Requirements for the Degree

Master of Science

Mohammad Sazzad Rahman

Spring 2015

Machine Vision Techniques for Crane Workspace Mapping

Mohammad Sazzad Rahman

APPROVED:

Joshua Vaughan Assistant Professor of Mechanical Engineering	Ahmed Khattab Associate Professor of Industrial Technology & Mechanical Engineering
Charles Taylor Assistant Professor of Mechanical Engineering	Mary Farmer-Kaiser Dean of the Graduate School

DEDICATION

To my mother,

Hasina Begum,

for endless inspiration

ACKNOWLEDGMENTS

I would like to acknowledge the following:

- My mother, for all the sacrifices she made for me throughout my entire life. I would not be able to pursue a masters degree if it was not for her.
- My sisters, for their immense support.
- My advisor, Dr. Joshua Vaughan, for watching my back all the time during my stay at Lafayette, and for teaching me the proper way of doing research and writing scientific documents.
- My committee members, Dr. Charles Taylor and Dr. Ahmed Khattab, for agreeing to be in my thesis committee.
- My adoptive mother, Kara Murphy, father, Harry Murphy, sister, Colette and brother, Alex, for their unconditional love for me. It means so much having a family on the other side of the planet from home.
- My lab-mates, Bobby, Dare, Seema, Nicole, Yasmeen, Jasmin, Jordan, and Beau, for their excellent co-operation. It was a great pleasure for me to work with them.
- My friends in Lafayette, for their company. There are too many to name. Special thanks to the Bangladeshi community in Lafayette, and my Bangladeshi friends who live in the US, for making my life in Lafayette easier.
- Last but not the least, all the faculty and staff of the Mechanical Engineering Department at the University of Louisiana at Lafayette, especially Dr. Sally Mclnerny for the financial support I received from the department.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Cranes are used worldwide for transportation and material handling in a variety of industries and facilities, including manufacturing industries, shipyards, and warehouses. Safety and efficiency in crane operations are a concern, since these issues are closely related to productivity. One of the reasons for crane-related accidents is mistakes by the operator, some of which can be attributed to the limitations of the operator's field of view, depth perception, and knowledge of the workspace. These limitations are exacerbated by the dynamic environment of the workspace. One possible solution to these problems could be aiding the operator with a dynamic map of the workspace that shows the position of obstacles within it. In this thesis, two methods for mapping the crane workspace in near-realtime using computer vision are introduced. Several computer vision algorithms are integrated, and new techniques are introduced to generate a machine-vision-based map. A QR code-based mapping algorithm is also formulated. The algorithms can work independently. However, they can also be integrated, and the results show that a combination of these two mapping techniques produce the best results. The success of the pure machine-vision-based map and the QR code-based map depend on successful segmentation of color regions and detection of the QR codes, respectively. The combination of the two algorithms is a novel approach that ensures maximum obstacle detection. The algorithms produce a workspace map that can help the crane operator drive the crane more safely and efficiently.

Chapter 1

INTRODUCTION

1.1 Problem Statement

In overhead bridge cranes, like the one shown in Figure 1.1, a payload is suspended from a trolley that moves along a bridge. The bridge itself can also move, enabling the crane to serve a large area. Cranes have been used in construction and material handling since antiquity and have made great contributions to the progress of civilization. However, there are some risks in crane operation. One of the most dangerous risks in crane operation is payload oscillation. When the crane is moved through its workspace, the payload oscillates. If the oscillation is not controlled, the payload may collide with objects or people, causing damage and injury. According to the Bureau of Labor statistics, there were 78 crane-related fatal injuries per year from 2003 to 2005 [5]. These incidents occurred either directly or indirectly because of the crane, or the operator. According to the Crane Inspection and Certification Bureau, 90% of all crane accidents occur due to human error [6]. Half of U.S. crane accidents that had injuries in 2009 resulted in fatalities.

Control techniques, including both feed-forward and feedback methods, have been invented to eliminate crane payload oscillation. One of the most successful crane control techniques is input shaping [7–9], which shapes the command by convolving a sequence of impulses with crane operator input. The convolved signal is then used as the reference command. This technique has proven useful for vibration reduction of cranes [10–13].

Even though the control techniques used for vibration reduction greatly reduce the risk

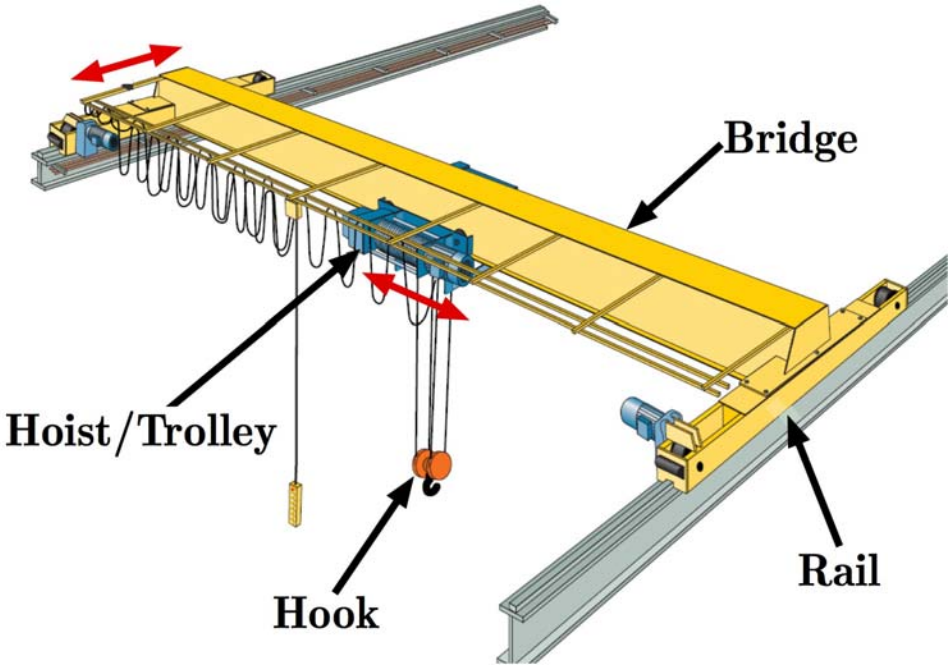


Figure 1.1: Overhead Crane [1]

of accidents from payload oscillation, there are still other kinds of risks associated with crane operation. Mistakes by the operator resulting from poor knowledge of the entire workspace or the operator's vision being blocked by the payload or obstacles are just two such risks. Either of these problems can lead to unsafe conditions and collisions with workspace obstacles.

To prevent payload collision with obstacles, companies like CAMotion [14] have implemented virtual boundaries around obstacles in crane workspace. However, obstacle boundaries typically have to be manually entered either through a programming interface or a series of offline obstacle-identification steps. Normal crane operations must be stopped when these obstacle boundaries are being set up. This can work well for permanent obstacles, such as big machines, tanks and assembly lines. However, they are not suitable for dynamic obstacles, such as fork lifts, loading and unloading zones, and temporary machines.

Another company, named KONECRANE [15], has implemented user-defined protected areas. The crane automatically slows down and stops, and the operator is alarmed by the user-interface when the crane goes near a protected area. However, these techniques are not very useful for obstacles that change positions frequently, which is very common in crane workspaces. A robust automated workspace mapping and obstacle-identification method is obviously preferable. This thesis is the first step to an automated crane workspace mapping using machine-vision techniques.

In this thesis, two novel methods of crane workspace mapping using computer vision is proposed. The first algorithm uses only one camera mounted on the crane trolley. The map updates automatically as the crane moves through the workspace during normal operation, so the operator always has up-to-date knowledge of the workspace. The map could be particularly helpful when the operator's view is partially blocked by the payload or obstacles, or a new operator starts working and finds obstacles in new positions. Using the automatically generated map, virtual boundaries could be defined around the obstacles, to prevent payloads from running into them. These boundaries could be updated every time the map updates. A second technique for mapping using Quick Response (QR) codes is introduced, which uses the same camera and can either work individually or along with the first method. Together, these two methods produce a complete two-dimensional map of the crane workspace showing the most recently known positions of the obstacles.

1.2 Background and Literature Review

Machine-vision-based mapping or obstacle avoidance is not a totally new idea. Research has been conducted to model the workspace in order to avoid crushing obstacles during operations of heavy equipment by modeling the objects in 3D [16]. In addition, vision systems have been used for mapping and navigation for mobile robots [17–19] and developing augmented reality workspaces [20, 21]. Mapping and navigation were

carried out using robots equipped with sophisticated sensors such as stereo-vision cameras, RGB-D cameras, or a combination of multiple sensors. These techniques are quite expensive and can be difficult to implement in overhead cranes, where navigation is not a concern. In addition, a small number of sensors is preferred.

Simultaneous Localization and Mapping (SLAM) is a well established method for mapping an unknown environment or updating a map within a known environment by mobile robots and autonomous vehicles while they keep track of their current location [22, 23]. In the case of crane workspaces, the precise position of the crane is known, so a simpler algorithm can be applied to map the workspace.

Machine vision systems have been implemented on tower cranes [24, 25] where a camera is mounted on the crane facing downwards to get a bird's eye view of the site. The sole purpose was to help the operator see the workspace clearly. No mapping was attempted in this research, so it is difficult for the operator to locate the obstacles while driving the crane. Some researchers have used machine vision as a means of measuring the crane hook location for feedback control [26], and some have conducted research into crane control for obstacle avoidance [27, 28]. However, in this work, there was no attempt to map the crane workspace.

A critical part of the machine vision technique for mapping is image segmentation. The accuracy of the algorithm presented in this thesis heavily depends on how well the obstacles are separated from the background. A simple method for image segmentation is using thresholds. Some researchers worked on calculating thresholds automatically from picture contents [29], using fuzzy sets [30], or Kalman-Filtering [31]. Data clustering methods have also been applied in image segmentation [32, 33]. Edge-based and region-based algorithms are two of the most common types of image segmentation techniques. Edge-based algorithms detect the sharp changes in an image by de-

ciding whether pixels belong to an edge or not [34]. These algorithms work well with simple and grayscale images with very few features [34], and therefore are not suitable for segmenting a complex image taken of a crane workspace. Region-based algorithms are pixel-based segmentation methods because they require initial seed points to be selected [35]. This algorithms use data clustering to decide whether a neighboring pixel belongs to the seeded region or not and keep iterating until all the pixels in the image are examined. Region-based algorithms always give closed contours, which is important for the algorithm presented in this research, because the obstacles are drawn in the map based on the contours.

For region-based algorithms, the image should be properly seeded. Every seed leads to a segmented region in the image. Researchers have tried to make the seeding automated [36, 37]; however, these methods separate one region from another. The mapping technique presented in this thesis requires segmenting all the objects from one common background. The decision of whether a region belongs to the background or not plays a significant role in the success of the algorithm presented in this thesis.

One of the most-used seeded image segmentation algorithms is the watershed transformation [38, 39], which is a combination of both edge-based and region-based algorithms [40]. This algorithm requires markers in every region to be segmented. The markers act as seeds and can be manually entered by the user. To make the watershed transformation unsupervised, the markers should be determined automatically. Researchers have been working on creating automated markers using clustering and morphological operations [41, 42]. However, these methods are application specific, and none of them exactly apply to the segmentation required for crane workspace mapping. The methods treat every region as a separate object, instead of segmenting all the objects from a common background.

To measure the effectiveness of the machine-vision-based mapping, the efficiency of the segmentation method used should be evaluated. Different kinds of segmentation evaluation methods are available [43–46]. The evaluation methods can be divided into two categories: analytical and empirical [47]. The analytical methods measure the goodness of an algorithm by analyzing the algorithm itself and its internal parameters. The empirical methods, which are more suitable for this thesis, use test images as standards and compare them with the segmented images to measure their accuracy.

Empirical methods can be further divided into two subcategories: empirical goodness and empirical discrepancy methods [47]. Empirical goodness is the measure of percentage match between the segmented image and the reference image, and empirical discrepancy is the measure of percentage discrepancy between the segmented image and the reference image. These two methods help assess the mapping performance by comparing the similarity and discrepancy between the map and the workspace. Both are used in this thesis.

Although the idea of workspace mapping using QR codes is new, the innovative and intelligent use of QR codes is not. QR codes have been used for various purposes, such as self localization of mobile robots [48], where the codes were used as landmarks and contained a complete dataset, including geometrical position, normal vector, physical size and shape. QR codes have also been used in developing robot indoor position and orientation methods [49]. In this method, a camera was mounted on the robot facing upward. The camera detects the QR codes attached to the ceiling, and calculates the position and orientation of the QR code. From that information, the robot calculates its absolute position and direction of heading. QR codes have also been used in developing guide robots [50] that use a similar approach. The guide robot follows a path while detecting QR codes used as landmarks along the path. Another use of QR codes is the design and implementation of augmented reality systems [51]. These techniques

heavily depend on quick and accurate detection and recognition of QR codes.

Extensive research has been carried out for fast and accurate detection of QR codes [52], image analysis for QR code recognition [53], detection of low resolution QR codes using super resolution images generated from multiple low resolution images [54], QR code recognition in snapshot taken by mobile phones [55, 56], and extraction of QR codes from a non-uniform background [57]. Detection accuracy as high as 98% has been achieved from low resolution images [54]. For fast detection of QR codes from arbitrarily acquired image, the Viola-Jones algorithm has been used [58]. The fast detection is made possible by focusing on the most promising regions of the image, and classifying the patterns using rapid feature calculation. Thanks to these works, innovative uses of QR codes are now possible; mapping crane workspaces is one of them.

In this thesis, image processing techniques such as blurring, stitching, template matching, masking, thresholding, and watershed transformation are used to produce a computer vision based map. For the QR code-based mapping algorithm, QR code encoding and decoding techniques are used to encode and decode the QR codes, and edge detection technique is used to calculate the orientation of the QR codes. The empirical goodness and empirical discrepancy methods are applied to evaluate the performance of each of the mapping algorithms, and the combination of the two algorithms.

1.3 Thesis Contributions

This thesis aims to improve the crane operations by providing the crane operator with a near-realtime map of the entire workspace. A novel approach for mapping the workspace using computer vision is introduced. Image processing algorithms including image stitching, image thresholding, and watershed transformation are intelligently combined to generate the final map. Older maps included in the final map help in knowing the older positions the obstacles, which in turn helps in predicting the future positions of

the obstacles. A novel approach of mapping the workspace with QR codes is also introduced. The QR code based map can detect obstacles that are labeled with a QR code. On the other hand, the color-based segmentation used in the pure machine-vision-based mapping technique can detect any obstacle, provided that there is a good contrast between the foreground and the background. Therefore, it is preferred that the two techniques be applied together when possible.

This thesis will enable some significant future research, including automatic obstacle avoidance using computer vision. The map could be used to formulate an optimum path-planning algorithm. It could also be extended to the third dimension to produce a 3D map providing depth information, which would help decide the operator whether the crane can be hoisted over the obstacles.

1.4 Structure of the Thesis

In the next chapter, the tools used in this research are briefly introduced.

In Chapter 3, Mapping Using Machine Vision, the machine-vision-based mapping process is presented. The image acquisition method for mapping is discussed. Then, the image processing steps necessary to produce the map are introduced, followed by the image stitching algorithm. Next, the mapping process is explained in detail including the decay function and memory factor, and, in conclusion, the effect of different parameters on the mapping performance is discussed with examples.

In Chapter 4, Mapping Using QR Codes, an algorithm using QR codes for mapping the crane workspace is presented. The method of creating an obstacle database, the encoding and decoding method of the QR codes, and finding the center and angular orientation of the QR codes are discussed. Then the method of generating the map from the data read from the QR codes is explained. Next, the combination of these two maps is presented and compared with the QR code generated map and the machine-

vision-based map. At the end of this thesis, the conclusions made from this thesis are discussed.

Chapter 2

TOOLS USED FOR MAPPING

This chapter is an overview of the tools used in this research. The hardware used includes a small-scale crane, several types of cameras, and a variety of obstacles for testing. The OpenCV library was used for image processing, the python QR code package, qrcode 5.1, was used for generating QR codes, and the open-source ZBAR library was used for decoding QR codes. All code was written in the C++ programming language, except for the QR code generation, which was written in Python.

2.1 Experimental Setup

Figure 2.1 shows the workspace of the small-scale crane used for the examples presented in this thesis. The workspace is 108×89.75×69.25 inches. It is controlled using a Siemens PLC and driven by Siemens AC servomotors. For the mapping technique presented in this thesis, a camera was mounted on the crane trolley.

2.2 Cameras Used

The color images used in this thesis were taken with a Logitech C920 webcam, a Nokia Lumia 521, and an iPhone. The grayscale images in this thesis were taken with a Cognex In-Sight 7000 camera. The lens used with this camera was a manually-focused Fujinon lens with 9 mm focal length, f/1.4 to f/16 aperture, and a maximum 29°52′ horizontal and 22°37′ vertical angle-of-view. Figure 2.2 shows a Cognex In-Sight 7000 camera.

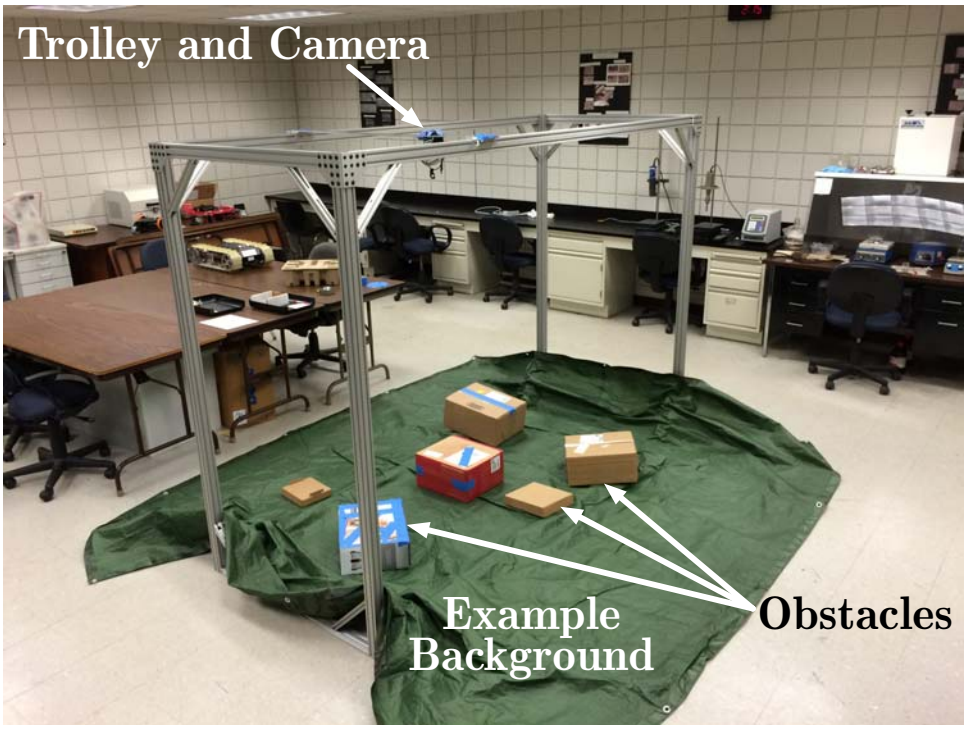


Figure 2.1: Small-Scale Crane Used in this Thesis

2.3 OpenCV

OpenCV is a widely used open-source computer vision library, written in C and C++. It supports Windows, Mac OS, iOS, Android, and Linux [59]. This library can be used in C, C++, Java, and Python. In this thesis, OpenCV 2.4 was used in most of the image processing, including image stitching, template matching, image thresholding, histogram calculation, watershed transformation, and morphological operations. All the code in this thesis was written in C++.

2.4 ZBAR

ZBAR is an open-source barcode reading library [60]. It can read barcodes from images, video streams, and sensors. ZBAR also supports reading and decoding QR codes. In this thesis, the C++ interface of the ZBAR library was used to decode all of the QR codes in the workspace.

Figure 2.2: Cognex Insight 7000 [2]

2.5 QRCODE 5.1

qrcode 5.1 is a python package for QR code generation. To generate QR codes it uses Python Imaging Library (PIL). While generating QR codes, the user can specify the size, version, level of error correction, and box size, depending on the application and amount of data to be encoded. All of the QR codes in this thesis were generated with this module.

Chapter 3

MAPPING USING MACHINE VISION

This chapter will explain the mapping algorithm using machine vision in detail. This algorithm combines a number of machine vision tools including stitching, template matching, and image segmentation to produce the final map. The method of including older maps with the latest map to give an idea of previous positions of the obstacles will be introduced. The mapping performance will be evaluated, and the segmentation methods will be compared. The results show that the mapping method is promising and can be further developed for industrial use.

3.1 Overview of Mapping

Figure 3.1 shows a flowchart of the mapping process [61]. A camera mounted on the crane trolley takes a picture each time it moves to a designated position. The camera can not capture the entire workspace in a single image, so the individual images need to be stitched together. The camera is mounted directly over the hook, so there is always the crane hook/payload in the image. The hook is located and masked before the images are stitched together.

Simple thresholding or the watershed transformation algorithm for foreground-background segmentation is applied for obstacle detection. For the watershed algorithm, a marker image is created where portions of the foreground and background are identified through a series of thresholding operations. The contours of the obstacles are extracted from the segmented image, and polygonal curves are estimated from the contours using the

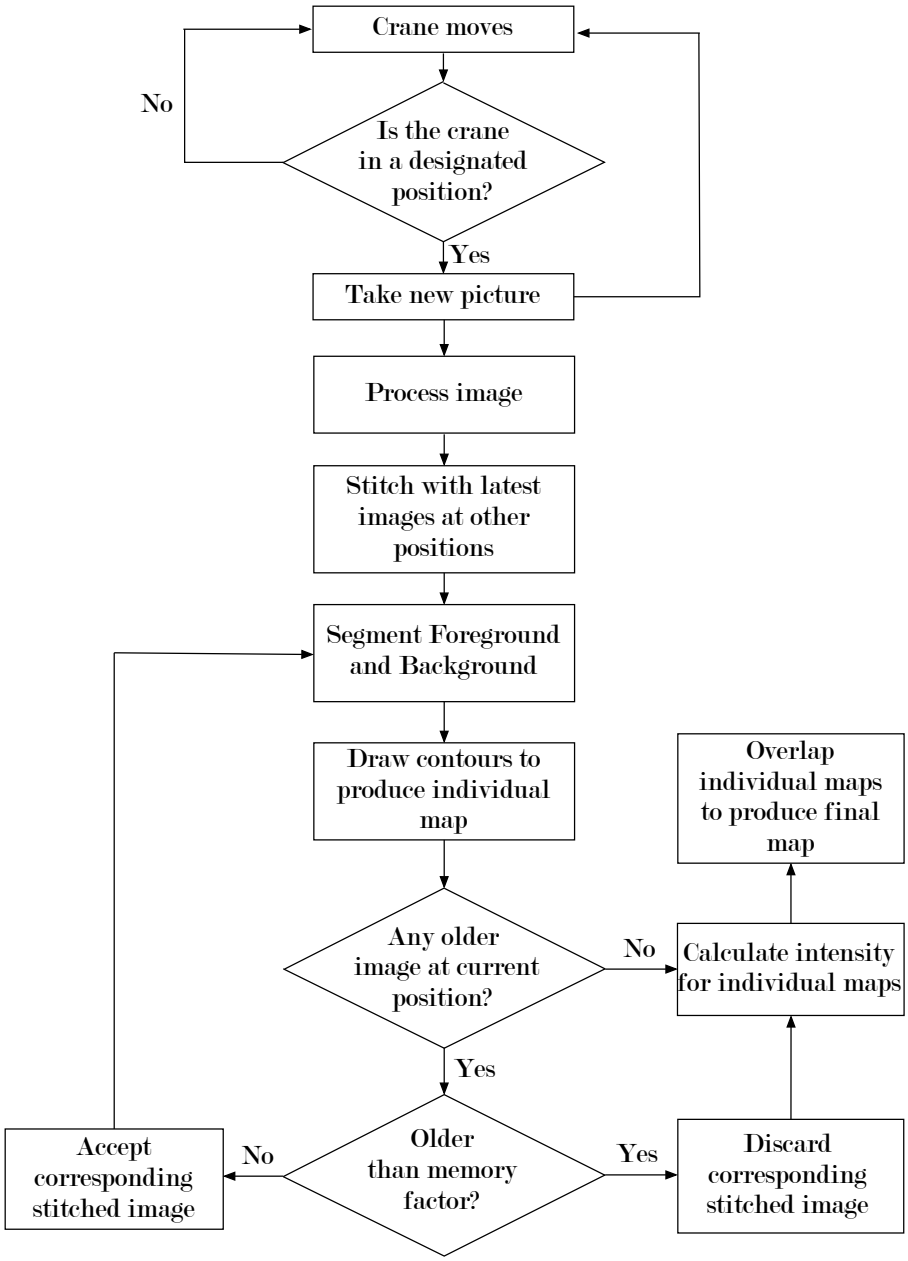


Figure 3.1: Flowchart of the Mapping Algorithm

Douglas-Peucker algorithm [62]. Then the map is drawn. This map is overlapped with the older maps, and a final map is generated. In the final map, the most recent positions of the obstacles are shown in red. The older positions are shown in yellow, the intensity of which decreases exponentially with time and depends on number of maps

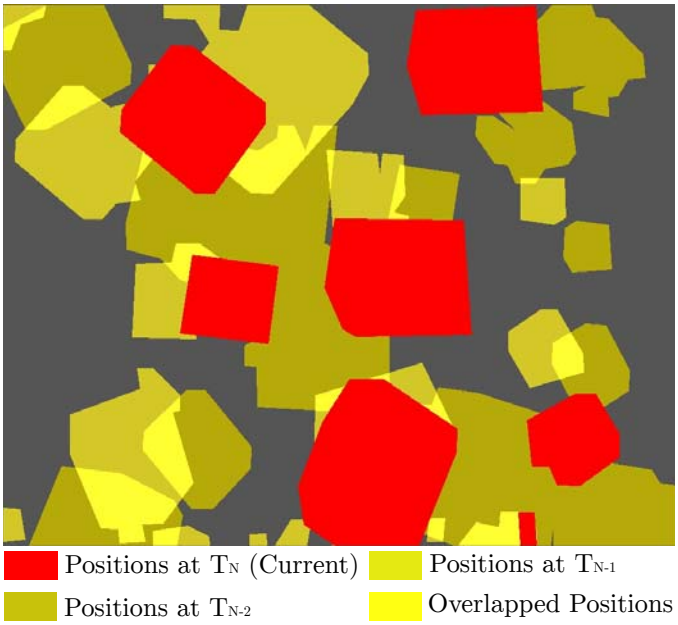


Figure 3.2: Crane Workspace Map

available.

An example map resulting from this process is shown in Figure 3.2 for the workspaces shown in Figure 3.3. The time indicated in the figures is the time when the latest image of each workspace was taken. The intensity of the yellow regions in the final map indicates how recently there were objects at those locations. For example, a high intensity of yellow indicates that an obstacle was at that location very recently. In Figure 3.2, T_{N-1} is more recent than T_{N-2}, so the obstacles at T_{N-1} are represented by a brighter shade of yellow than the obstacles at T_{N-2}. The locations of past obstacles are included with the intention of informing the operator of the possibility of obstacles being there in the future. For example, a product loading area in a factory may have been empty when the most recent map was generated, despite there often being obstacles there. The following sections will discuss each part of this algorithm in more detail.

(a) Workspace at Time T_{N-2} (b) Workspace at Time T_{N-1} (c) Workspace at Time T_N

Figure 3.3: Workspaces Used to Generate Map

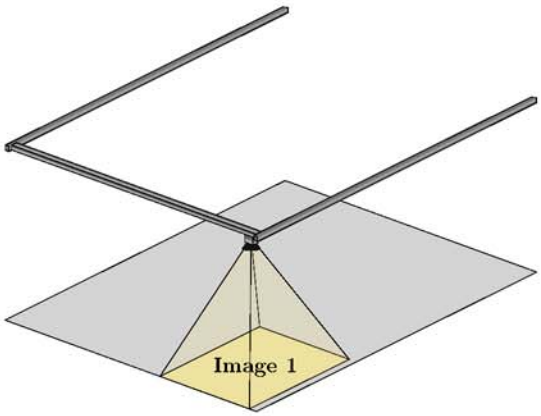
3.2 Image Acquisition

In order for the mapping process to work automatically, the image acquisition method also has to be automatic. The workspace is divided into segments, and a picture is taken in each segment to cover the entire workspace. There are designated positions with a buffered region around each. When the the crane trolley reaches one of these buffered regions, the camera automatically takes a picture. Then, it waits until it moves to another position, or for a certain period of time, depending on the frequency the map is updated if it stays at the same position, before taking the next picture. This automatic image acquisition works parallel to the normal crane operation. There is no stoppage time. However, for the first-time operation, it is recommended that the crane be moved through the entire workspace. Figure 3.4 shows the image acquisition process.

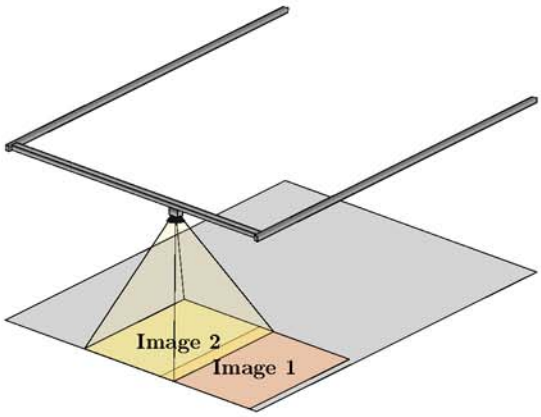
3.3 Individual Image Processing

In order to eliminate noise in the images, the workspace images are first blurred. Blurring smooths the images by reducing image noise and details. Gaussian blur and median blur are two of the most commonly used methods for blurring and are used in this work. Figure 3.5 shows an example image before and after it has been blurred.

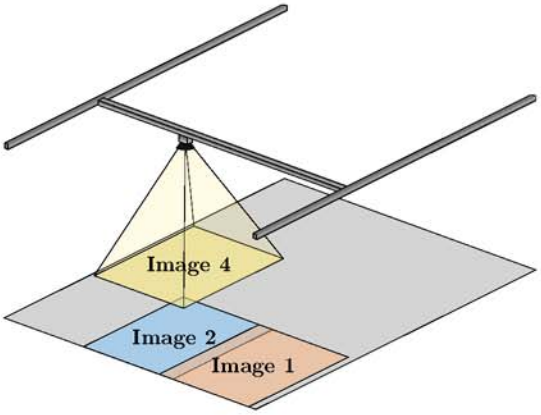
To separate the obstacles from the background, it is necessary to detect the background. If the background color is known and uniform, a simple thresholding can be used to



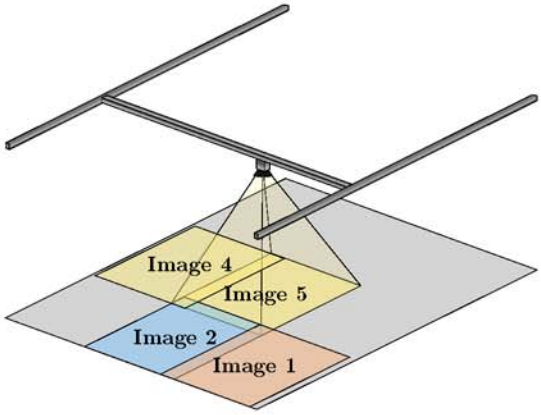
(a) Image at Position 1



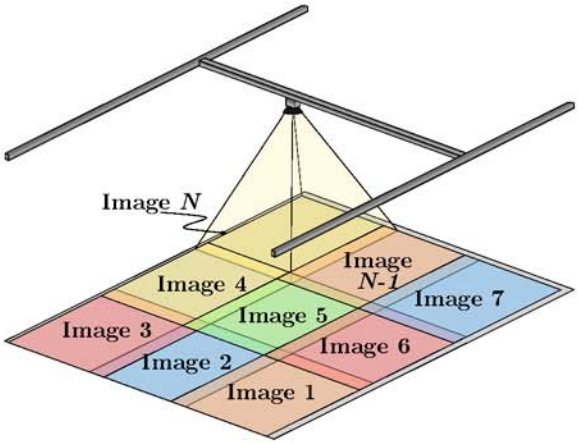
(b) Image at Position 2



(c) Image at Position 4



(d) Image at Position 5



(e) Image at Position N

Figure 3.4: Image Acquisition for Mapping

(a) Original Image (b) Blurred Image

Figure 3.5: Image Blurring

separate the background from the foreground. However, depending on the noise level and lighting, the background of the workspace may vary significantly. Moreover, the reflections of the lighting source and shadows of the objects can make it difficult to distinguish the obstacles from the background using simple thresholding.

A simple way to select the average background is calculating the image histogram for all three channels of an RGB image, as shown in Figure 3.6. It is assumed in this work that the background occupies a larger area than the obstacles. If this is the case, the most frequent pixel value can be selected as the background value for each channel. Figure 3.7 shows the background selection of the blue channel. The background of the other two channels are selected similarly.

Because the camera is mounted directly over the hook, the crane hook/payload will always be in the image. A simple solution is to cover that part of image with the calculated background color, as shown in Figure 3.8. In this figure, the crane hook mask is shown as semi-transparent to aid in understanding the algorithm. In practice, it is opaque.

One way to locate the hook in the image is template matching [63], which is a tech-

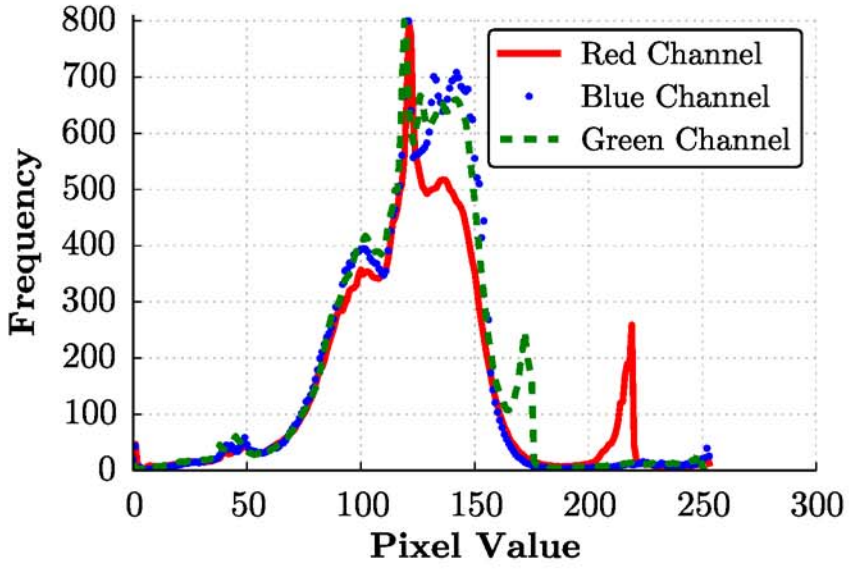


Figure 3.6: Histogram of an RGB Image

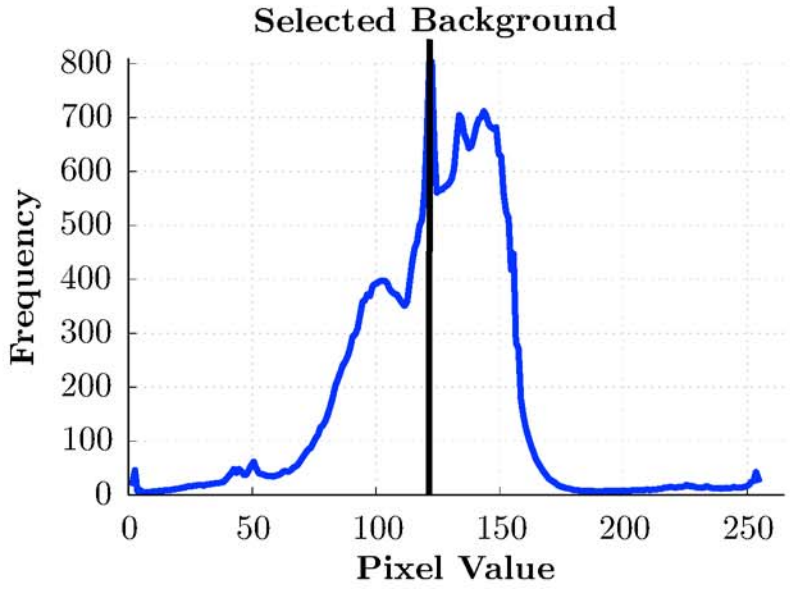


Figure 3.7: Background Selection of the Blue Channel from Histogram

nique for finding a part of a source image that matches with a smaller image known as a template image. The template image slides through the source image one pixel at a time in order to find a match. In this case, the source image is the picture taken by the camera, and the template image is a picture of the crane hook. A metric that repre-

Figure 3.8: Individual Image after Crane Hook is Masked

sents the goodness of the match is calculated at each position, and stored in a resultant matrix which is of the same size as the source image, as shown in Figure 3.9.

Each point of the resultant matrix contains the match-metric at that point. There are different matching methods available. The location in the resultant matrix where the highest or lowest value of the metric is found, depending on the matching method used, is the base point of the matching portion of the source image with the template image. For the matching method used in this particular example, the point in the resultant matrix with the lowest value is the base point of the location where the best match is found. The source image with the matched template is shown in Figure 3.10. After locating the hook and calculating its dimensions, it can be masked.

3.4 Image Stitching

Because the camera can not capture the entire workspace in a single image, the individual images need to be stitched together, as shown in Figures 3.11 and 3.12. The OpenCV stitching pipeline is used to stitch images, which uses automatic panoramic image stitching using invariant features [64]. This algorithm can seamlessly stitch mul-

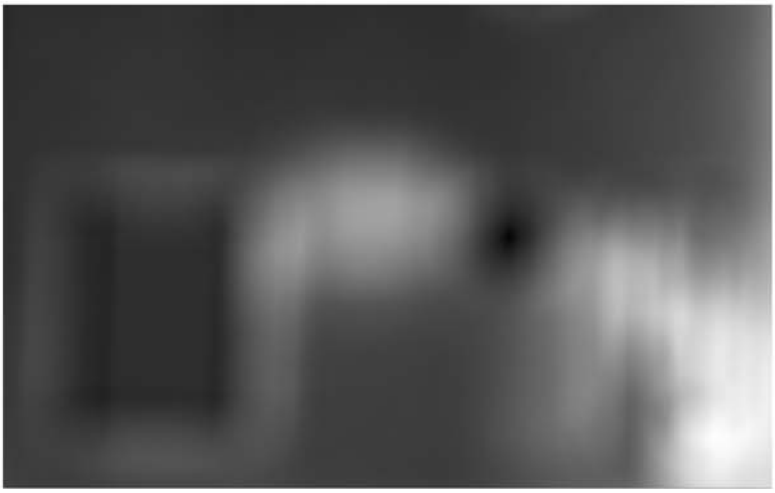


Figure 3.9: Resultant Matrix from Comparison of Source and Template Images

Figure 3.10: Source Image with the Template Located

tiple images irrespective of order or brightness, which makes it suitable for the mapping algorithm. It can also handle small rotations and blends the images seamlessly. In this algorithm, the SIFT (scale-invariant feature transform) features of all the images to be stitched are extracted and matched [65]. RANSAC (random sample consensus) [66] is used to find the feature matches between images that are geometrically consistent, and a probabilistic model is used to verify those matches. Then, connected components of the images are found and bundle adjustment [67] is performed to simul-

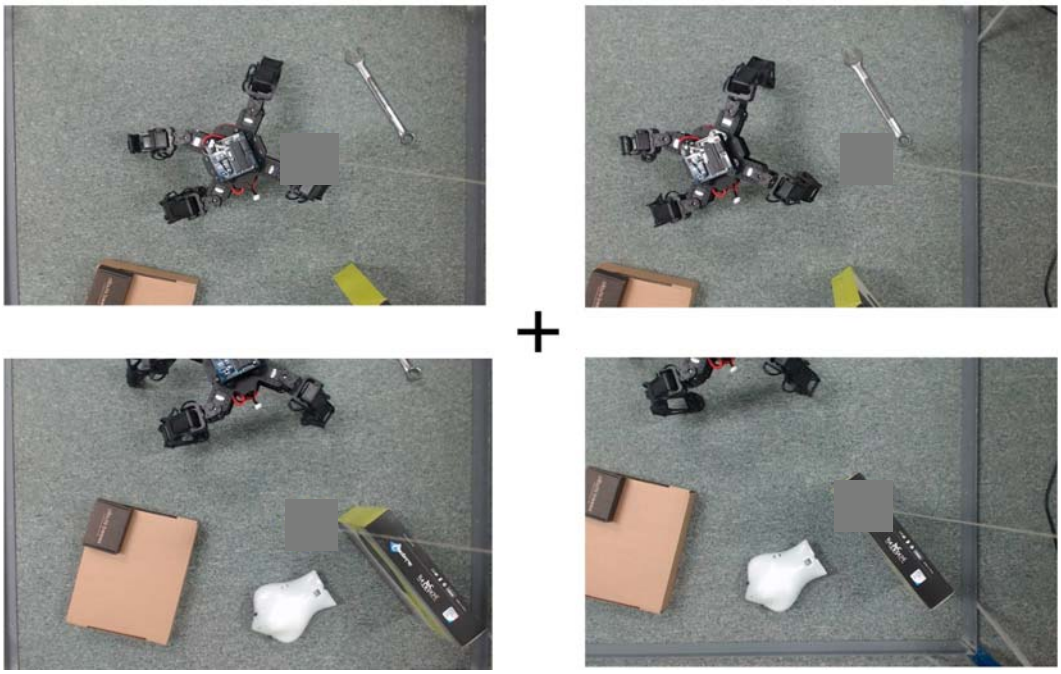


Figure 3.11: Individual Images to be Stitched

taneously solve for all the camera parameters. The images are then straightened and blended together using multi-band blending [64] so that the output stitched image is smooth.

Despite the advantages, there are some problems in using this stitching algorithm. There is a distortion in the stitched image because the distance of the camera from the objects is small. Also, it takes processing power and time to stitch multiple images together.

3.5 Obstacle Detection

Some advanced methods for foreground-background segmentation are available, including graph partitioning methods [68], region-based methods [69], and integration of multiple cues [70]. In this research, the watershed transformation, which is a combination of edge-based and region-based algorithms is applied because it always provides closed contours with a low level of noise and low computation time [71].

Figure 3.12: Stitched Image

For obstacle detection, a marker image has to be created where a portion of the foreground and background are marked. The foreground and the background of the image are segmented by the watershed algorithm based on these marked areas. To create the marker, the image is thresholded twice, through two three-channel scalars about the previously-selected background value. For the first thresholding operation, the threshold scalar range is larger, so that only the portions of the image that fall outside the threshold range are sure to be the foreground and are marked as such. For the second thresholding operation, the threshold range is smaller, so that only the parts of the image that fall inside the threshold range are sure to be the background and are marked as background. This is shown in Figure 3.13 for only blue channel. Using these operations, a marker is created, and the aforementioned regions are labeled inside it, as shown in Figure 3.14. Here, the background is marked in grey, the foreground is marked in white, and the rest of the image is black.

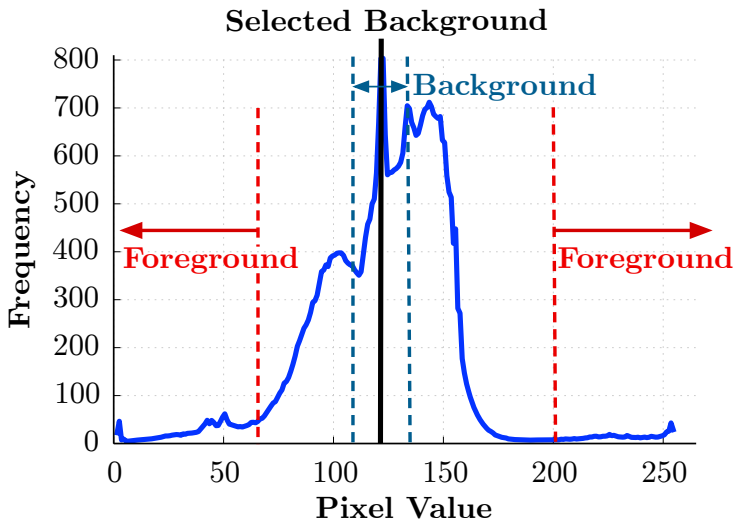


Figure 3.13: Background and Threshold Values Used for Marker Image Generation

Once the marker is determined, the watershed algorithm is applied to obtain the segmented image. The contours of the obstacles are extracted from the segmented image. Then, a polygonal curve is estimated using the Douglas-Peucker algorithm [62], and the contours are drawn, as shown in Figure 3.15.

Another simple method for image segmentation is converting the stitched image into grayscale, then thresholding it through an upper and a lower threshold value. The upper threshold is larger than the average background value, and the lower threshold is smaller than the average background value. Thresholding through the upper and lower thresholding values yield two segmented images, one in which objects brighter than the background are segmented, and one in which objects darker than the background value are segmented. Adding these two segmented images produce a image in which all the obstacles are separated. Then, the contours can be extracted and a polygonal curve can be drawn as before. Figure 3.16 shows the entire process for the same workspace shown before. The problem with this method is finding the right threshold values, and even if the right value is found, it is more sensitive to noise than the watershed transformation. Conservative threshold values can result in missed obstacles,

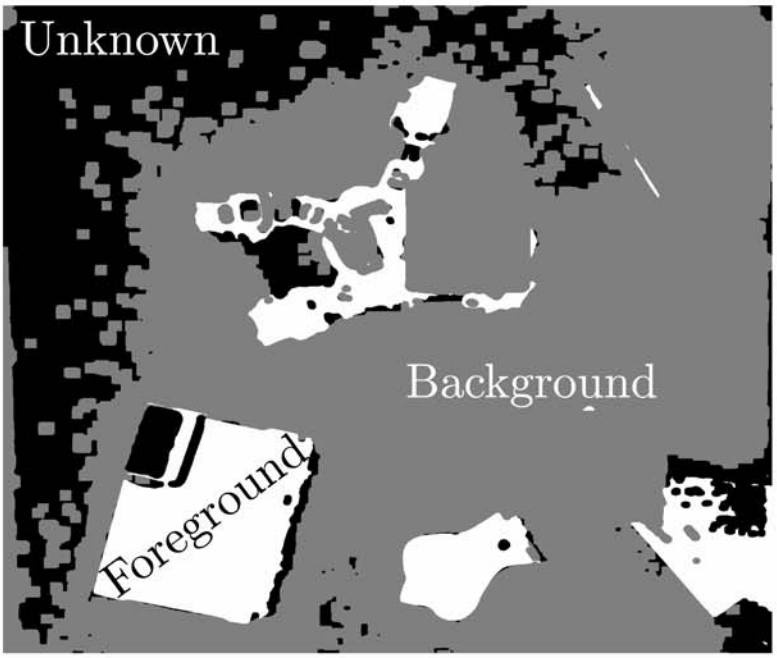


Figure 3.14: Marker Image Used for Seeding

Figure 3.15: Segmented Image After Contours Are Drawn

and large values can result in noise in the map. However, for a particular workspace condition, the threshold values are consistent. Therefore, these values can be calibrated using an interface like the one shown in shown in Figure 3.17, and used for the particular workspace condition.

The resultant map from the threshold values found from this calibration is shown in Figure 3.18. This map contains some noise. However, the process of generating it is simpler than the segmentation algorithms and works in some conditions. It also works with images of different colorspaces, after converting them into grayscale.

3.6 Overlapping Individual Maps

The mapping process presented in this thesis takes both the latest and older positions of the obstacles into account. The older positions of obstacles can be used as an indicator of the likelihood of there being an obstacle at that location in the future. Each time a new picture is taken by the camera at a particular position, an individual map is generated by stitching that picture with the most recent pictures at other positions. Then a final map is created by overlapping the latest map with older individual maps. The individual maps are overlapped in a way that the latest map is shown in red to indicate certainty of finding an obstacle. Older maps are shown in yellow, the intensity of which decreases with time when that maps were generated according to:

$$I_i = 100e^{-cm^2 t_i/(t_0 - t_l)} \tag{3.1}$$

where I_i represents percentage intensity of i^{th} map, m is the number of maps available, c is a scaling factor, t_i represents time since i^{th} map, and t_l and t_0 represent time since the most recent map and the oldest map, respectively. The scaling factor depends on the workspace conditions and the frequency with which the map is updated. For example, if the map is updated very often, a higher value of c is chosen, so that there is

(a) Obstacles Darker than the Background

(b) Obstacles Brighter than the Background

(c) Combination of the Two Thresholds

(d) Resultant Map

Figure 3.16: Grayscale Thresholding Process

Figure 3.17: Screenshot of the Calibration of Upper and Lower Threshold Values

Figure 3.18: Resultant Map After Calibration

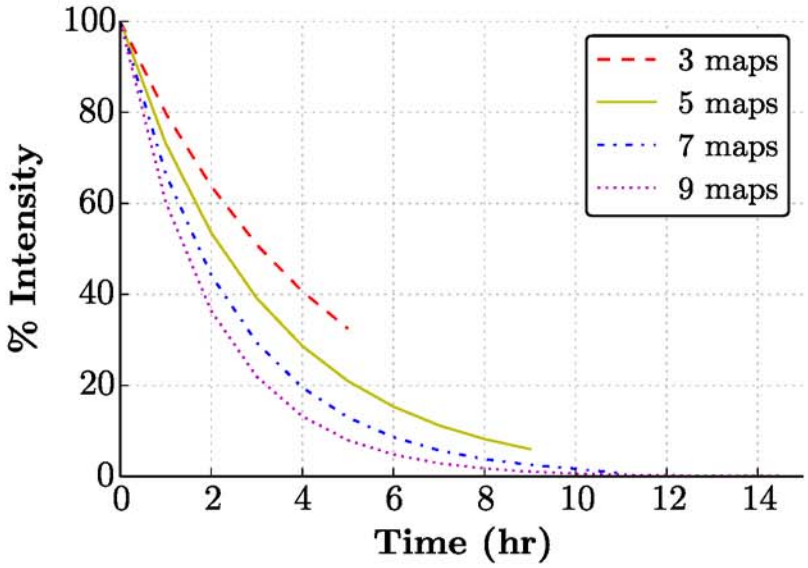


Figure 3.19: Example Definition of Intensity as a Function of Time

enough contrast between maps. On the other hand, if the map is updated less often, a lower value of c is chosen, so that the older maps are clearly visible.

Relatively recent maps are more likely to indicate an actual obstacle position. So, as time progresses, the intensity of older images is decreased. This indicates the decreased certainty of finding an obstacle at that location. The intensity depends both on the number of images available and time when the image was taken, as shown in Figure 3.19. With respect to time, intensity approaches linearity when the number of images available is reduced, making sure that the images do not lose weight too quickly when not many images are available in a particular location.

Using multiple older images at a given location provides additional information. If there is an overlap in the obstacle locations from past images, the overlapped area is brighter than in the individual maps. The overlapped area indicates there has been an obstacle present at these locations in the workspace at multiple times in the past. This suggests that the probability of finding an obstacle there in the future is greater.

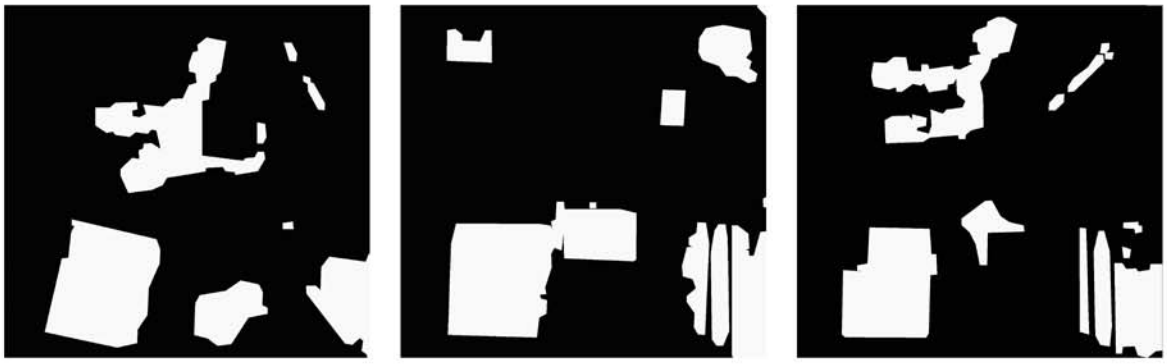
3.7 Memory Factor

If too many old maps are overlapped with the most recent map, the map becomes cluttered and too many obstacles at different points of time make the map confusing. Also, maps that are too old do not give much indication of probable future locations of the obstacles. To solve this problem, if an individual map is too old, it is discarded. A memory factor determines when an individual map is discarded. The larger the memory factor is, the older an individual map can be and still be taken into account for generating the final map. The memory factor is calculated using:

$$M = \begin{cases} t_0 e^{-nc_1} & \text{if } n > 2 \\ t_0 & \text{if } n \leq 2 \end{cases} \tag{3.2}$$

where M is the memory factor, t is the time since the oldest stitched image available, n represents number of older stitched images available, and c_1 is a scaling factor. If an image is older than M, then it is forgotten. If the number of images available is less than or equal to two, no image is forgotten. The memory factor is a function of both number of images, n, and time since the oldest available image, t. For a particular t, the memory factor reduces exponentially with number of images available. If a large number of images are available, the memory factor is smaller, so that more images are forgotten. If the number of images available is small, the memory factor becomes bigger, so that the older images are not forgotten. A maximum memory factor could be set up to make sure that the maximum number of images considered is limited.

As an example, individual maps generated at times T_N, T_{N-1}, T_{N-2}, where N is number of images, are shown in Figure 3.20. The resulting map is shown in Figure 3.21. In this map, the red areas show the most recent positions of the obstacles. Medium-bright yellow areas show next to the most recent positions, and the dark yellow areas show the oldest positions of the obstacles. The overlapping areas between the older obstacle



(a) Map at Time T_{N-2} (b) Map at Time T_{N-1} (c) Map at Time T_N

Figure 3.20: Individual Maps Used in the Final Map

Figure 3.21: Final Workspace Map

positions are shown by the brightest shade of yellow.

3.8 Performance Evaluation

Although this mapping technique is promising, there are some problems yet to be addressed. The background detection method described is based on the assumption that the background area is larger than the total obstacle area, which may not always be true. The thresholding values used for creating markers for the watershed algorithm are also variable. However, the threshold values are fixed for a particular workspace in a particular condition, and they can be calibrated. If there is an object that has the same color as the background, it may go undetected by the segmentation algorithms. Depending on the time of the day, the lighting condition changes in the workspace, which makes adjustment necessary. The effect of various parameters on the mapping performance are discussed in the following sections.

3.8.1 Effect of Memory Factor

Figure 3.22 shows the effect of memory factor on the obstacles detected as a percentage of total workspace area. The total obstacle area, past obstacle areas, and the overlap between past obstacle areas are shown. Obstacles occupy more area if the memory factor is bigger. In this example, when the memory factor is greater than T_{N-2}, all three individual maps are taken into account. The area occupied by obstacles is greater. When the memory factor is less than T_{N-2} but greater than T_{N-1}, the oldest of the three individual maps is forgotten. In this case, both the total obstacle area and past obstacle area are smaller. If the memory factor is less than T_{N-1}, only the most recent individual map is taken into account, the other two are forgotten. The total obstacle area is the smallest in this case, and there is no past obstacle area shown in the map in yellow. The effect of memory factor on the appearance of the map is shown in Figure 3.23.

As can be seen in (3.2), the memory factor depends on number of images available and the scaling factor. The scaling factor is chosen based on how frequently the map up-

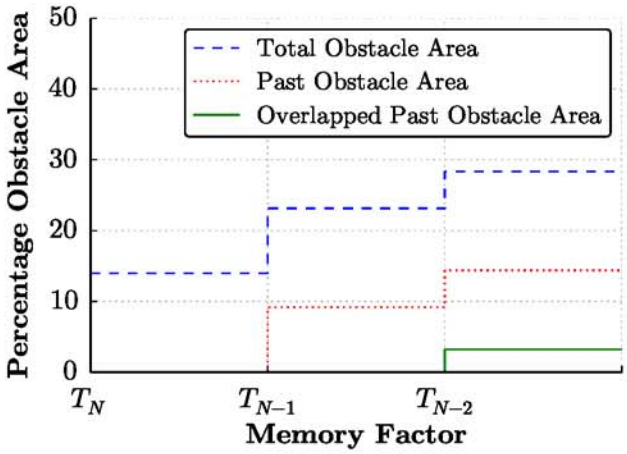


Figure 3.22: Obstacle Detected as a Percentage of Workspace Area vs Memory Factor

(a) M<T_{N-1} (b) T_{N-1}<M<T_{N-2} (c) M>T_{N-2}

Figure 3.23: Effect of Memory Factor on Final Map

dates. If the crane operates slowly, the map also updates slowly. Therefore, a smaller scaling factor is desired, so that older maps are not discarded quickly. Figure 3.24 shows the effect of these parameters on the memory factor for a time duration of 13 hours. For a particular scaling factor, the memory factor is greater if there is a fewer number of past images available. For example, in Figure 3.24, for a scaling factor of 0.2, images older than 3.2 hours are forgotten when number of images available is 7, while images older than 7.1 hours are forgotten when only 3 images are available. This ensures that a fewer number of images are forgotten when there are fewer images available.

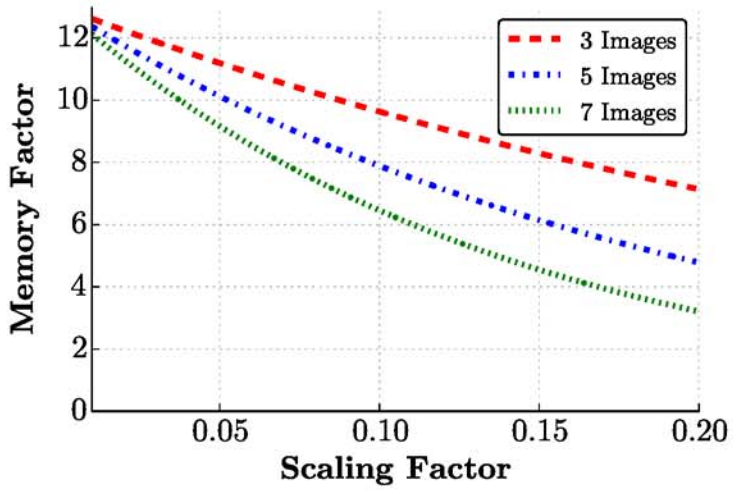


Figure 3.24: Memory Factor as a Function of Scaling Factor

3.8.2 Effect of Parameters on Intensity

The effect of number of available older maps on the intensity of older maps as a function of time is shown in Figure 3.25 for a normalized time range of 0.04 to 1. The normalized time $t_i/(t_0 - t_l)$ is the time since i^{th} map was generated divided by the total range of time over the available past maps. The latest map, which is shown in red is omitted to show the intensity of yellow in older maps clearly. From Figure 3.25, it can be concluded that for a particular number of available images, the scaling factor dictates the contrast between the older maps. The larger the scaling factor is, the greater the contrast.

The absolute intensity of obstacles as a function of normalized time for three different scaling factors is shown in Figure 3.26, which shows that the intensity curve declines quicker for larger value of scaling factor, yielding greater contrast. The intensity of past maps depends on the number of older maps available. Intensity decreases with an increase in the number of overlapped past maps available. This ensures that the map does not get cluttered when the number of past maps is large. In Figure 3.26, the markers on the curves represent the images used to form the maps in Figure 3.25.

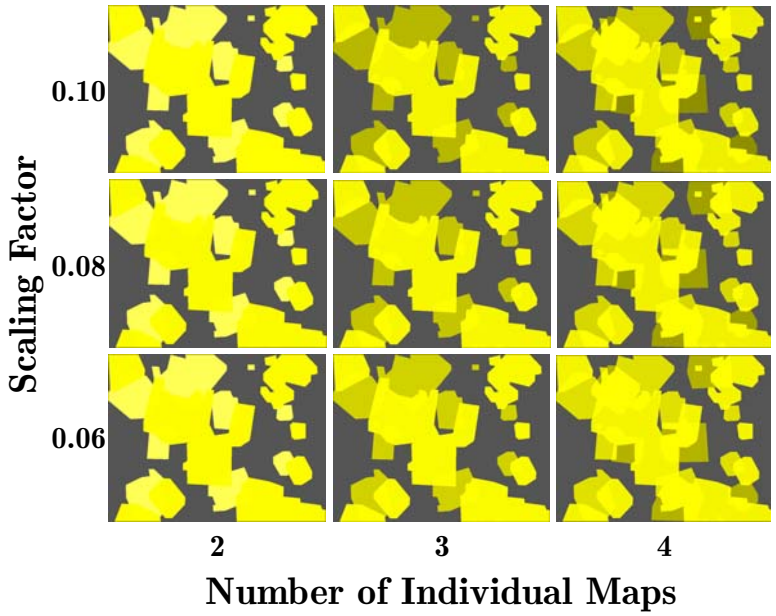


Figure 3.25: Effect of Number of Past Maps and Scaling Factor on Final Map

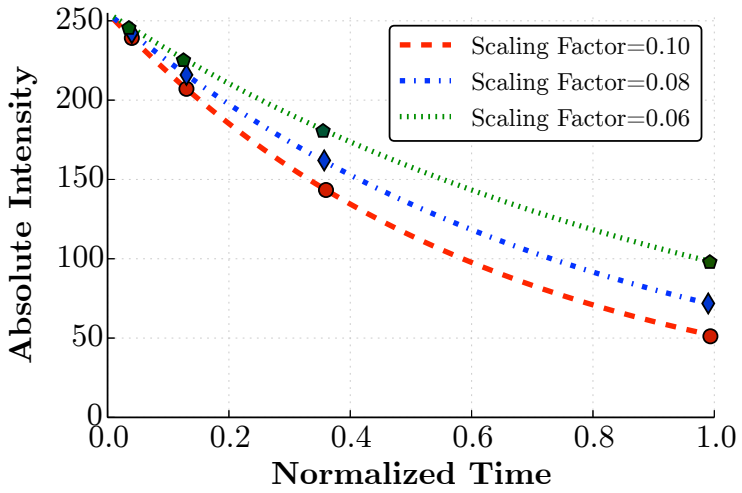


Figure 3.26: Absolute Intensity as a Function of Normalized Time

3.8.3 Comparison of Segmentation Methods

To evaluate the effectiveness of the proposed method, they are compared against a manually-generated map, drawn directly on top of the stitched images, rather than the measurements of the actual obtacles. As such it provides a method to compare segmentation algorithms against one another, but is only an approximation of their absolute accuracy. The results differ based on the contrast between background and foreground and different camera parameters such as exposure, brightness and contrast. However, for a particular camera and a particular set of workspaces, trends are found when the two segmentation methods are compared.

The workspaces used for the comparison of the two segmentation techniques are shown in Figure 3.27. These workspaces vary in background color, workspace size, and the color, shape and size of the obstacles. Segmented images resulting from grayscale thresholding and watershed transformation are compared against the manually-generated images for all of these eight workspaces. Workspace 1 is shown in Figure 3.28, along with the manually-generated image used for comparison and the segmentation results.

Figure 3.29 compares the percentage match of pixels with the manually-generated map for the grayscale thresholding method and the watershed transformation for different workspaces. Every pixel of the map is compared with the corresponding pixel of the manually-generated map. The figure shows that in most cases the watershed transformation algorithm yields a more accurate map than the grayscale thresholding algorithm. Figure 3.30 compares percentage obstacle pixels missed, that is the percentage of obstacle pixels in the manual map detected as background in the actual map.It shows that the grayscale thresholding results in higher percentage of missed obstacles than the watershed transformation. Figure 3.31 compares falsely detected obstacle pixels, that is the percentage of foreground pixels in the generated map that are background pixels in the manually drawn map. It shows that the watershed transfor-

(a) Workspace 1

(b) Workspace 2

(c) Workspace 3

(d) Workspace 4

(e) Workspace 5

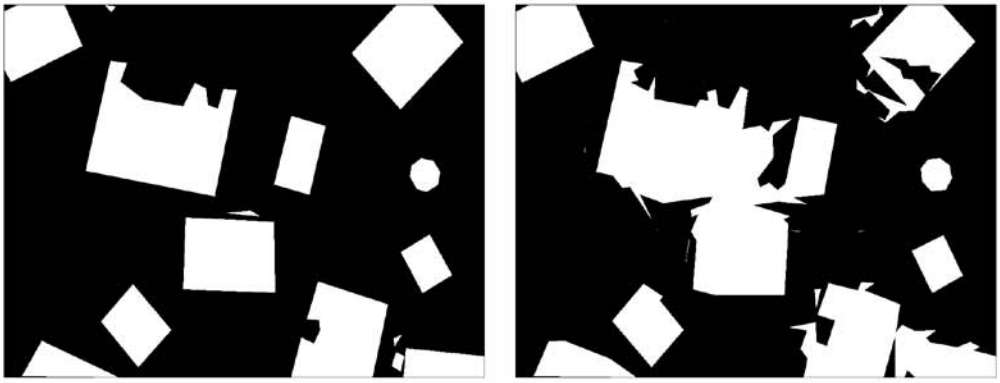
(f) Workspace 6

(g) Workspace 7

(h) Workspace 8

Figure 3.27: Workspaces Used for Evaluating Segmentation Performance

(a) Workspace Image (b) manually-generated Image



(c) Map Using Grayscale Thresholding (d) Map Using Watershed Tansformation

Figure 3.28: Example Segmentation Comparison

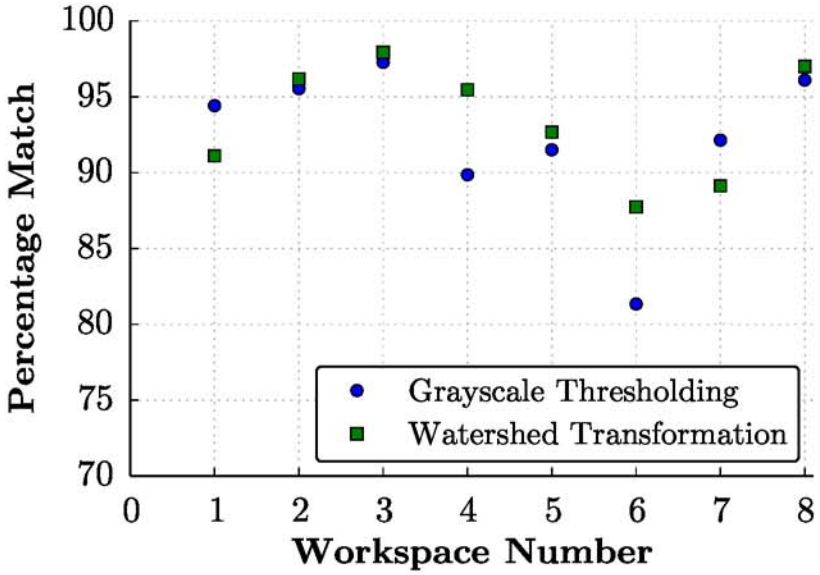


Figure 3.29: Percentage Match of Obstacle Pixels in Different Workspaces

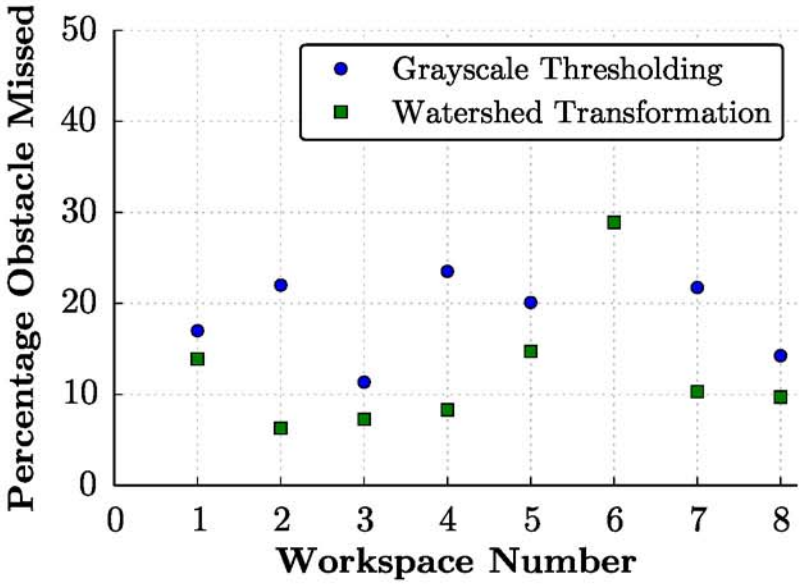


Figure 3.30: Percentage Obstacle Pixels Missed in Different Workspaces

mation algorithm has a tendency of detecting larger area than the actual obstacle size. Overall, the watershed transformation yields better results. However, it is harder to determine the threshold values for three channels in compared to only one channel. Moreover, the computation power and time required for the grayscale thresholding is less than the watershed transformation.

3.9 Chapter Conclusion

In this chapter, a novel approach of machine-vision-based mapping of crane workspace was introduced. The image acquisition method and image processing methods used were discussed. A simple way to mask the crane hook after it had been located using template matching was explained, and the image stitching method to stitch the images together and image segmentation methods to identify the obstacles were discussed. The idea of showing the older maps with different intensities depending on time and the criteria of discarding an older map was explained. Finally, mapping performance depending on different parameters was discussed, and the results of the two segmentation methods were compared. The results show that the mapping techniques can suc-

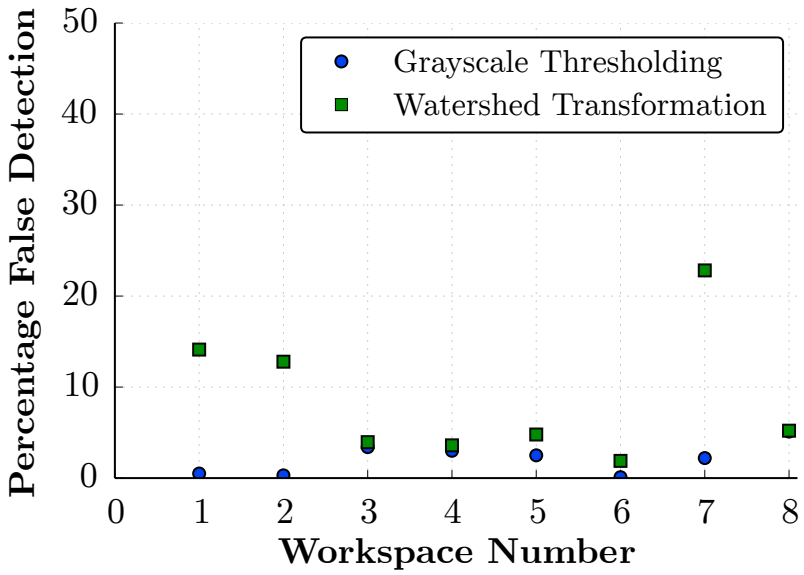


Figure 3.31: Percentage of Falsely Detected Obstacles in Different Workspaces

cessfully detect as much as 96% of the obstacle area, and the watershed transformation method is more robust when it comes to successfully detecting obstacles.

Chapter 4

MAPPING USING QR CODES

This chapter will introduce another method for mapping the crane workspace, which is based on QR codes. The algorithm requires the knowledge of the dimensions of the obstacles likely to appear in the map. These obstacles are labeled with QR codes, and a database is created to store the information of the known obstacles. The camera mounted on the crane trolley takes pictures of the workspace. The QR codes in the pictures taken by the camera are read, and a map is generated by matching the identified QR codes to the database. Unless all the obstacles in the workspace are labeled with a QR code, this mapping technique shows only a fraction of the total number of obstacles. The QR code-based map is combined with the previously described machine-vision-based map to ensure that maximum number of obstacles are shown in the map.

4.1 QR Code Overview

The Quick Response code (QR code) is a kind of two dimensional barcode. Barcodes are used all over the world for encoding information of the items onto which they are affixed. The QR code was first developed for the automotive industry in Japan [72]. Now, it is very popular worldwide because of its greater data storage capacity and quick readability. The datatypes QR codes support include binary, numeric, alphanumeric and kanji. Figure 4.1 shows an example QR code, encoded with the C.R.A.W.LAB web address (http://www.ucs.louisiana.edu/~jev9637/).

Figure 4.1: QR code for the URL of CRAWLAB

4.1.1 Encoding and Decoding

There are several standards for encoding data in QR codes. A QR code consists of black dots, called modules, on a white background. These modules create a unique pattern for the data encoded in the QR codes. For decoding, an imaging device, typically a camera, and a processor are used. The imaging device reads the QR codes, and the processor extracts the data from the pattern present on the QR codes [73]. The processor first locates the position blocks, shown in Figure 4.2, at the corners of the QR coded image. It also locates a fourth square, known as an alignment block, near the other corner to correct distortions. Then, the small modules are converted to binary numbers and validated with an error correcting code.

4.1.2 Storage

The storage capacity of a QR code depends on the encoded datatype, version, and error correction level. The possible datatypes are numeric, alphanumeric, binary and kanji. There are 40 available versions of QR codes. Version 1 is a 21x21 matrix, which can store the lowest amount of data. With each higher version, four additional rows

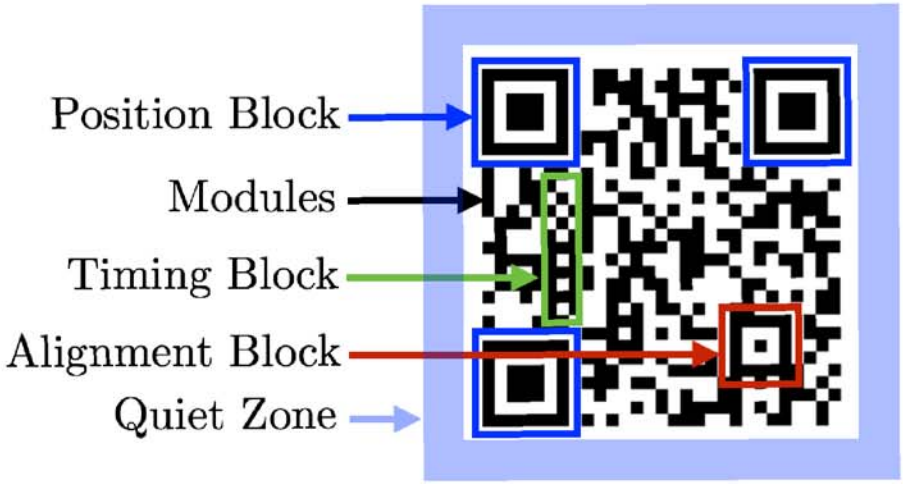


Figure 4.2: Design of a QR Code

Table 4.1: Different Levels of Error Correction in QR codes

Level L	up to 7 % damage
Level M	up to 15% damage
Level Q	up to 25% damage
Level H	up to 30% damage

and columns are added. Version 2 is a 25x25 matrix, version 3 is a 29x29 matrix, and so forth. Version 40 is a 117x117 matrix; it has a maximum storage capacity of 1,264 characters of ASCII text.

4.1.3 Error Correction

QR codes are robust. They can sustain damage and remain readable even if a part of the code is unreadable. This is made possible by Reed-Solomon Error Correction [74]. Backup data are added to make sure that the QR code is readable even if part of it is damaged. The error correction levels are summarized in Table 4.1. It shows that the QR codes with correction level L can survive upto 7% damage, and the QR codes with correction level H can survive up to 30% damage and be still readable.

For higher correction levels, the QR code has to accommodate backup data as well as the original data, which means more rows and columns of modules are required. As

a result, the QR code becomes denser. This is shown in Figure 4.3. It shows that the modules become smaller and denser as the error correction level increases. Two modules at the bottom left corner of a QR code shows the error correction level used in that QR code. Higher correction levels are convenient for industrial use where it is challenging to keep the QR codes clean and undamaged. However, higher correction levels reduce the data storage capacity of the QR codes.

It is also desired to have higher correction levels for the mapping technique proposed in this thesis. Since a portion of the image taken from the crane trolley is blocked by the crane payload, a higher correction rate provides a better chance to retrieve the data even if part of the QR code is obscured by the payload.

4.2 Mapping Algorithm

The advantages offered by QR codes facilitate a method for workspace mapping. For known obstacles, this method could be more reliable than the algorithm presented in the previous chapter, because the exact dimensions of the obstacles are already in the

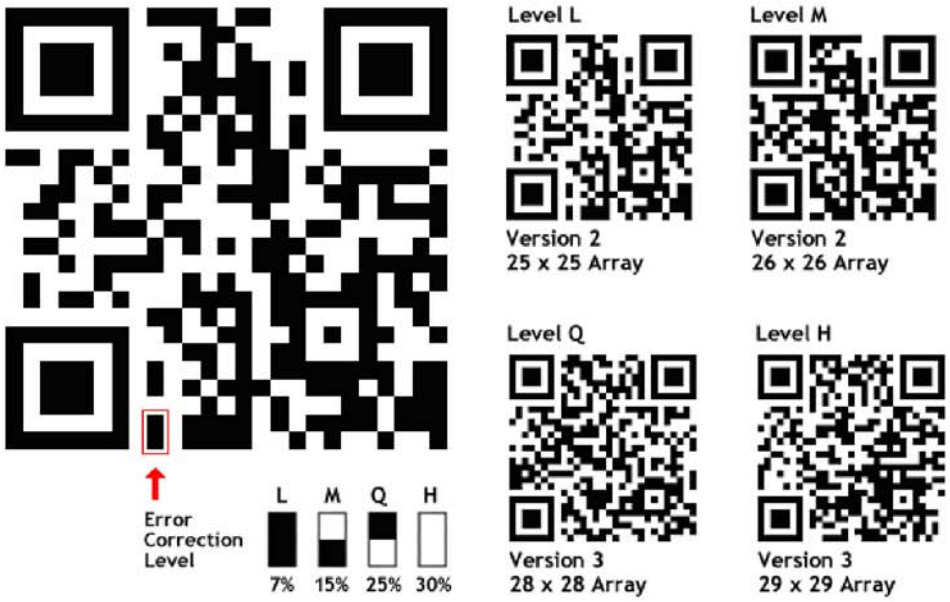


Figure 4.3: QR Code Error Correction Levels [3]

database and the number of undetected obstacles can be counted. The database file should be updated whenever an object is added or removed from the workspace.

Figure 4.4 shows the key steps of mapping using QR codes. In order for this algorithm to work, first a database of all the known obstacles is created. Each obstacle listed in the database is labeled with a QR code, the position of which with respect to the obstacle is listed in the database.

The latest stitched image of the workspace is taken as the input. All the QR codes in the stitched image are decoded, and the data are recorded. The recorded data includes a keyword for the obstacle each QR code is attached to and the position and orientation of that QR code. Next, for each QR code found in the image, the database is searched for the matching obstacle using the keyword. When a match is found, the dimensions of the obstacle and its relative position from the QR code is read from the database and from the obstacle height, the conversion factor from the physical units to pixels is calculated.

Next, the obstacle position and orientation is calculated from the QR code position and orientation, and using the conversion factor, the data are converted into pixel units. Finally, the obstacle is drawn on the map. The process continues until all the obstacles labeled with QR codes read from the stitched image are drawn. Figure 4.5 shows an example workspace, a complete map is shown in Figure 4.6.

4.2.1 Creating Obstacle Database

Since the mapping technique using QR codes works only for known obstacles, a database is necessary for the algorithm, which is created in text format where obstacles with their types and dimensions are listed. The position of each QR code relative to the obstacle onto which it is affixed is also listed. Taking precise measurements of the obstacles and putting them in correct order is imperative for the algorithm to work properly.

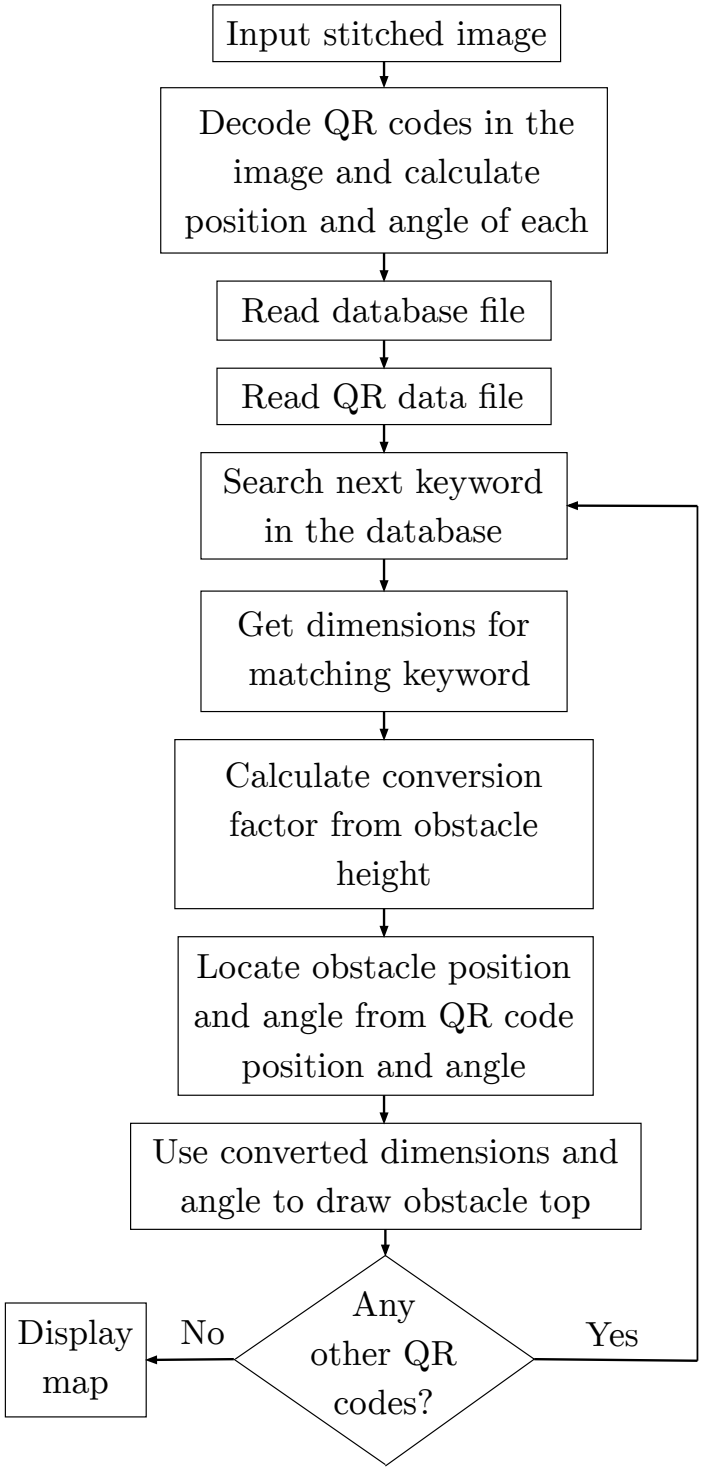


Figure 4.4: Flow Chart of Mapping Algorithm Using QR code

Figure 4.5: Workspace with QR Code Labeled Obstacles

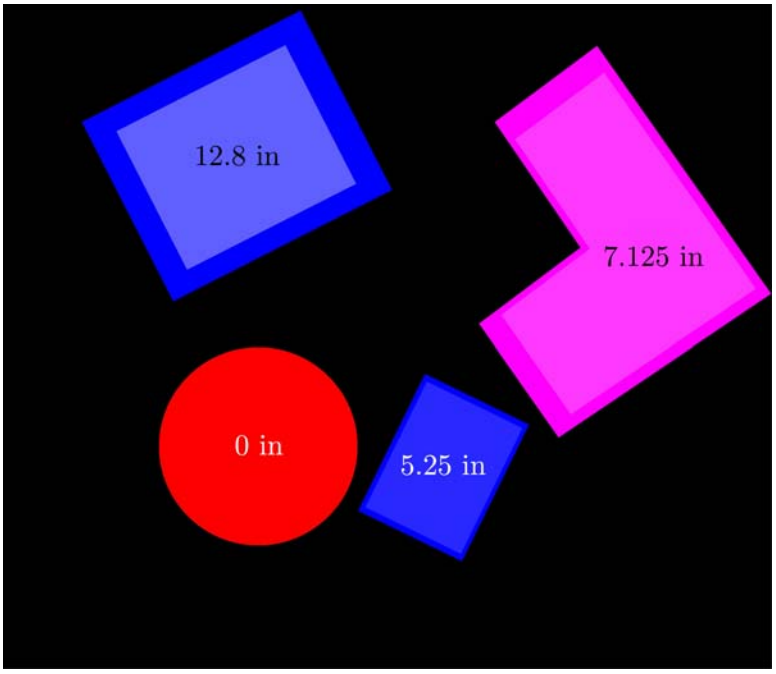


Figure 4.6: Resultant Map of the Workspace

Every obstacle in the database file is assigned with a code, which is also encoded in the QR code of the corresponding obstacle. This code is the keyword for a QR code

Table 4.2: Data Entry System for Circular Obstacles

tank1	r_x	radius	height	circle
c1	6	11	7	

to match with the database file. For example, the keywords for circular obstacles are `c1`, `c2`, ...,`cn`, and for rectangular obstacles, they are `r1`, `r2`, ...,`rn`. The database is updated with every new obstacle added to the workspace. For every obstacle, the data is listed in the following order: obstacle type (circle/rectangle/polygon), obstacle code (example: `c1`, `r3`, `p2` etc.), and dimensions.

For circular obstacles, the dimensions listed in the database are the relative distance of the QR code from the center, the radius, and the height of the circular obstacle. Since rectangular obstacles are common, and the labeling and data entry is easier than that of polygonal obstacles, they are treated separately. Moreover, polygons close to rectangular shape can be simplified as rectangles. For rectangular obstacles, the dimensions listed are the relative distance of the QR code from the lower left vertex, and the length, width, and height of the obstacle. In the case of polygonal obstacles, the number of sides of the polygon is included in the database. The position of each corner relative to the QR code center is listed in polar coordinate, followed by the height of the obstacle. All dimensions are specified in inches.

An example database entry for circular obstacles is shown in Table 4.2. Here, `tank1` is a circular object, and `c1` is the unique keyword for `tank1`. The QR code is placed at a radial distance of 6 units from the center, perpendicular to the radius, and the left side facing the center, as shown in Figure 4.7. The tank has a radius of 11 units, and a height of 7 units.

An example data entry for rectangular obstacles is shown in Table 4.3. Here, `machine1` is a rectangular object, and `r1` is the keyword for `machine1`. The QR code center is

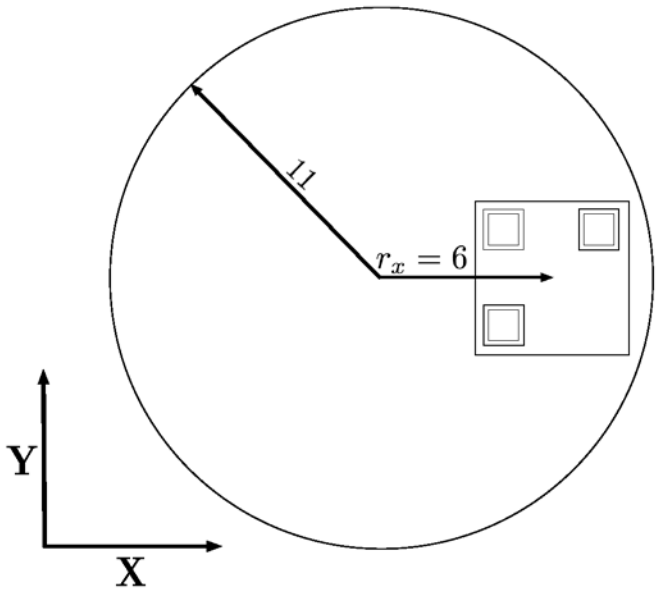


Figure 4.7: Labeling of a Cylindrical Obstacle

Table 4.3: Data Entry System for Rectangular Obstacles

machine1	r_x	r_y	length	width	height	rectangle
r1	10	9	21	17.25	15	

placed at a distance of (r_x, r_y) units from the origin, which in this case is the lower left vertex of the rectangle relative to the QR code. The longer side of the rectangle is chosen as the X-axis. The top, left, right and bottom of the QR code are identified from the position of the position blocks. The corner of the QR code without a position block is the lower right corner of the QR code. The QR code should be parallel to the edges and the lower left corner towards the origin, as shown in Figure 4.8. The length, width and height of `machine1` are 21, 17.25 and 15 units, respectively.

For all other polygons, including triangles, the data entry method is different, as shown in Table 4.4. Here, `machine2` is a polygon with 6 sides. The positions or the vertices relative to the QR code center are listed in polar coordinates, after selecting a convenient horizontal and vertical axis. In the example shown in Figure 4.9, point `pt1` is

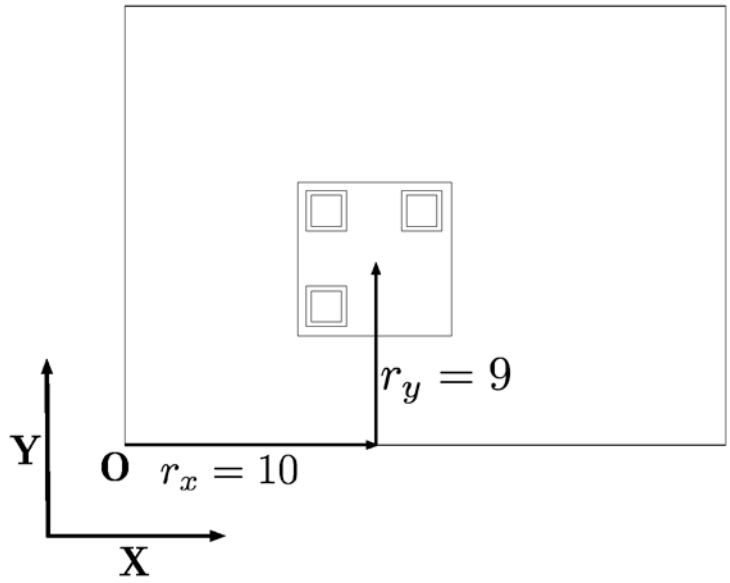


Figure 4.8: Labeling of a Rectangular Obstacle

Table 4.4: Data Entry System for Polygonal Obstacles

machine2	sides	pt1	pt2	pt3	pt4	pt5	pt6	height	polygon
p1	6	40 0°	30 5°	30 135°	40 180°	30 225°	0 315°	18	

40 units and 0 degrees from the QR code center, point `pt2` is 30 units and 45 degrees from the QR code center and so forth.

4.2.2 Encoding QR Codes

Every obstacle in the database is labeled with a QR code encoded with the unique keyword for it. Numerous tools are available for generating QR codes. A Python QR code package named qrcode 5.1 is used in this research [75]. While encoding, the version of the QR code, the level of error correction and the box size can be specified. Since the keyword is short, the level of error correction does not affect the data storage capacity or the size of the QR code, which allows any level of error correction to be chosen. However, since the payload blocks a part of the image, it is recommended to use the highest level of error correction for maximum chance of data retrieval from the QR

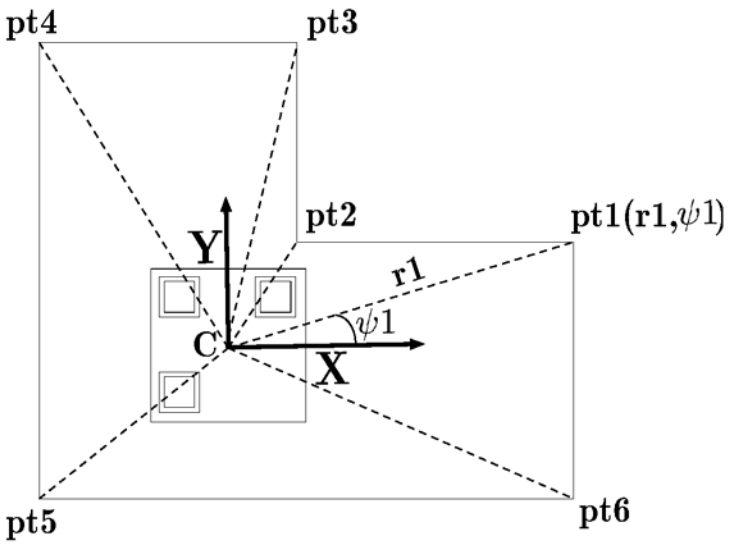


Figure 4.9: Labeling of a Polygonal Obstacle

codes.

QR code version 1 is used for this thesis, because it consists of the least number of rows and columns, allowing the size of the QR code to be large and readable from a long distance. The storage capacity of version 1 is enough for the keyword encoded in it, along with the maximum level of error correction. The box size parameter controls how many pixels each "box" of the QR code is. A higher box size is desired for a good-resolution QR code, allowing it to be printed at large sizes.

4.2.3 Decoding QR Codes

Every time an image of the entire workspace is generated, all the QR codes of the image are read using a QR code decoding library. In this research, the open-source ZBAR library [60] is used for decoding the QR codes. The decoding result, which is the keyword, is written to a text file, followed by the angle with respect to the horizontal and the center of the QR code. The center and angle information are used to calculate the position and orientation of the obstacles. A flow chart of the decoding process is shown in Figure 4.10. Figure 4.11 shows an example image with two obstacles labeled with

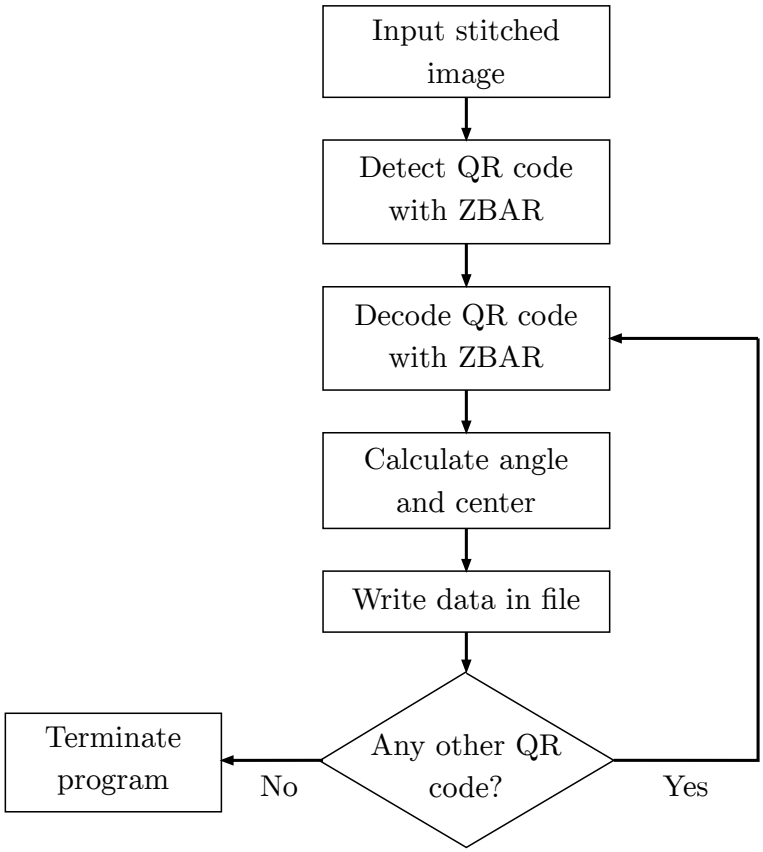


Figure 4.10: Flow Chart of the QR code Decoding Process

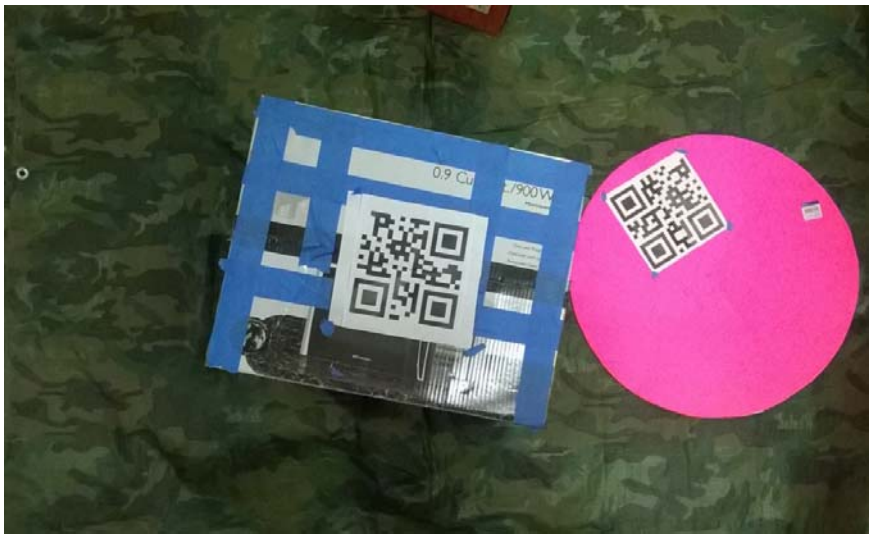


Figure 4.11: Example Image to be Decoded

QR codes. The decoding results are shown in Table 4.5.

Table 4.5: Result of the Decoded QR Codes from the Example Image

c1	237°	820	256
r1	189°	501	326

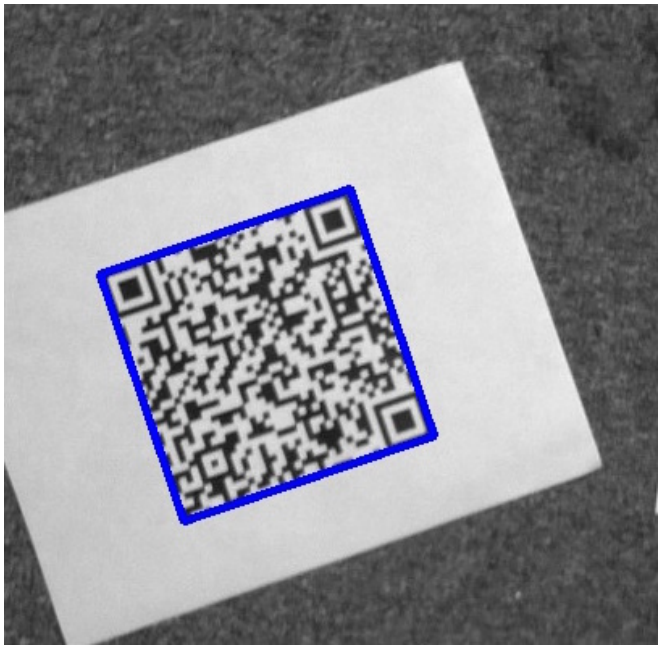


Figure 4.12: Minimum Enclosing Rectangle for finding the QR code center

In Table 4.5, `c1` is the keyword encoded in the QR code on `tank1`. The QR code's position is (820, 256) pixels, and it is at an angle of 237 degrees relative to the positive X-axis. Similarly, `r1` is the keyword encoded in the QR code on `machine1` and the QR code's position is (501, 326) pixels, and it is at an angle of 189 degrees relative to the positive X-axis.

4.2.4 Calculating the Center of QR Codes

For every QR code detected, a minimum rectangle enclosing the QR code is calculated, as shown in Figure 4.12. The four vertices of the rectangle are averaged to find the approximate center of the QR code.

4.2.5 Calculating the Angle of QR Codes

To calculate the QR code angle, the position blocks are first detected. The canny edge detection algorithm [76] is applied to detect all the edges of the image containing the QR code. Then, a contour finding algorithm is used to detect all the contours with hierarchy in the image. The position markers contain five nested contours, as shown in Figure 4.13. This distinguishes the position markers from the other modules of the QR code.

After the three position markers have been identified from the number of nested contours they contain, the relative position of them with respect to each other is determined, and the markers are named as the top, right, and bottom markers. This can be done by using a triangle ABC shown in Figure 4.14, formed by connecting the center of each of the three contours of the three position blocks. The vertex not involved in the largest side is out-lying, and it can be named as the top marker. In this case, C is the top marker. In order determine the right and the bottom marker from the remaining two, the slope of the straight line AB they form, and the distance of AB from C is calculated.

* if *slope* and *distance* are positive, A is BOTTOM and B is RIGHT

* if *slope* in negative and *distance* is positive, A is RIGHT and B is BOTTOM

* if *slope* in positive and *distance* is negative, A is RIGHT and B is BOTTOM

* if *slope* and *distance* are negative, A is BOTTOM and B is RIGHT

Once the top, right, and bottom markers are determined, the angle of the straight line connecting the top and right marker with respect to horizontal is the angle of the QR code with respect to horizontal.

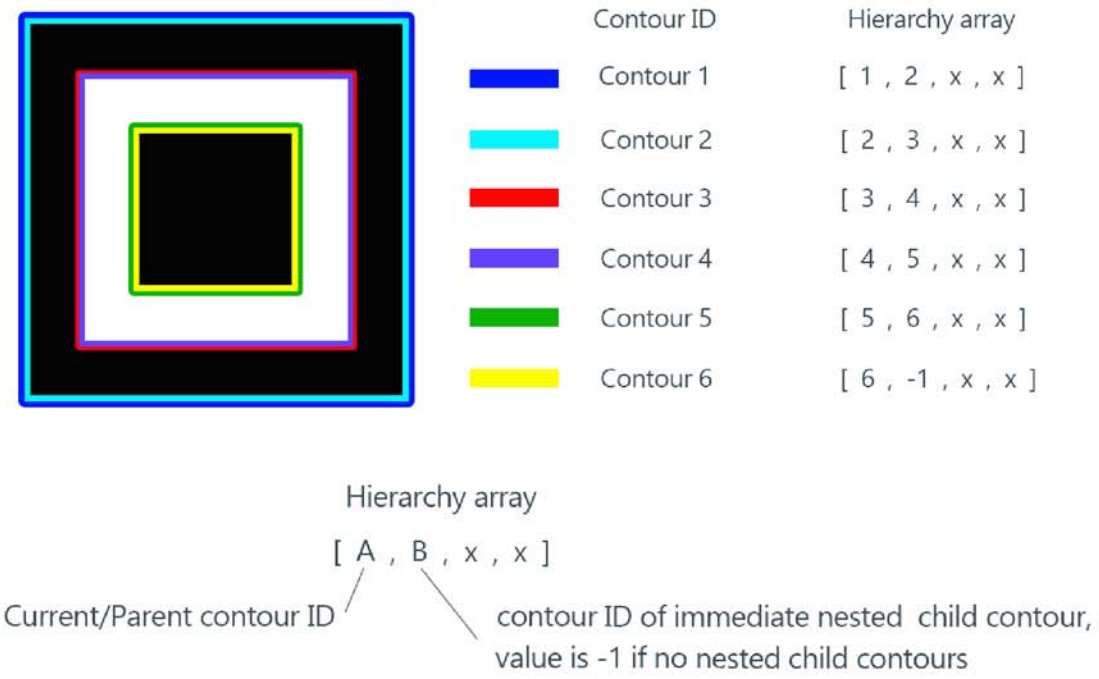


Figure 4.13: Detection of Position Marker from Number of Nested Contours [4]

4.2.6 Generating Maps from QR Code Data

To generate the map from the QR code data, first an empty map of the same size as the stitched image is created. Then, the text file where all the data for the decoded QR codes are saved is read. When a keyword is found, the database is searched for the keyword. Every keyword corresponds to a unique obstacle. When the keyword is found in the database, the shape and dimensions of the corresponding obstacle are read from the database. The center position and angle of the QR code are read from the data associated with the keyword in the QR data file. Then the conversion factor from inches to pixels is calculated from the height of the obstacle and used to draw the top face of the obstacle. If the obstacle height is not uniform, the highest point is assumed to be the obstacle height, and the conversion factor is calculated with that height. Then the obstacle position and orientation is calculated from the QR code position and orientation.

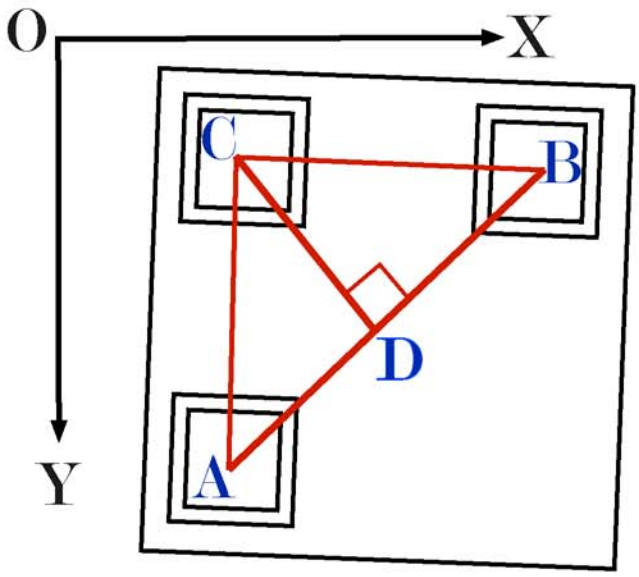


Figure 4.14: Detection of Top, Right and Bottom Markers for Angle Calculation

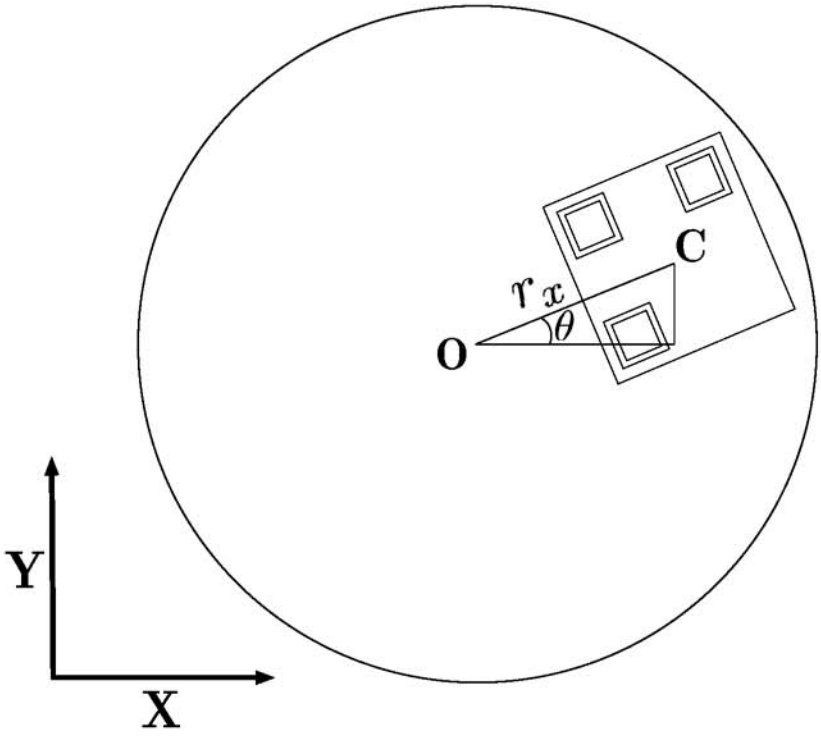


Figure 4.15: Determination of Cylindrical Obstacle Center From QR Code Position

Figure 4.15 shows how the center of the cylindrical obstacle is calculated from the QR code position and orientation. If the QR code center $C(x, y)$ is at a distance r_x from

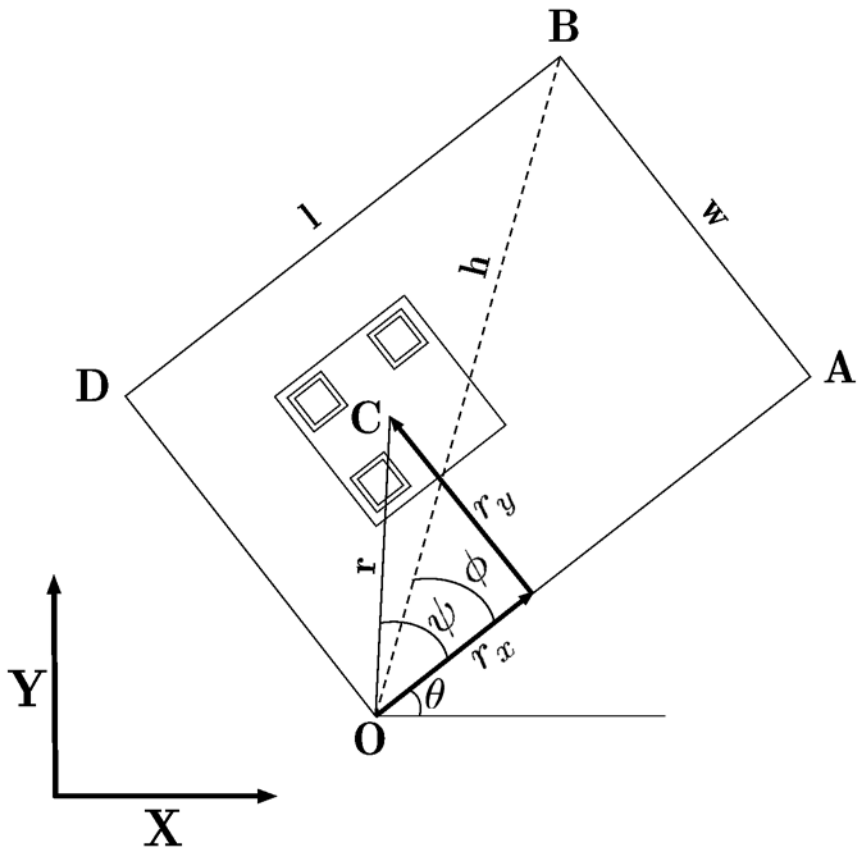


Figure 4.16: Determination of Rectangular Obstacle Vertices from QR Code Data

the center of the obstacle and perpendicular to the radius, and at an angle θ with the horizontal axis, then the obstacle center O is $(x - r_x \cos\theta,\ y - r_x \sin\theta)$. Once the position of the center of the obstacle is calculated, the obstacle can be drawn with the radius data read from the database file. The QR code can also be positioned at the center of the obstacle. In that case, r_x is zero.

To draw rectangular obstacles, all the vertices are calculated from the center position and orientation of the QR code. If the center of the QR code is $C(x, y)$ and it is at an angle of θ with the horizontal axis, then the vertices can be calculated as follows:

Vertex O:

$$x_o, y_o = x - r\cos(\psi + \theta),\ y - r\sin(\psi + \theta) \tag{4.1}$$

Vertex A:

$$x_a, y_a = x_o + l\cos\theta,\ y_o + l\sin\theta \tag{4.2}$$

Vertex B:

$$x_b, y_b = x_o + h\cos(\theta+\phi),\ y_o + h\sin(\theta+\phi) \tag{4.3}$$

Vertex D:

$$x_d, y_d = x_o - w\sin\theta,\ y_o + w\cos\theta \tag{4.4}$$

where,

$$r^2 = r_x^2 + r_y^2 \tag{4.5}$$

$$\psi = \tan^{-1}(r_y/r_x) \tag{4.6}$$

$$h^2 = l^2 + l^2 \tag{4.7}$$

and

$$\phi = \tan^{-1}(w/l) \tag{4.8}$$

Since in polygonal obstacles, the vertices are defined in polar coordinates relative to the QR code center, once the position of the QR code Center $C(x_o, y_o)$ and the QR code angle θ with respect to positive horizontal axis are read from the QR code data, the vertices can be calculated from the following formula:

$$(x, y)_i = x_o + r_i\cos(\psi_i+\theta),\ y_o + r_i\sin(\psi_i+\theta) \tag{4.9}$$

where r_i is the distance of i^{th} vertex from the QR code center, and ψ_i is the angle of the vertex with respect to the horizontal axis.

After all the vertices are found for the rectangle or polygon, it can be drawn by connecting the vertices with straight lines. Figure 4.18 shows an example workspace, a

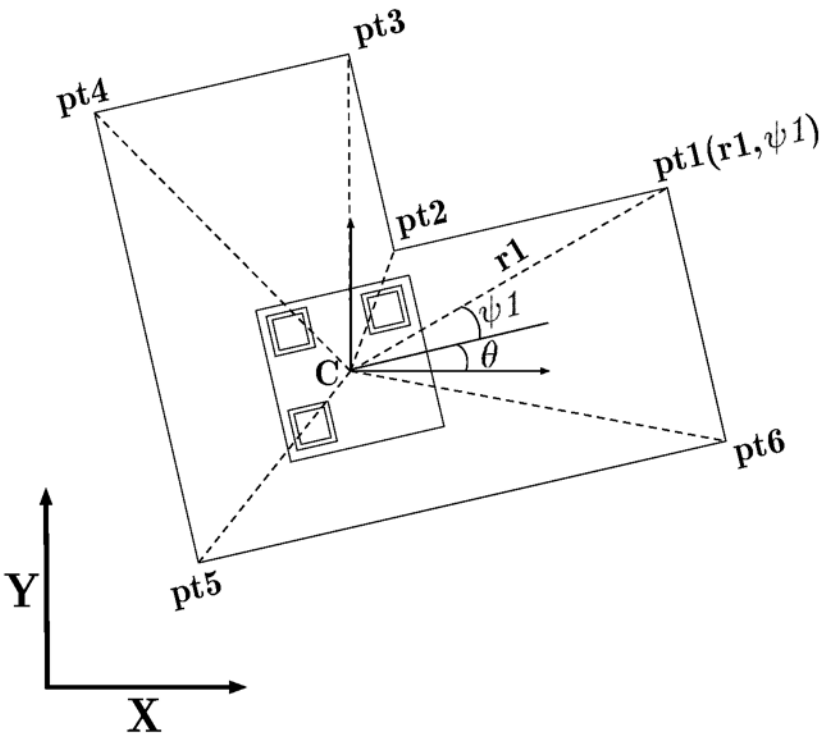


Figure 4.17: Determination of Polygonal Obstacle Vertices from QR Code Data

complete map is shown in Figure 4.19. Since the map is directly drawn from the dimensions specified in the database and the position and orientation of the obstacles, the accuracy of the map depends on the precision of measurements of the dimensions of the obstacles and the accuracy of placement of the QR codes. The distortion of the images captured by the camera, and the distortion of the stitched image also have a significant effect on the mapping accuracy.

4.2.7 Adding the Depth Information of the Obstacles

The map shown in Figure 4.19 shows only the top face of the obstacles. It is difficult to know the height and relative size of the obstacles from this map. It can be improved by adding the depth information of the obstacles. The height can be written in plain text on each obstacle, as shown in Figure 4.20. One way that a sense of depth of the obstacles can be given is by drawing the base of the obstacles. The obstacle bases in the map change size and color depending on their height. The base becomes smaller

Figure 4.18: Workspace with QR Code Labeled Obstacles

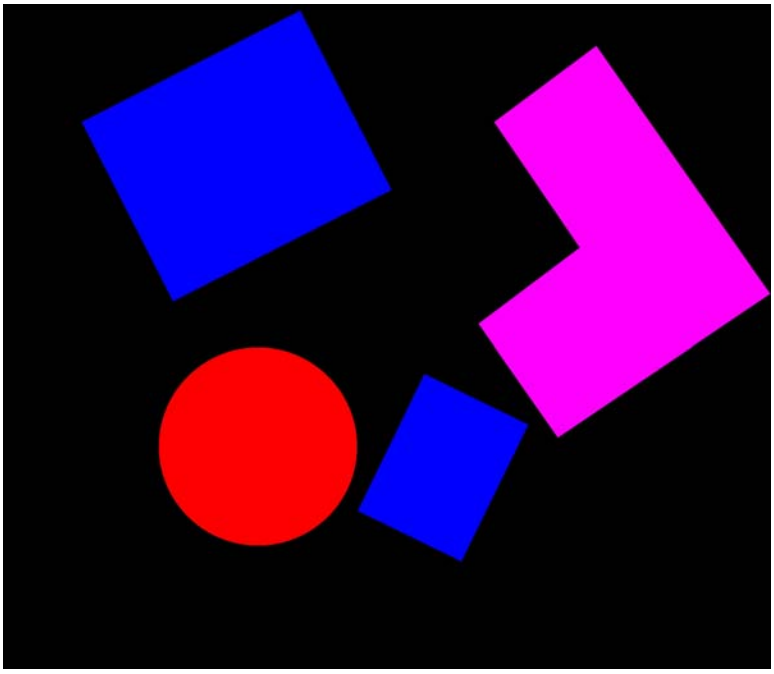


Figure 4.19: Resultant Map of the Workspace

and less bright with increasing height. The contrast and relative size between the top and the base give a sense of height of the obstacles.

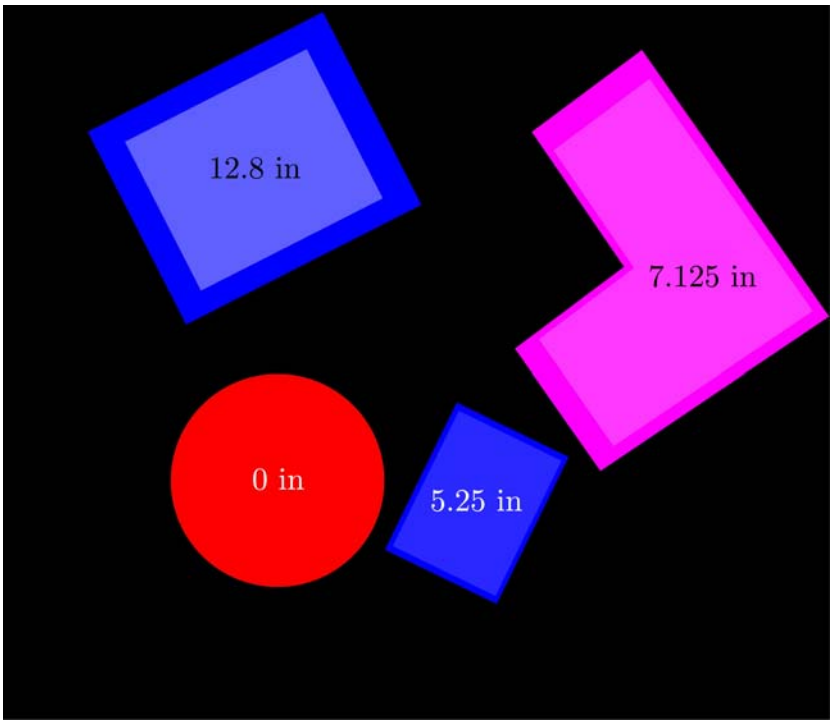


Figure 4.20: Workspace Map with Height Information

4.3 Combination with Machine-Vision-Based Mapping

The proposed QR-code-based method can be combined with the pure machine vision method introduced in Chapter III. A simple technique to combine these two maps is by drawing the QR code map directly over the machine-vision-based map. An example of this technique is shown in Figure 4.21, where the white areas behind the obstacles represent the machine-vision-based map. In this example, the QR code attached on the rectangular obstacle was slightly misplaced, so it shows slight discrepancy from the machine-vision-based map.

4.4 Performance Evaluation of QR Code-Based Mapping Algorithm

The performance of the mapping algorithm using QR codes was tested in different workspaces. Stitched images of these workspaces are shown in Figure 4.22. These workspaces have different backgrounds and different obstacle positions and spacing. The obstacles used for testing are of polygonal, rectangular, and circular shapes. The

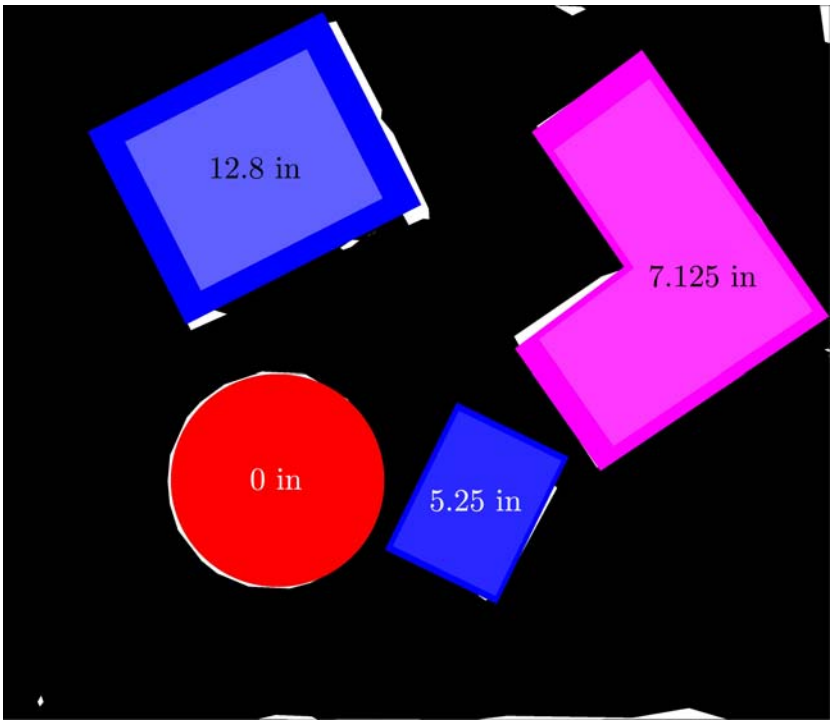


Figure 4.21: Workspace Map with Two Mapping Techniques Together

QR code maps are compared against manual maps drawn from the measurements of the workspaces. The coordinate of each vertex of each obstacle and the center in case of circular obstacles from an origin is measured, and the data are converted from inches to pixels. Then, the converted data are used to manually draw the top face of the obstacles.

Using this data, the total percentage of matched pixels, percentage of matched foreground pixels, percentage of missed obstacle pixels, and the percentage of falsely detected obstacle pixels are calculated. The percentage match plot shows what percentage of pixels of the manually-generated map match with the corresponding pixels of the QR code-based map, and the percentage foreground match plot shows what percentage of foreground pixels of the manually-generated map match with the corresponding pixels of the QR code-based map. These two plots indicate how good the map represents the actual workspace. The percentage missed obstacle pixels plot shows

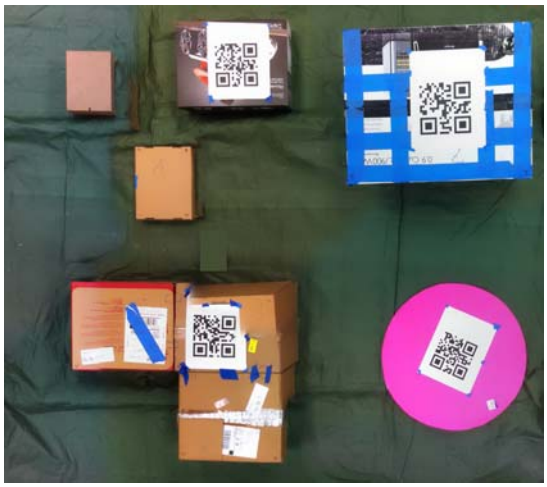
(a) Workspace 1 (b) Workspace 2 (c) Workspace 3

(d) Workspace 4 (e) Workspace 5 (f) Workspace 6

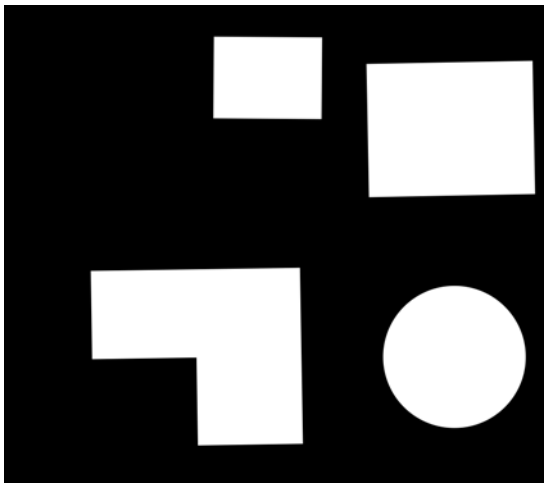
(g) Workspace 7

Figure 4.22: Workspaces Used for Evaluating QR Code Mapping Performance

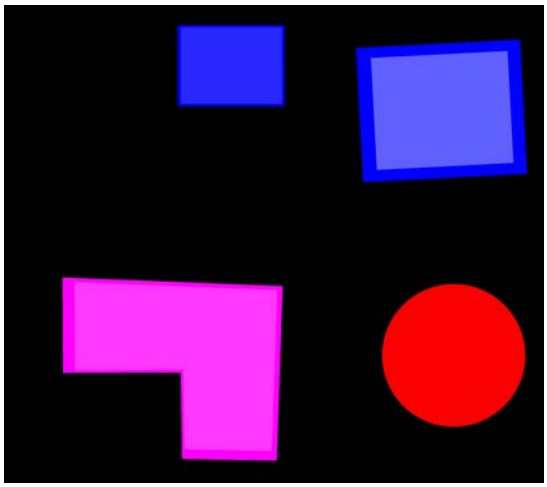
what percentage of foreground pixels in the manually-generated map are detected as a part of the background in the QR code-based map. The percentage falsely detected obstacles plot shows what percentage of background pixels in the manually-generated map are detected as a part of the foreground in the QR code-based map. Figure 4.23 shows an example workspace the corresponding manually-generated map and QR code map to be compared. Only the obstacles labeled with QR codes are considered for comparison. Some part of workspaces 6 and 7 are masked, because they are part of the



(a) Example Workspace



(b) Manually Drawn Map



(c) Map Using QR Code

Figure 4.23: Example Images for Evaluation

crane, not the workspace. The stitched image are of different sizes due to individual image resolution and stitching distortion differences. However, since the performance is evaluated as a percentage of total image pixels and total foreground pixels, the size variation doesn't affect the evaluation.

Figure 4.24 shows the percentage pixel match between the manually-generated map from the measurements of the workspace and the maps generated using QR codes. The results show as high as a 93% match between the manually-generated map and the QR code-based map. Figure 4.25 shows the percentage match of obstacle pixels of the QR

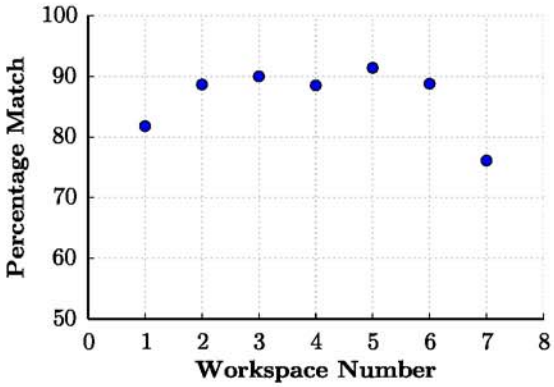


Figure 4.24: Percentage Match Between Actual Workspace and QR Code Map

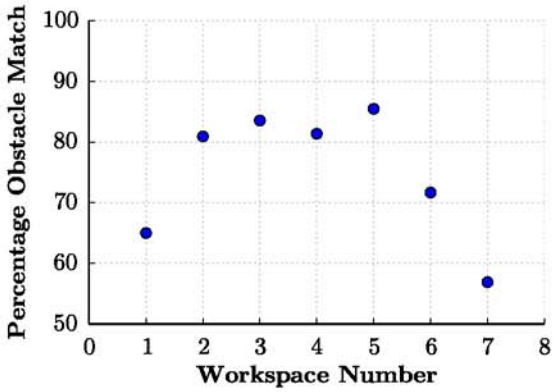


Figure 4.25: Percentage Obstacle Pixels Matched in QR Code Map

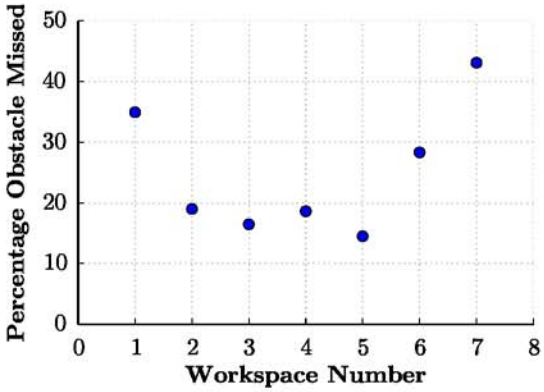


Figure 4.26: Percentage Obstacle Pixels missed in QR Code Map

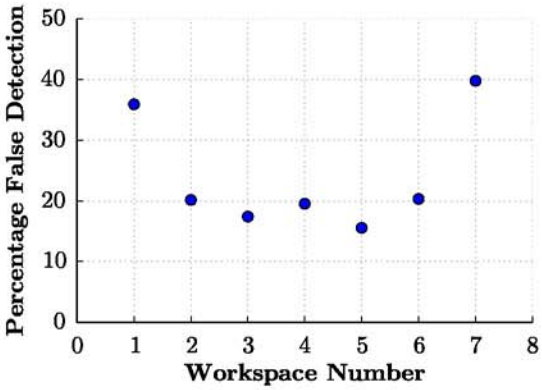


Figure 4.27: Percentage of Falsely Detected Obstacles in QR Code Map

code map compared to the manually-generated map. It shows the exact same pattern as Figure 4.24, which means the percentage match of obstacle pixels is almost proportional to the percentage match in total map. Figure 4.26 shows the percentage missed, and Figure 4.27 shows the percentage falsely detected obstacle pixels of the QR code map compared to the manually-generated map. From these two figures, it is clear that the percentage of missed obstacle pixels is proportional to the percentage of falsely detected obstacle pixels. This suggests that, if a map misses a high percentage of obstacles, then it also detects a high percentage of false obstacles. In all four figures it is noticeable that workspace 1 and workspace 7 have the highest amount of error. In these two cases there is a high distortion in stitching, resulting in a significant amount of de-

viation of the stitched image from the actual workspaces. Due to the distortion, the center and angle detection of the QR codes result in some errors, affecting the accuracy of the map.

4.5 Comparison of Mapping Techniques

The pure machine vision map, the QR code-based map, and a combination of these two maps are compared against the maps drawn from the physical measurements of the obstacles for the same workspaces shown in Figure 4.22. Obstacles both with and without QR codes are Considered for comparison in these workspaces. The percentage pixel match between actual map and manually-generated map is shown in Figure 4.28. The percentage match is close for the pure vision map, QR code-based map, and combination map in most cases, and there is no particular trend. However, Figure 4.29 shows that the percentage foreground match is higher for the combination map than the individual pure machine vision map and QR code map. The pure machine vision map shows slightly better match than the QR code map. The reason for that is that the obstacles without QR codes are not displayed in the QR code-based map, therefore only part of the actual workspace is shown in this map. This is also reflected on Figure 4.30 which compares the percentage obstacle pixels missed for the three kinds of map. This figure shows that the QR code-based map misses the most obstacle pixels. However, the previous section shows that the percentage missed pixels is reduced if the obstacles not labeled with QR codes are not considered for comparison. This figure also shows that the combination map misses fewer pixels that belong to an obstacle than the pure machine vision map and QR code-based map individually, which means the combination map is more reliable. However, Figure 4.31 shows that the combination map detects highest number of pixels as part of an obstacle where in reality they are not. It also shows that the pure vision map has the tendency of falsely detecting pixels as obstacles than the QR code-based map.

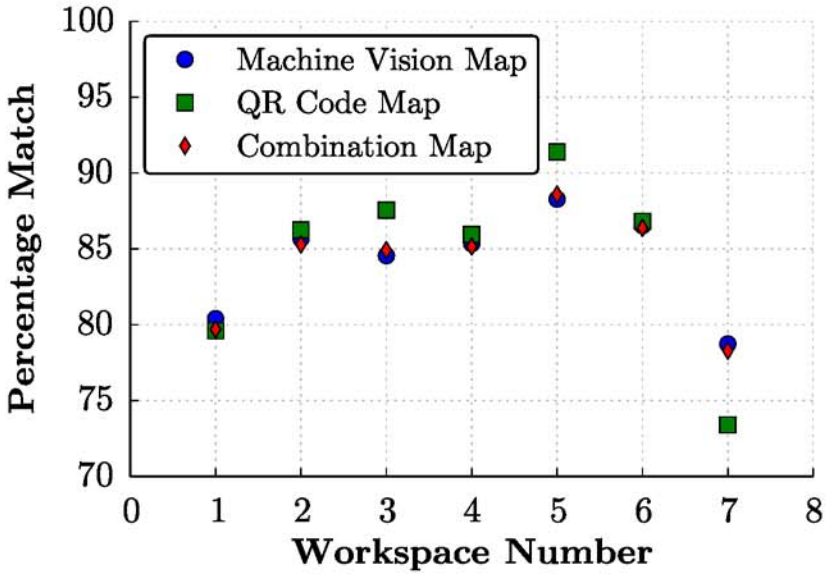


Figure 4.28: Percentage Pixel Match Between Actual Workspace and Generated Maps

For all three mapping techniques, errors are high for workspaces 1 and 7 because of the high distortion in stitched images. Another source of error is the assumption that each obstacle is viewed right from the top of the obstacles. The manual maps are drawn accordingly, by centering the obstacle top with the bottom. However, in reality that is not the case. The obstacles are viewed from an angle by the camera; therefore, the obstacle tops are skewed at different directions from the base.

It can be concluded from the analysis that, though the combination map has the tendency of falsely detecting greater number of pixels as obstacles, it detects the highest percentage of obstacle pixels and misses the lowest percentage of obstacle pixels. Falsely detecting obstacles is safer than missing obstacles in terms of safety in crane operation. Therefore, the combination map is more reliable, if not the most accurate, than the individual maps.

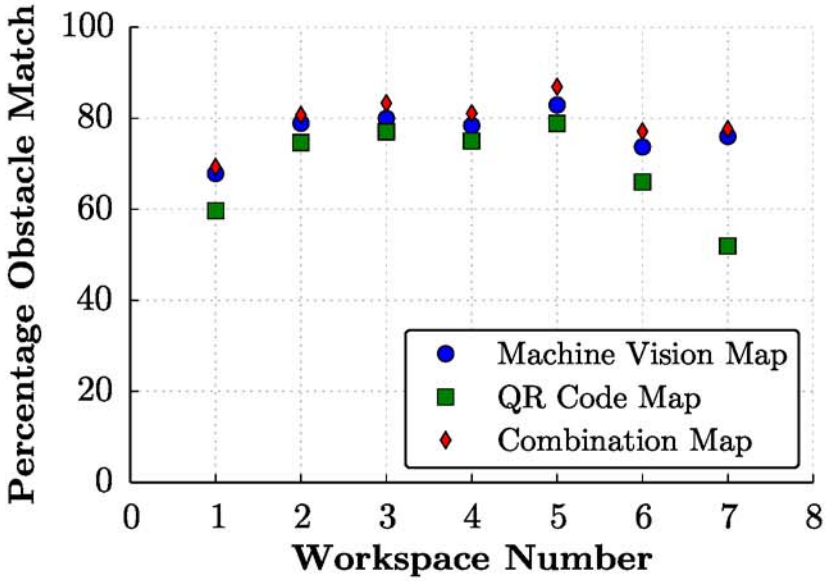


Figure 4.29: Percentage Obstacle Pixels Matched Between the Workspace and Generated Maps

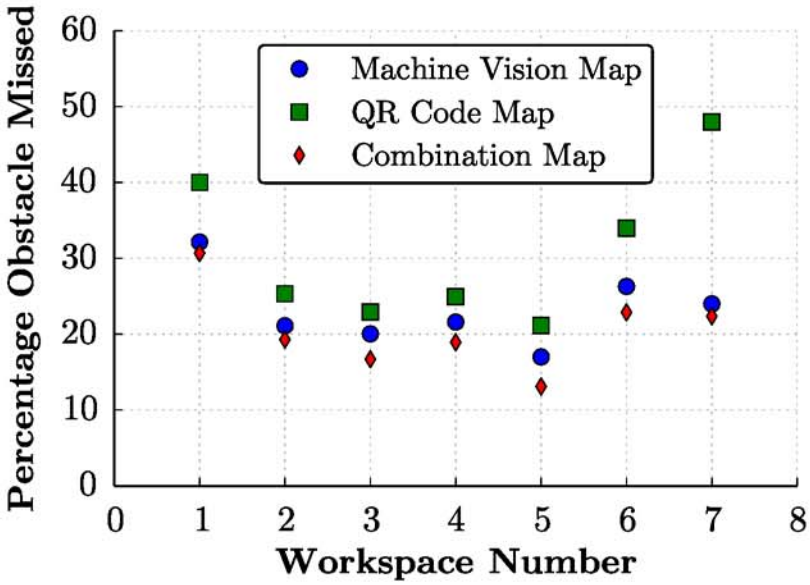


Figure 4.30: Percentage Obstacle Pixels Missed in Generated Maps

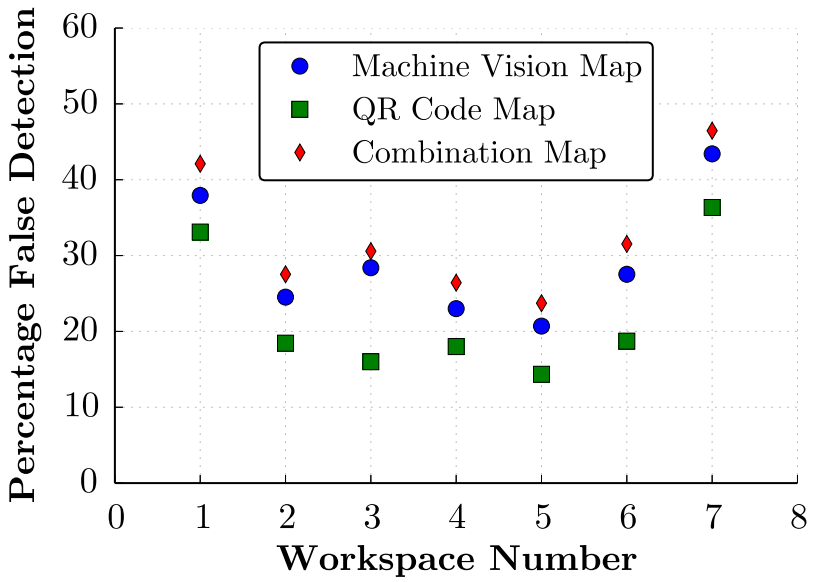


Figure 4.31: Percentage of Falsely Detected Obstacles in Generated Maps

4.6 Chapter Conclusion

In this chapter, an algorithm for mapping crane workspace using QR codes was introduced. All the steps of the mapping algorithm, including labeling of the obstacles with QR codes, creating database, encoding and decoding of the QR codes, angle and center calculation, and drawing the map were explained. The height of the obstacles was included in the drawn map. Then, the resultant map was integrated with the machine-vision-based map. The mapping performance of the QR code-based map was compared with the machine-vision-based map and the combination map. The efficiency of the QR code-based mapping algorithm was found to be as high as 93%. However, precise positioning of the QR codes and accurate measurements of the dimensions of the obstacles are important for optimum performance. If the QR codes are precisely positioned and clearly readable, and the measurements are accurate, the QR code-based mapping algorithm is preferable. However, it only works with QR code-labeled obstacles of known dimensions and therefore should be accompanied by the machine-vision-based mapping algorithm for a complete map of the crane workspace, unless all the

obstacles in the workspaces are labeled with a QR code.

Chapter 5

CONCLUSION

5.1 Summary and Contributions

In this thesis, first a novel approach of mapping the workspace of overhead cranes using pictures from a single camera was presented. Different image processing techniques used to accomplish the mapping, including image stitching, template matching, images segmentation, histogram calculation were discussed. The idea of displaying the older maps in the current map to show the previous obstacle positions as an indication of the likelihood of finding the obstacle at those positions again in the future was introduced. A memory factor was also introduced, which determines when to forget an old map. Two segmentation algorithms for obstacle detection, simple grayscale thresholding and watershed transform were compared. While the watershed transformation is more successful in detecting obstacles, it is also computationally expensive and the threshold values required to create the marker are difficult to determine.

In addition, a method of crane workspace mapping using QR codes was presented. The method for creating database, the labeling of the obstacles with QR codes, the encoding and decoding of the QR codes and the drawing of the obstacles on the map were discussed. The height information was included with the map to give it a 3D appearance, and the combination of the two mapping techniques were discussed. Then the performance of the mapping algorithm using QR codes was compared with the machine-vision-based map for different workspaces using a manually-generated map.

Image segmentation methods have applications in a number of different fields, such as

object detection and tracking. However, it has not been used in mapping crane workspaces before. This thesis shows the enormous potential of image segmentation methods to be used in mapping workspaces. Although image stitching techniques have been used before in aerial mapping, this is the first time the possibility of using image stitching together with image segmentation to formulate a crane workspace mapping algorithm has been explored.

Although QR codes were initially invented for storing product information, other applications of them have been introduced, such as using them as landmarks for localization of robots. This thesis presents an unique idea of using QR codes for mapping crane workspaces. The results obtained from this thesis demonstrate that QR codes have the potential of being widely used in mapping crane workspaces.

5.2 Future Work

The work in this thesis enables some advanced crane workspace mapping. More advanced image segmentation algorithms can be used to improve the mapping performance. Significant computation power can be saved by replacing the stitching with augmenting individual segmented images side-to-side. Instead of reading the QR codes from the stitched image, they could be read directly by the camera, and the obstacles could be drawn by locating the QR code in the camera coordinate combined with the position of the camera itself in the world coordinate.

The obstacle positions obtained from the map can be used for automatic obstacle avoidance by creating virtual boundaries around the obstacles, which in turn will make possible partial or fully-automated crane operation. This map can also be used to optimize the path for the crane, which will increase productivity by reducing travel distance and time.

BIBLIOGRAPHY

[1] [Online]. Available: http://www.cautructinnghia.com/

[2] [Online]. Available: http://www.cognex.com/products/machine-vision/in-sight-7000-series-integrated-vision-systems/

[3] Qrstuff. [Online]. Available:http://www.qrstuff.com/blog/2011/12/14/qr-code-error-correction

[4] Dsynflo. [Online]. Available:http://dsynflo.blogspot.com/2014/10/opencv-qr-code-detection-and-extraction.html

[5] B. of Labor Statistics. (2008, July) Crane-related occupational fatalities.

[6] Crane inspection and certification bureau. [Online]. Available:http://www.cicb.com/blog/posts/with-crane-related-injuries-on-the-rise-don-t-become-another-statistic.html

[7] O. J. M. Smith, *Feedback Control Systems*. New York: McGraw-Hill Book Co., Inc., 1958.

[8] N. C. Singer and W. P. Seering, "Preshaping command inputs to reduce system vibration," *Journal of Dynamic Systems, Measurement, and Control*, vol. Volume 112, pp. 76–82, March 1990.

[9] W. Singhose, N. Singer, and W. Seering, "Time-optimal negative input shapers," *Journal of Dynamic Systems, Measurement, and Control,*, vol. 119, pp. 198–205, June 1997.

[10] N. Singer, W. Singhose, and E. Kriikku, "An input shaping controller enabling cranes to move without sway," in *ANS 7th Topical Meeting on Robotics and Remote Systems*, vol. 1, Augusta, GA, 1997, pp. 225–31.

[11] K. Sorensen, W. Singhose, and S. Dickerson, "A controller enabling precise positioning and sway reduction in bridge and gantry cranes," *Control Engineering Practice*, vol. 15, no. 7, pp. 825–837, July 2007.

[12] A. Khalid, W. Singhose, J. Huey, J. Lawrence, and D. Frakes, "Study of operator behavior, learning, and performance using an input-shaped bridge crane," in *Proceedings of the 2004 IEEE International Conference on Control Applications*, vol. 1, September 2004, pp. 759–764.

[13] D. Kim and W. Singhose, "Performance studies of human operators driving double-pendulum bridge cranes," *Control Engineering Practice*, vol. 18, no. 6, pp. 567–576, June 2010.

[14] Camotion, inc. [Online]. Available: http://www.camotion.com

[15] [Online]. Available: http://www.konecranes.com/equipment/overhead-cranes/smart-features

[16] C. Kim, C. T. Haas, K. A. Liapi, and C. H. Caldas, "Human-assisted obstacle avoidance system using 3d workspace modeling for construction equipment operation," *Journal of computing in civil engineering*, vol. 20, no. 3, pp. 177–186, May 2006.

[17] D. Murray and C. Jennings, "Stereo vision based mapping and navigation for mobile robots," in *Proceedings of the IEEE International Conference on Robotics And Automation*, vol. 2, April 1997, pp. 1694–1699.

[18] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *EEE Transactions on Robotics*, vol. 30, no. 1, February 2014.

[19] C. Stiller, J. Hipp, C. Rossig, and A. Ewald, "Multisensor obstacle detection and tracking," *Image and Vision Computing*, vol. 18, no. 5, pp. 389–396, April 2000.

[20] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, November 2007, pp. 225–234.

[21] ——, "Parallel tracking and mapping on a camera phone," in *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality*, October 2009, pp. 83–86.

[22] S. Thrun and J. Leonard, *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.

[23] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 229-241, June 2001.

[24] A. Shapira, Y. Rosenfeld, and I. Mizrahi, "Vision system for tower cranes," *Journal of Construction Engineering and Management*, vol. 134, no. 5, pp. 320–332, May 2008.

[25] J. Yang, P. Vela, J. Teizer, and Z. Shi, "Vision-based tower crane tracking for understanding construction activity," *Journal of Computing in Civil Engineering*, vol. 28, no. 1, pp. 103–112, January 2014.

[26] Y. Yoshida and K. Tsuzuki, "Visual tracking and control of a moving overhead crane load," *9th IEEE International Workshop on Advanced Motion Control*, pp. 630–635, 2006.

[27] T. Miyoshi, K. Tsuchida, and K. Terashima, "Obstacle avoidance control for overhead crane with rotary motion of load," in *SICE Annual Conference in Fukui*, August 2003.

[28] S. Nagai, A. Kaneshige, and S. Ueki, "Three-dimensional obstacle avoidance online path-planning method for autonomous mobile overhead crane," in *International Conference on Mechatronics and Automation (ICMA)*, August 2011, pp. 1497–1502.

[29] C. Yuan Been, C. Oscal T-C *et al.*, "Image segmentation method using thresholds automatically determined from picture contents," *EURASIP Journal on Image and Video Processing*, 2009.

[30] O. J. Tobias and R. Seara, "Image segmentation by histogram thresholding using fuzzy sets," *Image Processing, IEEE Transactions on*, vol. 11, no. 12, pp. 1457–1465, Dec 2002.

[31] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using kalman-filtering," in *Proceedings of International Conference on recent Advances in Mechatronics*. Citeseer, 1995, pp. 193–199.

[32] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 11, pp. 1101–1113, 1993.

[33] A. E. Savakis, "Adaptive document image thresholding using foreground and background clustering," in *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*. IEEE, 1998, pp. 785–789.

[34] T. Uemura, G. Koutaki, and K. Uchimura, "Image segmentation based on edge detection using boundary code," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 10, pp. 6073–6083, 2011.

[35] R. Adams and L. Bischof, "Seeded region growing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 6, pp. 641–647, Jun 1994.

[36] F. Y. Shih and S. Cheng, "Automatic seeded region growing for color image segmentation," *Image and Vision Computing*, vol. 23, no. 10, pp. 877–886, Sepetember 2005.

[37] A. Mehnert and P. Jackway, "An improved seeded region growing algorithm," *Pattern Recognition Letters*, vol. 18, no. 10, pp. 1065–1071, 1997.

[38] S. Beucher, *SCANNING MICROSCOPY-SUPPLEMENT*, 1992.

[39] S. Beucher and C. Lantuéjoul, "Use of watersheds in contour detection," 1976.

[40] A. Hanbury, "Image segmentation by region based and watershed algorithms," *Wiley Encyclopedia of Computer Science and Engineering*, 2008.

[41] M. Gonzalez and V. Ballarin, "Automatic marker determination algorithm for watershed segmentation using clustering," *Latin American applied research*, vol. 39, no. 3, pp. 225–229, 2009.

[42] I. Pratikakis, I. Vanhamel, H. Sahli, B. Gatos, and S. Perantonis, "Unsupervised watershed-driven region-based image retrieval," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 153, no. 3, pp. 313–322, June 2006.

[43] M. Meng, L. Fan, and L. Liu, "A comparative evaluation of foreground/background sketch-based mesh segmentation algorithms," *Computers & Graphics*, vol. 35, no. 3, pp. 650–660, 2011.

[44] J. Pont-Tuset and F. Marques, "Measures and meta-measures for the supervised evaluation of image segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, IEEE. IEEE, 2013, pp. 2131–2138.

[45] M. Borsotti, P. Campadelli, and R. Schettini, "Quantitative evaluation of color image segmentation results," *Pattern recognition letters*, vol. 19, no. 8, pp. 741–747, 1998.

[46] Y. J. Zhang, "Evaluation and comparison of different segmentation algorithms," *Pattern recognition letters*, vol. 18, no. 10, pp. 963–974, 1997.

[47] ——, "A survey on evaluation methods for image segmentation," *Pattern recognition*, vol. 29, no. 8, pp. 1335–1346, 1996.

[48] H. Kobayashi, "A new proposal for self-localization of mobile robot by self-contained 2d barcode landmark," in *SICE Annual Conference*. Akita University, Akita, Japan: SICE, August 20-23 2012.

[49] G. Lin and X. Chen, "A robot indoor position and orientation method based on 2d barcode landmark," *Journal of Computers*, vol. 6, no. 6, June 2011.

[50] T. Suriyon, H. Keisuke, and B. Choompol, "Development of guide robot by using qr code recognition," in *The Second TSME International Conference on Mechanical Engineering*, vol. 21, 2011.

[51] J. tung Wang, C.-N. Shyi, T.-W. Hou, and C. Fong, "Design and implementation of augmented reality system collaborating with qr code," in *Computer Symposium (ICS), 2010 International*, Dec 2010, pp. 414–418.

[52] H.-W. Liu and Y. Yan, "Recognition and decoding of qr code," *Jisuanji Gongcheng yu Sheji(Computer Engineering and Design)*, vol. 26, no. 6, pp. 1560–1562, 2005.

[53] L. Zhao-lai, H. Ting-lei, W. Rui, and Z. Xiao-yan, "A method of image analysis for qr code recognition," in *Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on*. IEEE, 2010, pp. 250–253.

[54] Y. Kato, D. Deguchi, T. Takahashi, I. Ide, and H. Murase, "Low resolution qr-code recognition by applying super-resolution using the property of qr-codes," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, Sept 2011, pp. 992–996.

[55] A. Sun, Y. Sun, and C. Liu, "The qr-code reorganization in illegible snapshots taken by mobile phones," in *Computational Science and its Applications, 2007. ICCSA 2007. International Conference on.* IEEE, 2007, pp. 532–538.

[56] Y. Liu, J. Yang, and M. Liu, "Recognition of qr code with mobile phones," in *Control and Decision Conference, 2008. CCDC 2008. Chinese.* IEEE, 2008, pp. 203–206.

[57] Y.-H. Chang, C.-H. Chu, and M.-S. Chen, "A general scheme for extracting qr code from a non-uniform background in camera phones and applications," in *Multimedia, 2007. ISM 2007. Ninth IEEE International Symposium on.* IEEE, 2007, pp. 123–130.

[58] L. F. Belussi and N. S. Hirata, "Fast qr code detection in arbitrarily acquired images," in *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on.* IEEE, 2011, pp. 281–288.

[59] [Online]. Available: http://opencv.org/

[60] Zbar. [Online]. Available: http://zbar.sourceforge.net/

[61] M. S. Rahman and J. Vaughan, "Simple near-realtime crane workspace mapping using machine vision," in *Proceedings of Dynamic Systems and Control Conference.* San Antonio, TX: ASME, October 2014.

[62] A. Saalfeld, "Topologically consistent line simplification with the douglas-peucker algorithm," *Cartography and Geographic Information Science*, vol. 26, no. 1, pp. 7–18, 1999.

[63] F. Jurie and M. Dhome, "A simple and efficient template matching algorithm," in *Proceedings of the Eight IEEE International Conference on Computer Vision*, vol. 2, 2001, pp. 544–549.

[64] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007.

[65] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

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

[67] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *Vision algorithms: theory and practice*. Springer, 2000, pp. 298–372.

[68] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1101–1113, August 2002.

[69] S. Zhu and A. Yuille, "Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, pp. 884–900, September 1996.

[70] Q. Huang, B. Dom, D. Steele, J. Ashley, and W. . Niblack, "Foreground/background segmentation of color images by integration of multiple cues," *International Conference on Image Processing*, vol. 1, pp. 23–26, October 1995.

[71] L. J. Belard and W. Mourou, "Image segmentation: A watershed transformation algorithm," *Image Analysis and Stereology*, vol. 28, no. 2, pp. 93–102, 2011.

[72] J. F. Barrera, A. Mira, and R. Torroba, "Optical encryption and qr codes: secure and noise-free information retrieval," *Optics express*, vol. 21, no. 5, pp. 5373–5378, 2013.

[73] Denso. Qr code essentials.

[74] M. Sudan, "Decoding of reed solomon codes beyond the error-correction bound," *Journal of complexity*, vol. 13, no. 1, pp. 180–193, 1997.

[75] qrcode 5.1. [Online]. Available: https://pypi.python.org/pypi/qrcode

[76] J. F. Canny, "Finding edges and lines in images," *Massachusetts Inst. of Tech. Report*, vol. 1, 1983.

Rahman, Mohammad Sazzad. Bachelor of Science, Bangladesh University of Engineering and Technology, Spring 2011; Master of Science, University of Louisiana at Lafayette, Spring 2015
Major: Mechanical Engineering
Title of Thesis: Machine Vision Techniques for Crane Workspace Mapping
Committee Chair: Dr. Joshua Vaughan
Pages in Thesis: 96; Words in Abstract: 242

ABSTRACT

Cranes are used worldwide for transportation and material handling in a variety of industries and facilities, including manufacturing industries, shipyards, and warehouses. Safety and efficiency in crane operations are a concern, since these issues are closely related to productivity. One of the reasons for crane-related accidents is mistakes by the operator, some of which can be attributed to the limitations of the operator?s field of view, depth perception, and knowledge of the workspace. These limitations are exacerbated by the dynamic environment of the workspace. One possible solution to these problems could be aiding the operator with a dynamic map of the workspace that shows the position of obstacles within it. In this thesis, two methods for mapping the crane workspace in near-realtime using computer vision are introduced. Several computer vision algorithms are integrated, and new techniques are introduced to generate a machine-vision-based map. A QR code-based mapping algorithm is also formulated. The algorithms can work independently. However, they can also be integrated, and the results show that a combination of these two mapping techniques produce the best results. The success of the pure machine-vision-based map and the QR code-based map depend on successful segmentation of color regions and detection of the QR codes, respectively. The combination of the two algorithms is a novel approach that ensures maximum obstacle detection. The algorithms produce a workspace map that can help the crane operator drive the crane more safely and efficiently.

BIOGRAPHICAL SKETCH

Mohammad Sazzad Rahman was born and grew up in Chittagong, at the south-eastern part of Bangladesh. He is the second child of his mother, Hasina Begum. He attended Bangladesh University of Engineering and Technology, Dhaka where he earned a Bachelor of Science in Mechanical Engineering. He began his graduate studies in Mechanical Engineering at the University of Louisiana at Lafayette in Fall 2013. His research interests are Controls and Robotics.

Machine Vision Techniques for Crane Workspace Mapping

A Thesis

Presented to the

Graduate Faculty of the

University of Louisiana at Lafayette

In Partial Fulfillment of the

Requirements for the Degree

Master of Science

Mohammad Sazzad Rahman

Spring 2015

Machine Vision Techniques for Crane Workspace Mapping

Mohammad Sazzad Rahman

APPROVED:

Joshua Vaughan Assistant Professor of Mechanical Engineering	Ahmed Khattab Associate Professor of Industrial Technology & Mechanical Engineering
Charles Taylor Assistant Professor of Mechanical Engineering	Mary Farmer-Kaiser Dean of the Graduate School

DEDICATION

To my mother,

Hasina Begum,

for endless inspiration

ACKNOWLEDGMENTS

I would like to acknowledge the following:

- My mother, for all the sacrifices she made for me throughout my entire life. I would not be able to pursue a masters degree if it was not for her.
- My sisters, for their immense support.
- My advisor, Dr. Joshua Vaughan, for watching my back all the time during my stay at Lafayette, and for teaching me the proper way of doing research and writing scientific documents.
- My committee members, Dr. Charles Taylor and Dr. Ahmed Khattab, for agreeing to be in my thesis committee.
- My adoptive mother, Kara Murphy, father, Harry Murphy, sister, Colette and brother, Alex, for their unconditional love for me. It means so much having a family on the other side of the planet from home.
- My lab-mates, Bobby, Dare, Seema, Nicole, Yasmeen, Jasmin, Jordan, and Beau, for their excellent co-operation. It was a great pleasure for me to work with them.
- My friends in Lafayette, for their company. There are too many to name. Special thanks to the Bangladeshi community in Lafayette, and my Bangladeshi friends who live in the US, for making my life in Lafayette easier.
- Last but not the least, all the faculty and staff of the Mechanical Engineering Department at the University of Louisiana at Lafayette, especially Dr. Sally Mclnerny for the financial support I received from the department.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Cranes are used worldwide for transportation and material handling in a variety of industries and facilities, including manufacturing industries, shipyards, and warehouses. Safety and efficiency in crane operations are a concern, since these issues are closely related to productivity. One of the reasons for crane-related accidents is mistakes by the operator, some of which can be attributed to the limitations of the operator's field of view, depth perception, and knowledge of the workspace. These limitations are exacerbated by the dynamic environment of the workspace. One possible solution to these problems could be aiding the operator with a dynamic map of the workspace that shows the position of obstacles within it. In this thesis, two methods for mapping the crane workspace in near-realtime using computer vision are introduced. Several computer vision algorithms are integrated, and new techniques are introduced to generate a machine-vision-based map. A QR code-based mapping algorithm is also formulated. The algorithms can work independently. However, they can also be integrated, and the results show that a combination of these two mapping techniques produce the best results. The success of the pure machine-vision-based map and the QR code-based map depend on successful segmentation of color regions and detection of the QR codes, respectively. The combination of the two algorithms is a novel approach that ensures maximum obstacle detection. The algorithms produce a workspace map that can help the crane operator drive the crane more safely and efficiently.

Chapter 1

INTRODUCTION

1.1 Problem Statement

In overhead bridge cranes, like the one shown in Figure 1.1, a payload is suspended from a trolley that moves along a bridge. The bridge itself can also move, enabling the crane to serve a large area. Cranes have been used in construction and material handling since antiquity and have made great contributions to the progress of civilization. However, there are some risks in crane operation. One of the most dangerous risks in crane operation is payload oscillation. When the crane is moved through its workspace, the payload oscillates. If the oscillation is not controlled, the payload may collide with objects or people, causing damage and injury. According to the Bureau of Labor statistics, there were 78 crane-related fatal injuries per year from 2003 to 2005 [5]. These incidents occurred either directly or indirectly because of the crane, or the operator. According to the Crane Inspection and Certification Bureau, 90% of all crane accidents occur due to human error [6]. Half of U.S. crane accidents that had injuries in 2009 resulted in fatalities.

Control techniques, including both feed-forward and feedback methods, have been invented to eliminate crane payload oscillation. One of the most successful crane control techniques is input shaping [7–9], which shapes the command by convolving a sequence of impulses with crane operator input. The convolved signal is then used as the reference command. This technique has proven useful for vibration reduction of cranes [10–13].

Even though the control techniques used for vibration reduction greatly reduce the risk

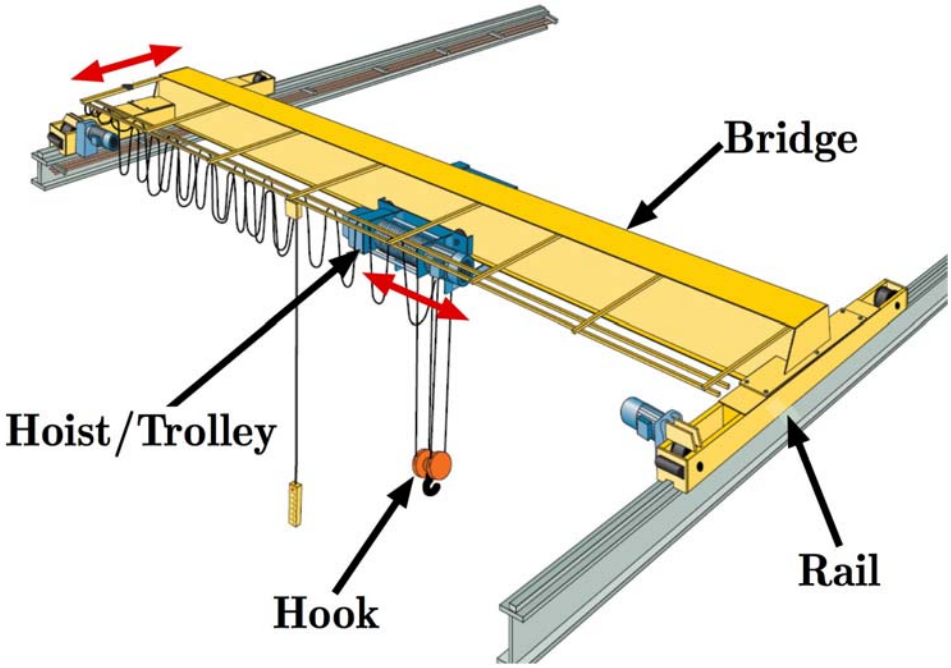


Figure 1.1: Overhead Crane [1]

of accidents from payload oscillation, there are still other kinds of risks associated with crane operation. Mistakes by the operator resulting from poor knowledge of the entire workspace or the operator's vision being blocked by the payload or obstacles are just two such risks. Either of these problems can lead to unsafe conditions and collisions with workspace obstacles.

To prevent payload collision with obstacles, companies like CAMotion [14] have implemented virtual boundaries around obstacles in crane workspace. However, obstacle boundaries typically have to be manually entered either through a programming interface or a series of offline obstacle-identification steps. Normal crane operations must be stopped when these obstacle boundaries are being set up. This can work well for permanent obstacles, such as big machines, tanks and assembly lines. However, they are not suitable for dynamic obstacles, such as fork lifts, loading and unloading zones, and temporary machines.

Another company, named KONECRANE [15], has implemented user-defined protected areas. The crane automatically slows down and stops, and the operator is alarmed by the user-interface when the crane goes near a protected area. However, these techniques are not very useful for obstacles that change positions frequently, which is very common in crane workspaces. A robust automated workspace mapping and obstacle-identification method is obviously preferable. This thesis is the first step to an automated crane workspace mapping using machine-vision techniques.

In this thesis, two novel methods of crane workspace mapping using computer vision is proposed. The first algorithm uses only one camera mounted on the crane trolley. The map updates automatically as the crane moves through the workspace during normal operation, so the operator always has up-to-date knowledge of the workspace. The map could be particularly helpful when the operator's view is partially blocked by the payload or obstacles, or a new operator starts working and finds obstacles in new positions. Using the automatically generated map, virtual boundaries could be defined around the obstacles, to prevent payloads from running into them. These boundaries could be updated every time the map updates. A second technique for mapping using Quick Response (QR) codes is introduced, which uses the same camera and can either work individually or along with the first method. Together, these two methods produce a complete two-dimensional map of the crane workspace showing the most recently known positions of the obstacles.

1.2 Background and Literature Review

Machine-vision-based mapping or obstacle avoidance is not a totally new idea. Research has been conducted to model the workspace in order to avoid crushing obstacles during operations of heavy equipment by modeling the objects in 3D [16]. In addition, vision systems have been used for mapping and navigation for mobile robots [17–19] and developing augmented reality workspaces [20, 21]. Mapping and navigation were

carried out using robots equipped with sophisticated sensors such as stereo-vision cameras, RGB-D cameras, or a combination of multiple sensors. These techniques are quite expensive and can be difficult to implement in overhead cranes, where navigation is not a concern. In addition, a small number of sensors is preferred.

Simultaneous Localization and Mapping (SLAM) is a well established method for mapping an unknown environment or updating a map within a known environment by mobile robots and autonomous vehicles while they keep track of their current location [22, 23]. In the case of crane workspaces, the precise position of the crane is known, so a simpler algorithm can be applied to map the workspace.

Machine vision systems have been implemented on tower cranes [24, 25] where a camera is mounted on the crane facing downwards to get a bird's eye view of the site. The sole purpose was to help the operator see the workspace clearly. No mapping was attempted in this research, so it is difficult for the operator to locate the obstacles while driving the crane. Some researchers have used machine vision as a means of measuring the crane hook location for feedback control [26], and some have conducted research into crane control for obstacle avoidance [27, 28]. However, in this work, there was no attempt to map the crane workspace.

A critical part of the machine vision technique for mapping is image segmentation. The accuracy of the algorithm presented in this thesis heavily depends on how well the obstacles are separated from the background. A simple method for image segmentation is using thresholds. Some researchers worked on calculating thresholds automatically from picture contents [29], using fuzzy sets [30], or Kalman-Filtering [31]. Data clustering methods have also been applied in image segmentation [32, 33]. Edge-based and region-based algorithms are two of the most common types of image segmentation techniques. Edge-based algorithms detect the sharp changes in an image by de-

ciding whether pixels belong to an edge or not [34]. These algorithms work well with simple and grayscale images with very few features [34], and therefore are not suitable for segmenting a complex image taken of a crane workspace. Region-based algorithms are pixel-based segmentation methods because they require initial seed points to be selected [35]. This algorithms use data clustering to decide whether a neighboring pixel belongs to the seeded region or not and keep iterating until all the pixels in the image are examined. Region-based algorithms always give closed contours, which is important for the algorithm presented in this research, because the obstacles are drawn in the map based on the contours.

For region-based algorithms, the image should be properly seeded. Every seed leads to a segmented region in the image. Researchers have tried to make the seeding automated [36, 37]; however, these methods separate one region from another. The mapping technique presented in this thesis requires segmenting all the objects from one common background. The decision of whether a region belongs to the background or not plays a significant role in the success of the algorithm presented in this thesis.

One of the most-used seeded image segmentation algorithms is the watershed transformation [38, 39], which is a combination of both edge-based and region-based algorithms [40]. This algorithm requires markers in every region to be segmented. The markers act as seeds and can be manually entered by the user. To make the watershed transformation unsupervised, the markers should be determined automatically. Researchers have been working on creating automated markers using clustering and morphological operations [41, 42]. However, these methods are application specific, and none of them exactly apply to the segmentation required for crane workspace mapping. The methods treat every region as a separate object, instead of segmenting all the objects from a common background.

To measure the effectiveness of the machine-vision-based mapping, the efficiency of the segmentation method used should be evaluated. Different kinds of segmentation evaluation methods are available [43–46]. The evaluation methods can be divided into two categories: analytical and empirical [47]. The analytical methods measure the goodness of an algorithm by analyzing the algorithm itself and its internal parameters. The empirical methods, which are more suitable for this thesis, use test images as standards and compare them with the segmented images to measure their accuracy.

Empirical methods can be further divided into two subcategories: empirical goodness and empirical discrepancy methods [47]. Empirical goodness is the measure of percentage match between the segmented image and the reference image, and empirical discrepancy is the measure of percentage discrepancy between the segmented image and the reference image. These two methods help assess the mapping performance by comparing the similarity and discrepancy between the map and the workspace. Both are used in this thesis.

Although the idea of workspace mapping using QR codes is new, the innovative and intelligent use of QR codes is not. QR codes have been used for various purposes, such as self localization of mobile robots [48], where the codes were used as landmarks and contained a complete dataset, including geometrical position, normal vector, physical size and shape. QR codes have also been used in developing robot indoor position and orientation methods [49]. In this method, a camera was mounted on the robot facing upward. The camera detects the QR codes attached to the ceiling, and calculates the position and orientation of the QR code. From that information, the robot calculates its absolute position and direction of heading. QR codes have also been used in developing guide robots [50] that use a similar approach. The guide robot follows a path while detecting QR codes used as landmarks along the path. Another use of QR codes is the design and implementation of augmented reality systems [51]. These techniques

heavily depend on quick and accurate detection and recognition of QR codes.

Extensive research has been carried out for fast and accurate detection of QR codes [52], image analysis for QR code recognition [53], detection of low resolution QR codes using super resolution images generated from multiple low resolution images [54], QR code recognition in snapshot taken by mobile phones [55, 56], and extraction of QR codes from a non-uniform background [57]. Detection accuracy as high as 98% has been achieved from low resolution images [54]. For fast detection of QR codes from arbitrarily acquired image, the Viola-Jones algorithm has been used [58]. The fast detection is made possible by focusing on the most promising regions of the image, and classifying the patterns using rapid feature calculation. Thanks to these works, innovative uses of QR codes are now possible; mapping crane workspaces is one of them.

In this thesis, image processing techniques such as blurring, stitching, template matching, masking, thresholding, and watershed transformation are used to produce a computer vision based map. For the QR code-based mapping algorithm, QR code encoding and decoding techniques are used to encode and decode the QR codes, and edge detection technique is used to calculate the orientation of the QR codes. The empirical goodness and empirical discrepancy methods are applied to evaluate the performance of each of the mapping algorithms, and the combination of the two algorithms.

1.3 Thesis Contributions

This thesis aims to improve the crane operations by providing the crane operator with a near-realtime map of the entire workspace. A novel approach for mapping the workspace using computer vision is introduced. Image processing algorithms including image stitching, image thresholding, and watershed transformation are intelligently combined to generate the final map. Older maps included in the final map help in knowing the older positions the obstacles, which in turn helps in predicting the future positions of

the obstacles. A novel approach of mapping the workspace with QR codes is also introduced. The QR code based map can detect obstacles that are labeled with a QR code. On the other hand, the color-based segmentation used in the pure machine-vision-based mapping technique can detect any obstacle, provided that there is a good contrast between the foreground and the background. Therefore, it is preferred that the two techniques be applied together when possible.

This thesis will enable some significant future research, including automatic obstacle avoidance using computer vision. The map could be used to formulate an optimum path-planning algorithm. It could also be extended to the third dimension to produce a 3D map providing depth information, which would help decide the operator whether the crane can be hoisted over the obstacles.

1.4 Structure of the Thesis

In the next chapter, the tools used in this research are briefly introduced.

In Chapter 3, Mapping Using Machine Vision, the machine-vision-based mapping process is presented. The image acquisition method for mapping is discussed. Then, the image processing steps necessary to produce the map are introduced, followed by the image stitching algorithm. Next, the mapping process is explained in detail including the decay function and memory factor, and, in conclusion, the effect of different parameters on the mapping performance is discussed with examples.

In Chapter 4, Mapping Using QR Codes, an algorithm using QR codes for mapping the crane workspace is presented. The method of creating an obstacle database, the encoding and decoding method of the QR codes, and finding the center and angular orientation of the QR codes are discussed. Then the method of generating the map from the data read from the QR codes is explained. Next, the combination of these two maps is presented and compared with the QR code generated map and the machine-

vision-based map. At the end of this thesis, the conclusions made from this thesis are discussed.

Chapter 2

TOOLS USED FOR MAPPING

This chapter is an overview of the tools used in this research. The hardware used includes a small-scale crane, several types of cameras, and a variety of obstacles for testing. The OpenCV library was used for image processing, the python QR code package, qrcode 5.1, was used for generating QR codes, and the open-source ZBAR library was used for decoding QR codes. All code was written in the C++ programming language, except for the QR code generation, which was written in Python.

2.1 Experimental Setup

Figure 2.1 shows the workspace of the small-scale crane used for the examples presented in this thesis. The workspace is 108×89.75×69.25 inches. It is controlled using a Siemens PLC and driven by Siemens AC servomotors. For the mapping technique presented in this thesis, a camera was mounted on the crane trolley.

2.2 Cameras Used

The color images used in this thesis were taken with a Logitech C920 webcam, a Nokia Lumia 521, and an iPhone. The grayscale images in this thesis were taken with a Cognex In-Sight 7000 camera. The lens used with this camera was a manually-focused Fujinon lens with 9 mm focal length, f/1.4 to f/16 aperture, and a maximum 29°52′ horizontal and 22°37′ vertical angle-of-view. Figure 2.2 shows a Cognex In-Sight 7000 camera.

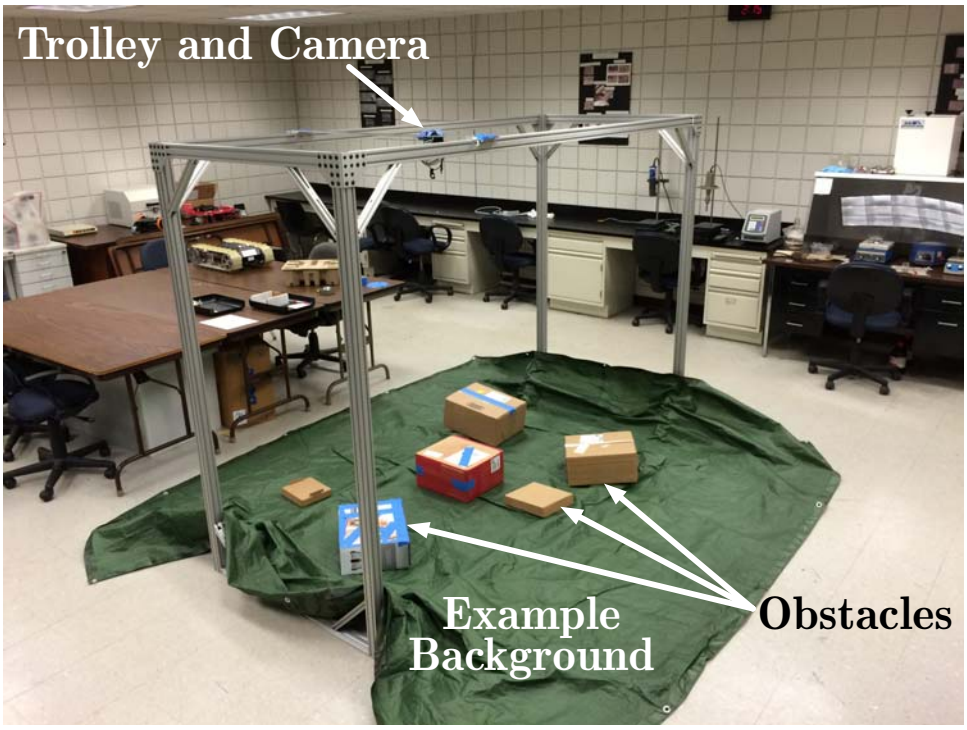


Figure 2.1: Small-Scale Crane Used in this Thesis

2.3 OpenCV

OpenCV is a widely used open-source computer vision library, written in C and C++. It supports Windows, Mac OS, iOS, Android, and Linux [59]. This library can be used in C, C++, Java, and Python. In this thesis, OpenCV 2.4 was used in most of the image processing, including image stitching, template matching, image thresholding, histogram calculation, watershed transformation, and morphological operations. All the code in this thesis was written in C++.

2.4 ZBAR

ZBAR is an open-source barcode reading library [60]. It can read barcodes from images, video streams, and sensors. ZBAR also supports reading and decoding QR codes. In this thesis, the C++ interface of the ZBAR library was used to decode all of the QR codes in the workspace.

Figure 2.2: Cognex Insight 7000 [2]

2.5 QRCODE 5.1

qrcode 5.1 is a python package for QR code generation. To generate QR codes it uses Python Imaging Library (PIL). While generating QR codes, the user can specify the size, version, level of error correction, and box size, depending on the application and amount of data to be encoded. All of the QR codes in this thesis were generated with this module.

Chapter 3

MAPPING USING MACHINE VISION

This chapter will explain the mapping algorithm using machine vision in detail. This algorithm combines a number of machine vision tools including stitching, template matching, and image segmentation to produce the final map. The method of including older maps with the latest map to give an idea of previous positions of the obstacles will be introduced. The mapping performance will be evaluated, and the segmentation methods will be compared. The results show that the mapping method is promising and can be further developed for industrial use.

3.1 Overview of Mapping

Figure 3.1 shows a flowchart of the mapping process [61]. A camera mounted on the crane trolley takes a picture each time it moves to a designated position. The camera can not capture the entire workspace in a single image, so the individual images need to be stitched together. The camera is mounted directly over the hook, so there is always the crane hook/payload in the image. The hook is located and masked before the images are stitched together.

Simple thresholding or the watershed transformation algorithm for foreground-background segmentation is applied for obstacle detection. For the watershed algorithm, a marker image is created where portions of the foreground and background are identified through a series of thresholding operations. The contours of the obstacles are extracted from the segmented image, and polygonal curves are estimated from the contours using the

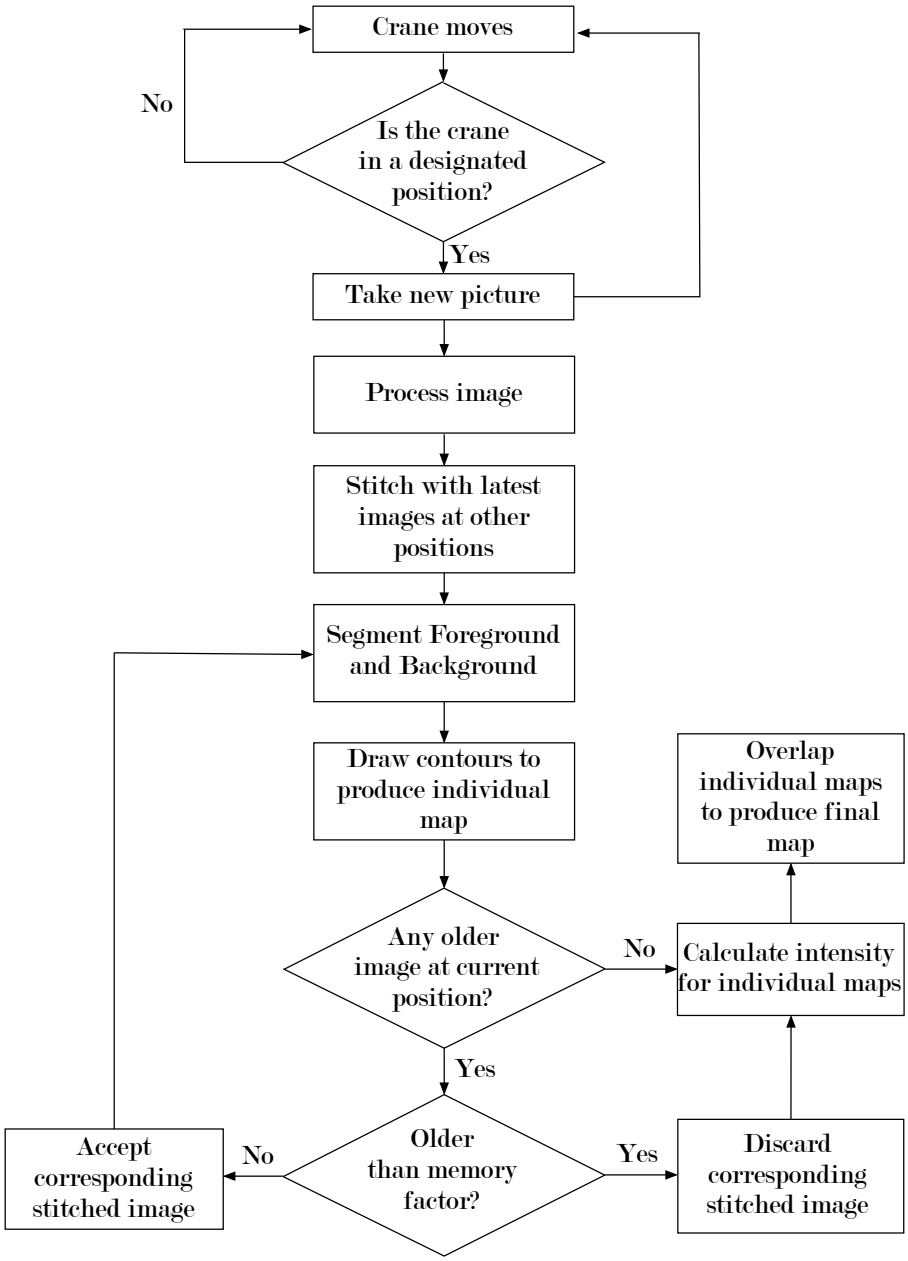


Figure 3.1: Flowchart of the Mapping Algorithm

Douglas-Peucker algorithm [62]. Then the map is drawn. This map is overlapped with the older maps, and a final map is generated. In the final map, the most recent positions of the obstacles are shown in red. The older positions are shown in yellow, the intensity of which decreases exponentially with time and depends on number of maps

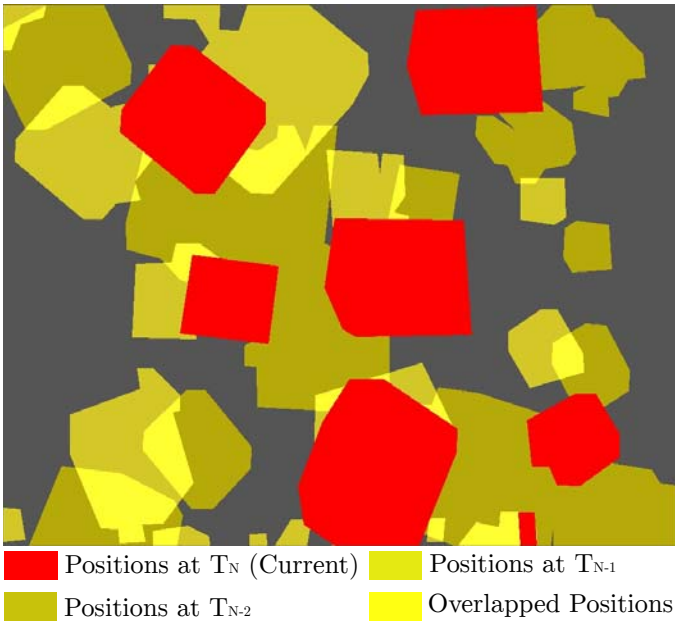


Figure 3.2: Crane Workspace Map

available.

An example map resulting from this process is shown in Figure 3.2 for the workspaces shown in Figure 3.3. The time indicated in the figures is the time when the latest image of each workspace was taken. The intensity of the yellow regions in the final map indicates how recently there were objects at those locations. For example, a high intensity of yellow indicates that an obstacle was at that location very recently. In Figure 3.2, T_{N-1} is more recent than T_{N-2}, so the obstacles at T_{N-1} are represented by a brighter shade of yellow than the obstacles at T_{N-2}. The locations of past obstacles are included with the intention of informing the operator of the possibility of obstacles being there in the future. For example, a product loading area in a factory may have been empty when the most recent map was generated, despite there often being obstacles there. The following sections will discuss each part of this algorithm in more detail.

(a) Workspace at Time T_{N-2} (b) Workspace at Time T_{N-1} (c) Workspace at Time T_N

Figure 3.3: Workspaces Used to Generate Map

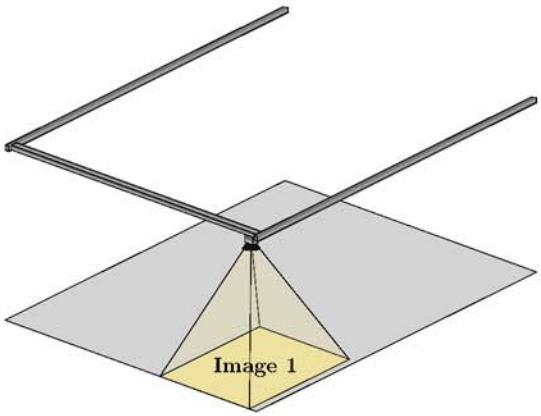
3.2 Image Acquisition

In order for the mapping process to work automatically, the image acquisition method also has to be automatic. The workspace is divided into segments, and a picture is taken in each segment to cover the entire workspace. There are designated positions with a buffered region around each. When the the crane trolley reaches one of these buffered regions, the camera automatically takes a picture. Then, it waits until it moves to another position, or for a certain period of time, depending on the frequency the map is updated if it stays at the same position, before taking the next picture. This automatic image acquisition works parallel to the normal crane operation. There is no stoppage time. However, for the first-time operation, it is recommended that the crane be moved through the entire workspace. Figure 3.4 shows the image acquisition process.

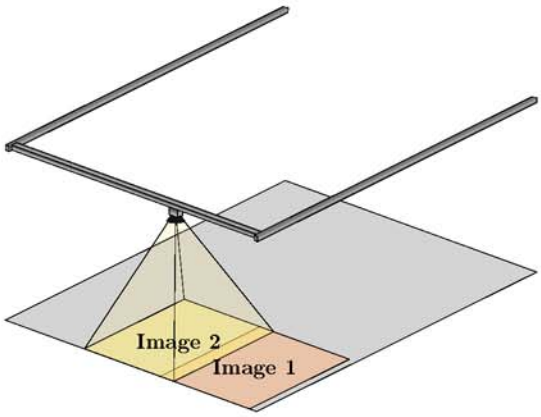
3.3 Individual Image Processing

In order to eliminate noise in the images, the workspace images are first blurred. Blurring smooths the images by reducing image noise and details. Gaussian blur and median blur are two of the most commonly used methods for blurring and are used in this work. Figure 3.5 shows an example image before and after it has been blurred.

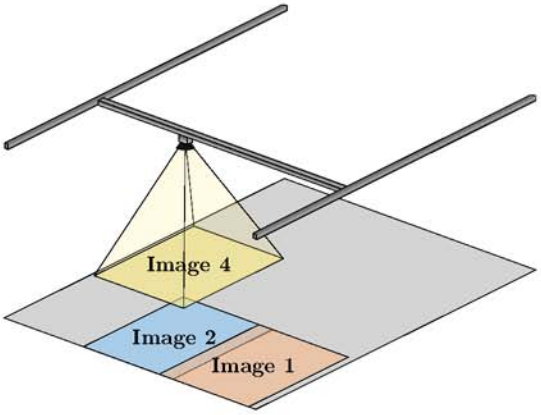
To separate the obstacles from the background, it is necessary to detect the background. If the background color is known and uniform, a simple thresholding can be used to



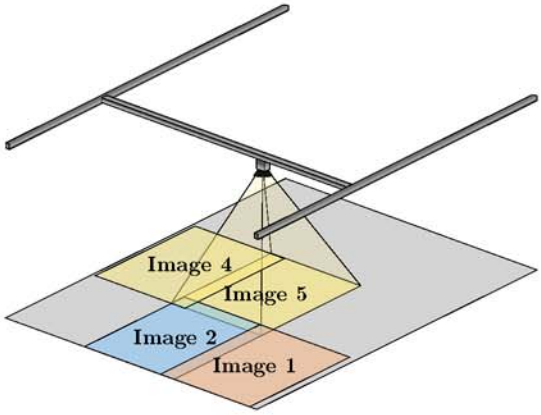
(a) Image at Position 1



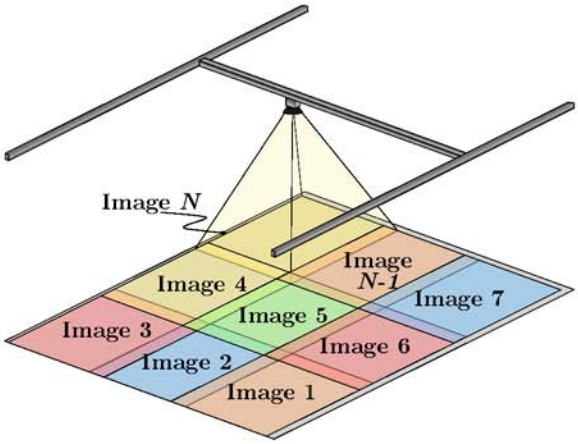
(b) Image at Position 2



(c) Image at Position 4



(d) Image at Position 5



(e) Image at Position N

Figure 3.4: Image Acquisition for Mapping

(a) Original Image (b) Blurred Image

Figure 3.5: Image Blurring

separate the background from the foreground. However, depending on the noise level and lighting, the background of the workspace may vary significantly. Moreover, the reflections of the lighting source and shadows of the objects can make it difficult to distinguish the obstacles from the background using simple thresholding.

A simple way to select the average background is calculating the image histogram for all three channels of an RGB image, as shown in Figure 3.6. It is assumed in this work that the background occupies a larger area than the obstacles. If this is the case, the most frequent pixel value can be selected as the background value for each channel. Figure 3.7 shows the background selection of the blue channel. The background of the other two channels are selected similarly.

Because the camera is mounted directly over the hook, the crane hook/payload will always be in the image. A simple solution is to cover that part of image with the calculated background color, as shown in Figure 3.8. In this figure, the crane hook mask is shown as semi-transparent to aid in understanding the algorithm. In practice, it is opaque.

One way to locate the hook in the image is template matching [63], which is a tech-

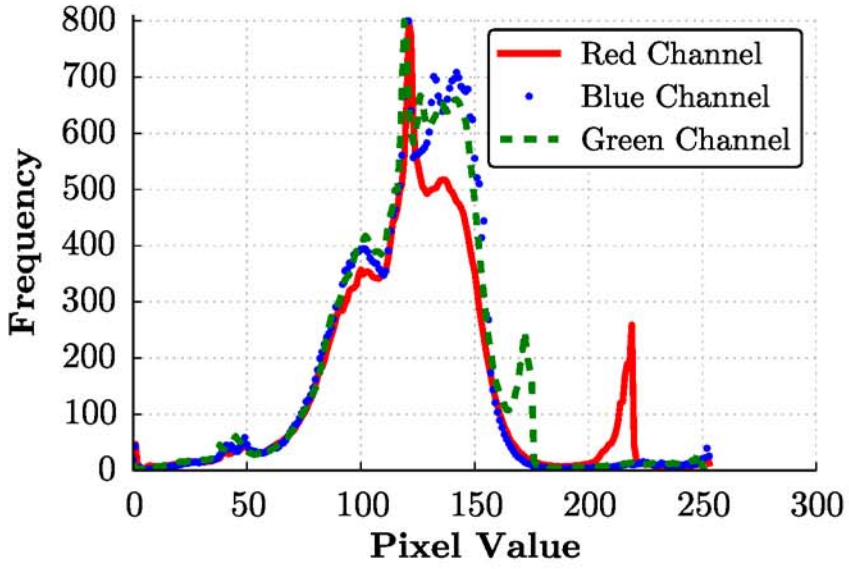


Figure 3.6: Histogram of an RGB Image

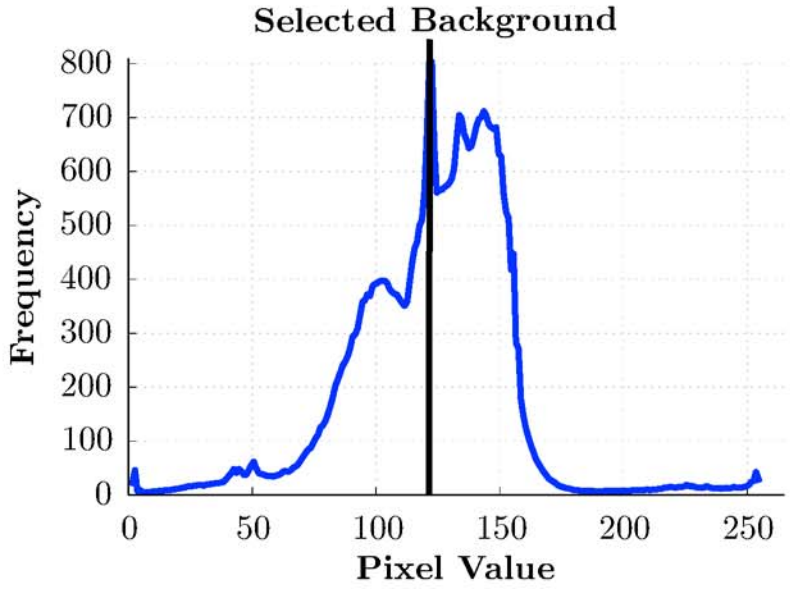


Figure 3.7: Background Selection of the Blue Channel from Histogram

nique for finding a part of a source image that matches with a smaller image known as a template image. The template image slides through the source image one pixel at a time in order to find a match. In this case, the source image is the picture taken by the camera, and the template image is a picture of the crane hook. A metric that repre-

Figure 3.8: Individual Image after Crane Hook is Masked

sents the goodness of the match is calculated at each position, and stored in a resultant matrix which is of the same size as the source image, as shown in Figure 3.9.

Each point of the resultant matrix contains the match-metric at that point. There are different matching methods available. The location in the resultant matrix where the highest or lowest value of the metric is found, depending on the matching method used, is the base point of the matching portion of the source image with the template image. For the matching method used in this particular example, the point in the resultant matrix with the lowest value is the base point of the location where the best match is found. The source image with the matched template is shown in Figure 3.10. After locating the hook and calculating its dimensions, it can be masked.

3.4 Image Stitching

Because the camera can not capture the entire workspace in a single image, the individual images need to be stitched together, as shown in Figures 3.11 and 3.12. The OpenCV stitching pipeline is used to stitch images, which uses automatic panoramic image stitching using invariant features [64]. This algorithm can seamlessly stitch mul-

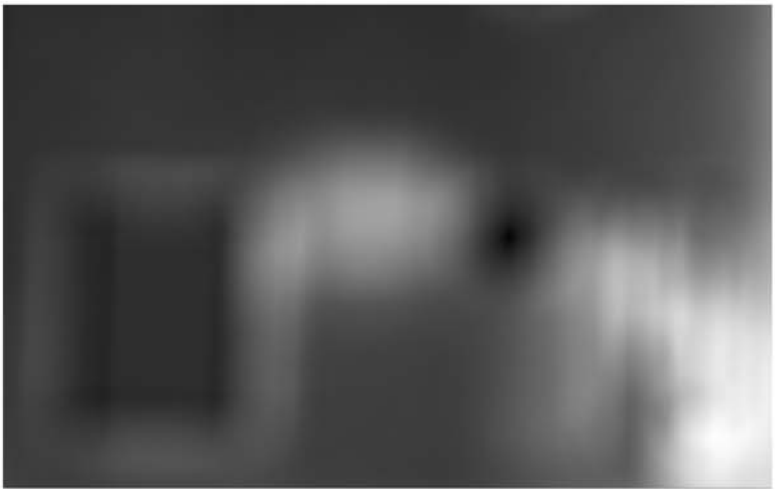


Figure 3.9: Resultant Matrix from Comparison of Source and Template Images

Figure 3.10: Source Image with the Template Located

tiple images irrespective of order or brightness, which makes it suitable for the mapping algorithm. It can also handle small rotations and blends the images seamlessly. In this algorithm, the SIFT (scale-invariant feature transform) features of all the images to be stitched are extracted and matched [65]. RANSAC (random sample consensus) [66] is used to find the feature matches between images that are geometrically consistent, and a probabilistic model is used to verify those matches. Then, connected components of the images are found and bundle adjustment [67] is performed to simul-

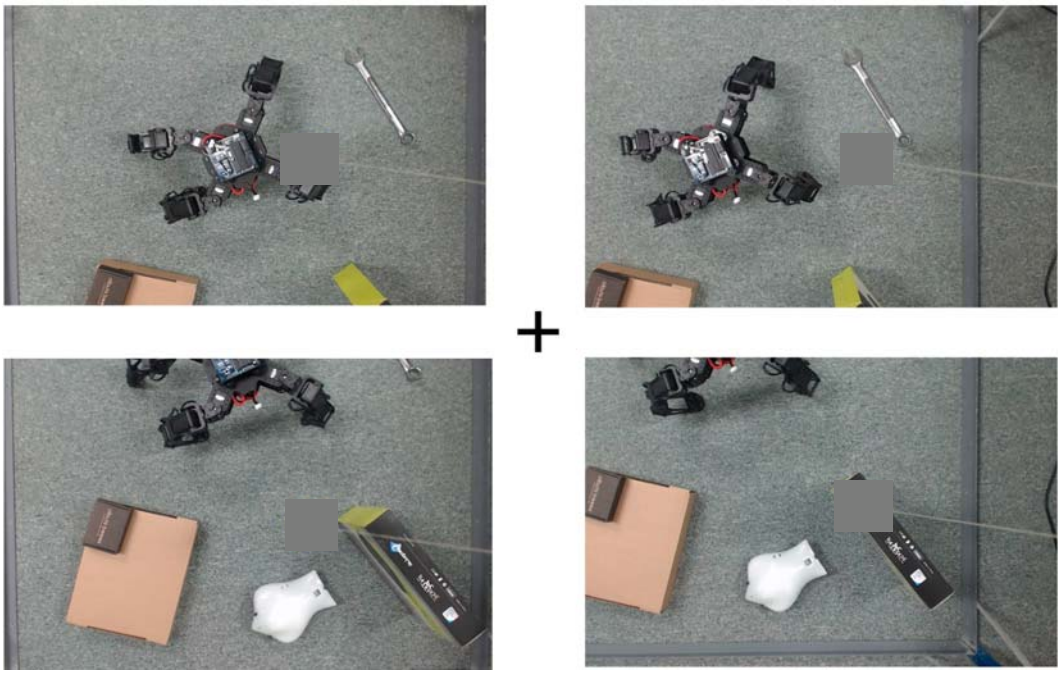


Figure 3.11: Individual Images to be Stitched

taneously solve for all the camera parameters. The images are then straightened and blended together using multi-band blending [64] so that the output stitched image is smooth.

Despite the advantages, there are some problems in using this stitching algorithm. There is a distortion in the stitched image because the distance of the camera from the objects is small. Also, it takes processing power and time to stitch multiple images together.

3.5 Obstacle Detection

Some advanced methods for foreground-background segmentation are available, including graph partitioning methods [68], region-based methods [69], and integration of multiple cues [70]. In this research, the watershed transformation, which is a combination of edge-based and region-based algorithms is applied because it always provides closed contours with a low level of noise and low computation time [71].

Figure 3.12: Stitched Image

For obstacle detection, a marker image has to be created where a portion of the foreground and background are marked. The foreground and the background of the image are segmented by the watershed algorithm based on these marked areas. To create the marker, the image is thresholded twice, through two three-channel scalars about the previously-selected background value. For the first thresholding operation, the threshold scalar range is larger, so that only the portions of the image that fall outside the threshold range are sure to be the foreground and are marked as such. For the second thresholding operation, the threshold range is smaller, so that only the parts of the image that fall inside the threshold range are sure to be the background and are marked as background. This is shown in Figure 3.13 for only blue channel. Using these operations, a marker is created, and the aforementioned regions are labeled inside it, as shown in Figure 3.14. Here, the background is marked in grey, the foreground is marked in white, and the rest of the image is black.

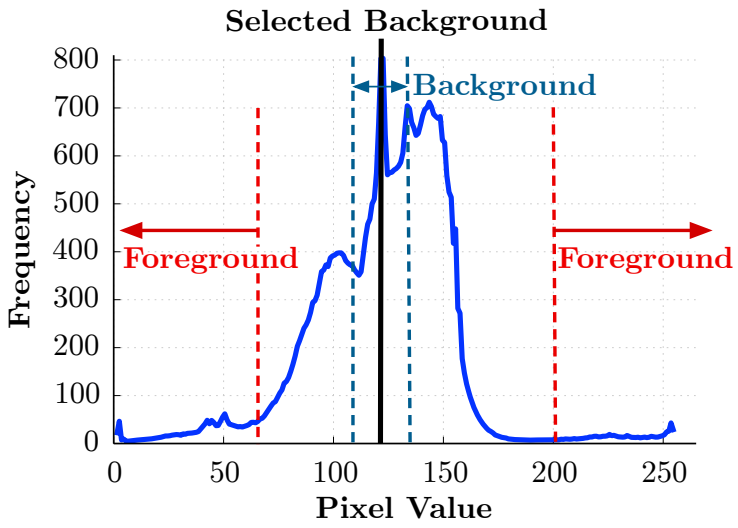


Figure 3.13: Background and Threshold Values Used for Marker Image Generation

Once the marker is determined, the watershed algorithm is applied to obtain the segmented image. The contours of the obstacles are extracted from the segmented image. Then, a polygonal curve is estimated using the Douglas-Peucker algorithm [62], and the contours are drawn, as shown in Figure 3.15.

Another simple method for image segmentation is converting the stitched image into grayscale, then thresholding it through an upper and a lower threshold value. The upper threshold is larger than the average background value, and the lower threshold is smaller than the average background value. Thresholding through the upper and lower thresholding values yield two segmented images, one in which objects brighter than the background are segmented, and one in which objects darker than the background value are segmented. Adding these two segmented images produce a image in which all the obstacles are separated. Then, the contours can be extracted and a polygonal curve can be drawn as before. Figure 3.16 shows the entire process for the same workspace shown before. The problem with this method is finding the right threshold values, and even if the right value is found, it is more sensitive to noise than the watershed transformation. Conservative threshold values can result in missed obstacles,

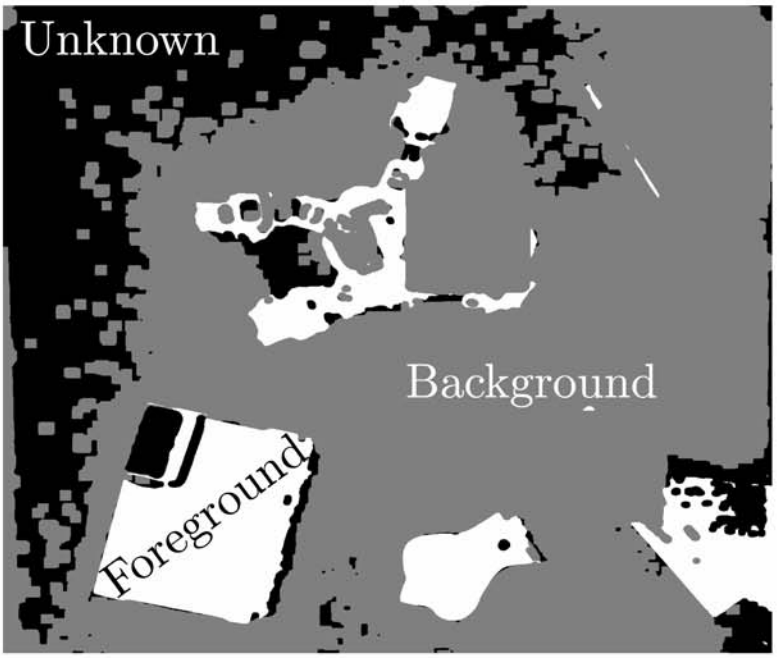


Figure 3.14: Marker Image Used for Seeding

Figure 3.15: Segmented Image After Contours Are Drawn

and large values can result in noise in the map. However, for a particular workspace condition, the threshold values are consistent. Therefore, these values can be calibrated using an interface like the one shown in shown in Figure 3.17, and used for the particular workspace condition.

The resultant map from the threshold values found from this calibration is shown in Figure 3.18. This map contains some noise. However, the process of generating it is simpler than the segmentation algorithms and works in some conditions. It also works with images of different colorspaces, after converting them into grayscale.

3.6 Overlapping Individual Maps

The mapping process presented in this thesis takes both the latest and older positions of the obstacles into account. The older positions of obstacles can be used as an indicator of the likelihood of there being an obstacle at that location in the future. Each time a new picture is taken by the camera at a particular position, an individual map is generated by stitching that picture with the most recent pictures at other positions. Then a final map is created by overlapping the latest map with older individual maps. The individual maps are overlapped in a way that the latest map is shown in red to indicate certainty of finding an obstacle. Older maps are shown in yellow, the intensity of which decreases with time when that maps were generated according to:

$$I_i = 100e^{-cm^2 t_i/(t_0 - t_l)} \tag{3.1}$$

where I_i represents percentage intensity of i^{th} map, m is the number of maps available, c is a scaling factor, t_i represents time since i^{th} map, and t_l and t_0 represent time since the most recent map and the oldest map, respectively. The scaling factor depends on the workspace conditions and the frequency with which the map is updated. For example, if the map is updated very often, a higher value of c is chosen, so that there is

(a) Obstacles Darker than the Background

(b) Obstacles Brighter than the Background

(c) Combination of the Two Thresholds

(d) Resultant Map

Figure 3.16: Grayscale Thresholding Process

Figure 3.17: Screenshot of the Calibration of Upper and Lower Threshold Values

Figure 3.18: Resultant Map After Calibration

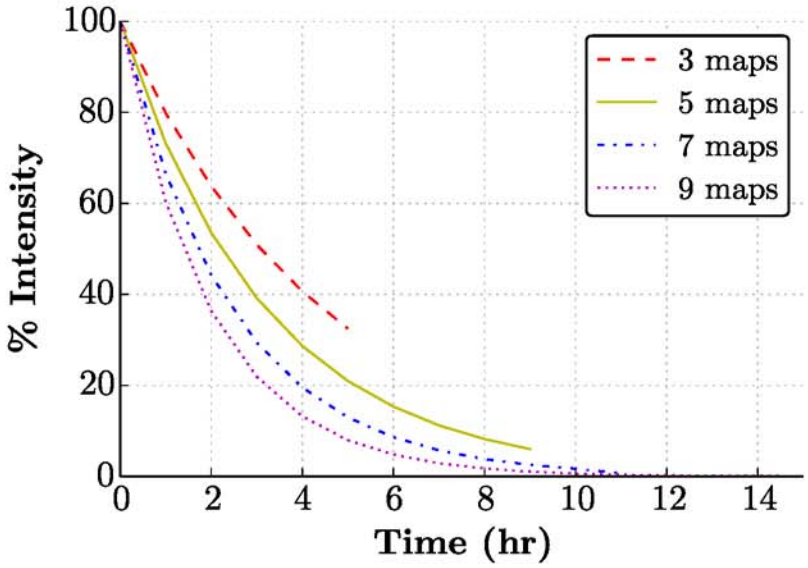


Figure 3.19: Example Definition of Intensity as a Function of Time

enough contrast between maps. On the other hand, if the map is updated less often, a lower value of c is chosen, so that the older maps are clearly visible.

Relatively recent maps are more likely to indicate an actual obstacle position. So, as time progresses, the intensity of older images is decreased. This indicates the decreased certainty of finding an obstacle at that location. The intensity depends both on the number of images available and time when the image was taken, as shown in Figure 3.19. With respect to time, intensity approaches linearity when the number of images available is reduced, making sure that the images do not lose weight too quickly when not many images are available in a particular location.

Using multiple older images at a given location provides additional information. If there is an overlap in the obstacle locations from past images, the overlapped area is brighter than in the individual maps. The overlapped area indicates there has been an obstacle present at these locations in the workspace at multiple times in the past. This suggests that the probability of finding an obstacle there in the future is greater.

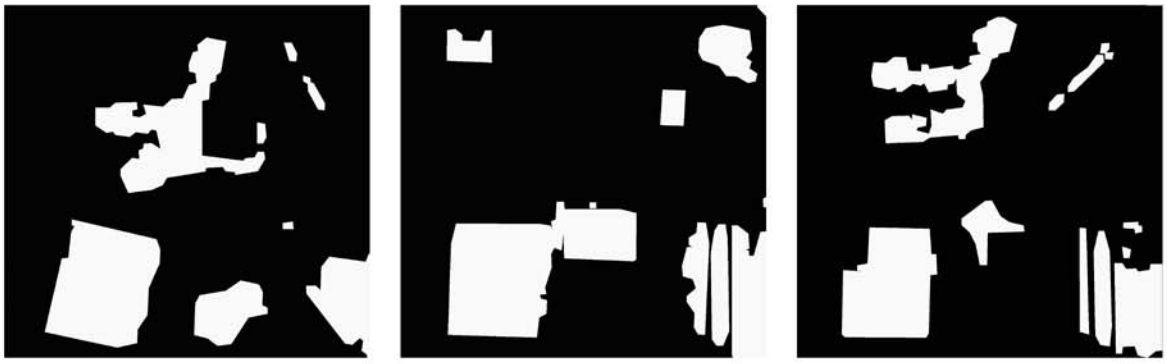
3.7 Memory Factor

If too many old maps are overlapped with the most recent map, the map becomes cluttered and too many obstacles at different points of time make the map confusing. Also, maps that are too old do not give much indication of probable future locations of the obstacles. To solve this problem, if an individual map is too old, it is discarded. A memory factor determines when an individual map is discarded. The larger the memory factor is, the older an individual map can be and still be taken into account for generating the final map. The memory factor is calculated using:

$$M = \begin{cases} t_0 e^{-nc_1} & \text{if } n > 2 \\ t_0 & \text{if } n \leq 2 \end{cases} \tag{3.2}$$

where M is the memory factor, t is the time since the oldest stitched image available, n represents number of older stitched images available, and c_1 is a scaling factor. If an image is older than M, then it is forgotten. If the number of images available is less than or equal to two, no image is forgotten. The memory factor is a function of both number of images, n, and time since the oldest available image, t. For a particular t, the memory factor reduces exponentially with number of images available. If a large number of images are available, the memory factor is smaller, so that more images are forgotten. If the number of images available is small, the memory factor becomes bigger, so that the older images are not forgotten. A maximum memory factor could be set up to make sure that the maximum number of images considered is limited.

As an example, individual maps generated at times T_N, T_{N-1}, T_{N-2}, where N is number of images, are shown in Figure 3.20. The resulting map is shown in Figure 3.21. In this map, the red areas show the most recent positions of the obstacles. Medium-bright yellow areas show next to the most recent positions, and the dark yellow areas show the oldest positions of the obstacles. The overlapping areas between the older obstacle



(a) Map at Time T_{N-2} (b) Map at Time T_{N-1} (c) Map at Time T_N

Figure 3.20: Individual Maps Used in the Final Map

Figure 3.21: Final Workspace Map

positions are shown by the brightest shade of yellow.

3.8 Performance Evaluation

Although this mapping technique is promising, there are some problems yet to be addressed. The background detection method described is based on the assumption that the background area is larger than the total obstacle area, which may not always be true. The thresholding values used for creating markers for the watershed algorithm are also variable. However, the threshold values are fixed for a particular workspace in a particular condition, and they can be calibrated. If there is an object that has the same color as the background, it may go undetected by the segmentation algorithms. Depending on the time of the day, the lighting condition changes in the workspace, which makes adjustment necessary. The effect of various parameters on the mapping performance are discussed in the following sections.

3.8.1 Effect of Memory Factor

Figure 3.22 shows the effect of memory factor on the obstacles detected as a percentage of total workspace area. The total obstacle area, past obstacle areas, and the overlap between past obstacle areas are shown. Obstacles occupy more area if the memory factor is bigger. In this example, when the memory factor is greater than T_{N-2}, all three individual maps are taken into account. The area occupied by obstacles is greater. When the memory factor is less than T_{N-2} but greater than T_{N-1}, the oldest of the three individual maps is forgotten. In this case, both the total obstacle area and past obstacle area are smaller. If the memory factor is less than T_{N-1}, only the most recent individual map is taken into account, the other two are forgotten. The total obstacle area is the smallest in this case, and there is no past obstacle area shown in the map in yellow. The effect of memory factor on the appearance of the map is shown in Figure 3.23.

As can be seen in (3.2), the memory factor depends on number of images available and the scaling factor. The scaling factor is chosen based on how frequently the map up-

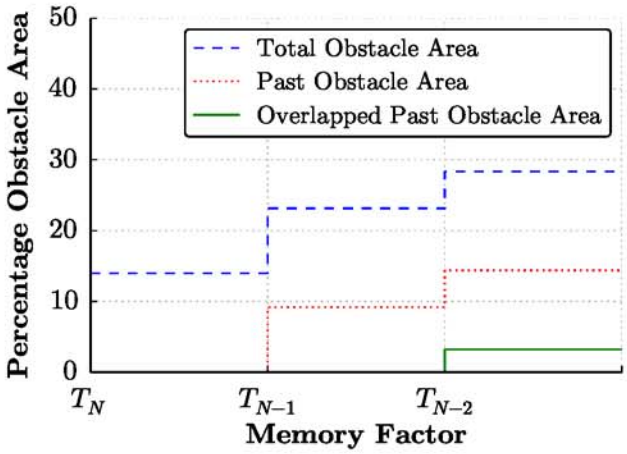


Figure 3.22: Obstacle Detected as a Percentage of Workspace Area vs Memory Factor

(a) M<T_{N-1} (b) T_{N-1}<M<T_{N-2} (c) M>T_{N-2}

Figure 3.23: Effect of Memory Factor on Final Map

dates. If the crane operates slowly, the map also updates slowly. Therefore, a smaller scaling factor is desired, so that older maps are not discarded quickly. Figure 3.24 shows the effect of these parameters on the memory factor for a time duration of 13 hours. For a particular scaling factor, the memory factor is greater if there is a fewer number of past images available. For example, in Figure 3.24, for a scaling factor of 0.2, images older than 3.2 hours are forgotten when number of images available is 7, while images older than 7.1 hours are forgotten when only 3 images are available. This ensures that a fewer number of images are forgotten when there are fewer images available.

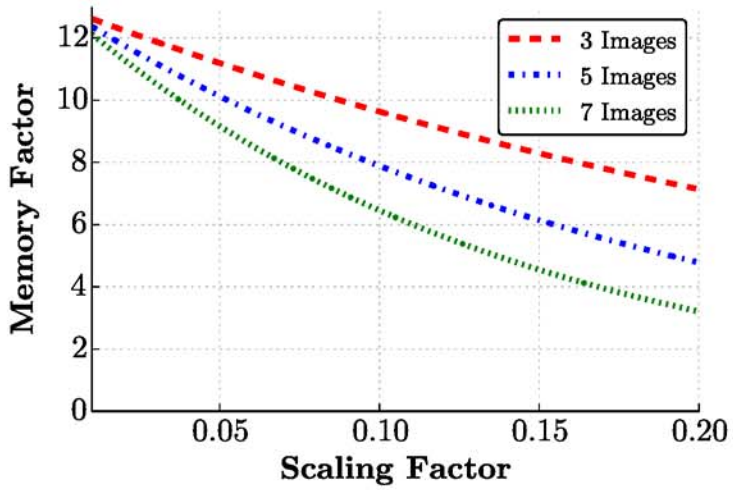


Figure 3.24: Memory Factor as a Function of Scaling Factor

3.8.2 Effect of Parameters on Intensity

The effect of number of available older maps on the intensity of older maps as a function of time is shown in Figure 3.25 for a normalized time range of 0.04 to 1. The normalized time $t_i/(t_0 - t_l)$ is the time since i^{th} map was generated divided by the total range of time over the available past maps. The latest map, which is shown in red is omitted to show the intensity of yellow in older maps clearly. From Figure 3.25, it can be concluded that for a particular number of available images, the scaling factor dictates the contrast between the older maps. The larger the scaling factor is, the greater the contrast.

The absolute intensity of obstacles as a function of normalized time for three different scaling factors is shown in Figure 3.26, which shows that the intensity curve declines quicker for larger value of scaling factor, yielding greater contrast. The intensity of past maps depends on the number of older maps available. Intensity decreases with an increase in the number of overlapped past maps available. This ensures that the map does not get cluttered when the number of past maps is large. In Figure 3.26, the markers on the curves represent the images used to form the maps in Figure 3.25.

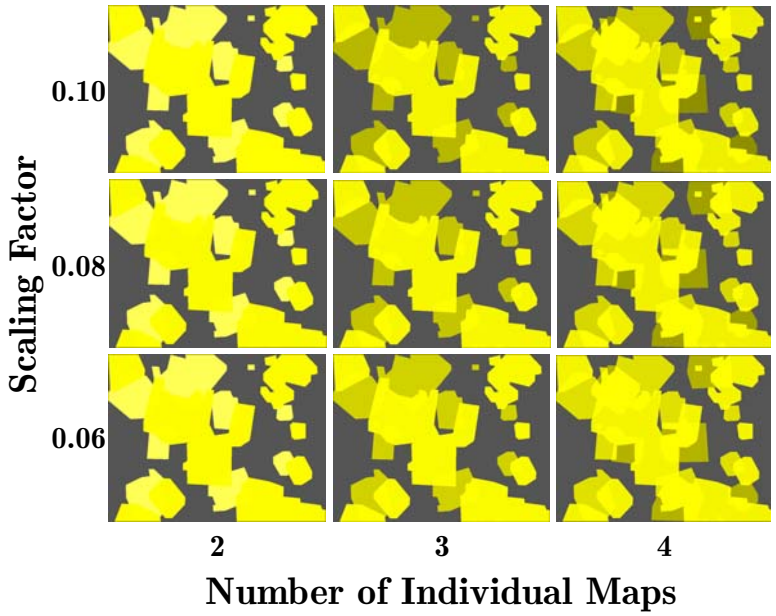


Figure 3.25: Effect of Number of Past Maps and Scaling Factor on Final Map

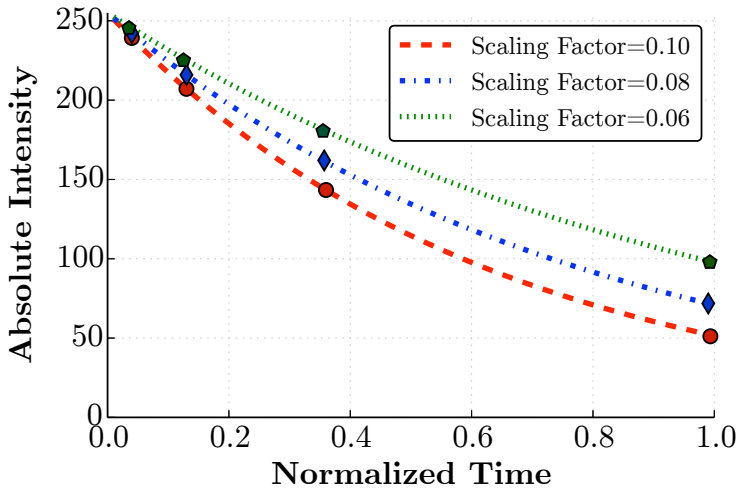


Figure 3.26: Absolute Intensity as a Function of Normalized Time

3.8.3 Comparison of Segmentation Methods

To evaluate the effectiveness of the proposed method, they are compared against a manually-generated map, drawn directly on top of the stitched images, rather than the measurements of the actual obtacles. As such it provides a method to compare segmentation algorithms against one another, but is only an approximation of their absolute accuracy. The results differ based on the contrast between background and foreground and different camera parameters such as exposure, brightness and contrast. However, for a particular camera and a particular set of workspaces, trends are found when the two segmentation methods are compared.

The workspaces used for the comparison of the two segmentation techniques are shown in Figure 3.27. These workspaces vary in background color, workspace size, and the color, shape and size of the obstacles. Segmented images resulting from grayscale thresholding and watershed transformation are compared against the manually-generated images for all of these eight workspaces. Workspace 1 is shown in Figure 3.28, along with the manually-generated image used for comparison and the segmentation results.

Figure 3.29 compares the percentage match of pixels with the manually-generated map for the grayscale thresholding method and the watershed transformation for different workspaces. Every pixel of the map is compared with the corresponding pixel of the manually-generated map. The figure shows that in most cases the watershed transformation algorithm yields a more accurate map than the grayscale thresholding algorithm. Figure 3.30 compares percentage obstacle pixels missed, that is the percentage of obstacle pixels in the manual map detected as background in the actual map.It shows that the grayscale thresholding results in higher percentage of missed obstacles than the watershed transformation. Figure 3.31 compares falsely detected obstacle pixels, that is the percentage of foreground pixels in the generated map that are background pixels in the manually drawn map. It shows that the watershed transfor-

(a) Workspace 1

(b) Workspace 2

(c) Workspace 3

(d) Workspace 4

(e) Workspace 5

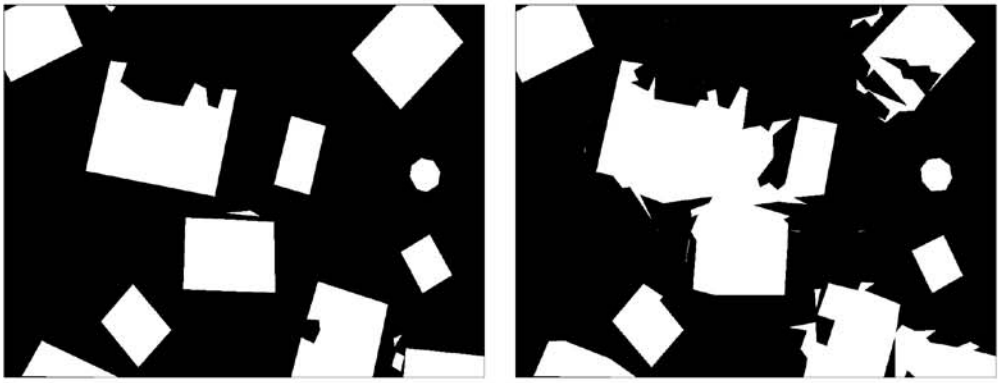
(f) Workspace 6

(g) Workspace 7

(h) Workspace 8

Figure 3.27: Workspaces Used for Evaluating Segmentation Performance

(a) Workspace Image (b) manually-generated Image



(c) Map Using Grayscale Thresholding (d) Map Using Watershed Tansformation

Figure 3.28: Example Segmentation Comparison

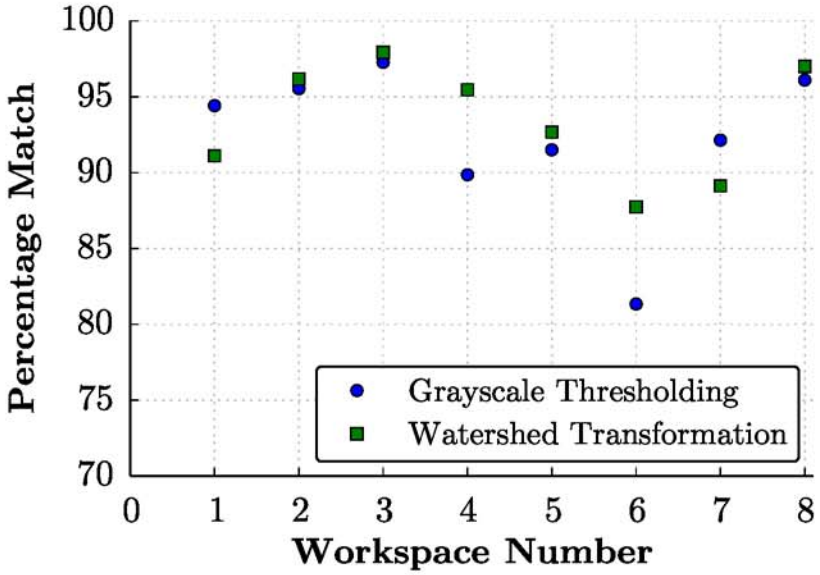


Figure 3.29: Percentage Match of Obstacle Pixels in Different Workspaces

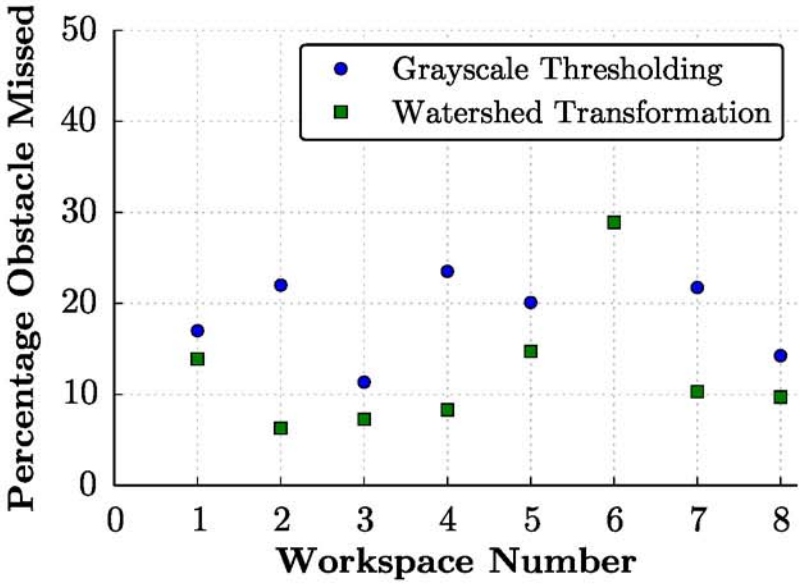


Figure 3.30: Percentage Obstacle Pixels Missed in Different Workspaces

mation algorithm has a tendency of detecting larger area than the actual obstacle size. Overall, the watershed transformation yields better results. However, it is harder to determine the threshold values for three channels in compared to only one channel. Moreover, the computation power and time required for the grayscale thresholding is less than the watershed transformation.

3.9 Chapter Conclusion

In this chapter, a novel approach of machine-vision-based mapping of crane workspace was introduced. The image acquisition method and image processing methods used were discussed. A simple way to mask the crane hook after it had been located using template matching was explained, and the image stitching method to stitch the images together and image segmentation methods to identify the obstacles were discussed. The idea of showing the older maps with different intensities depending on time and the criteria of discarding an older map was explained. Finally, mapping performance depending on different parameters was discussed, and the results of the two segmentation methods were compared. The results show that the mapping techniques can suc-

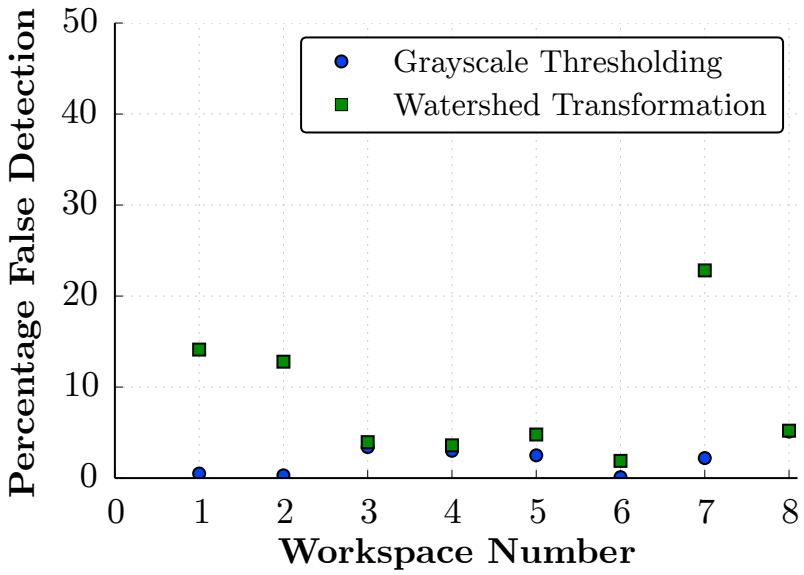


Figure 3.31: Percentage of Falsely Detected Obstacles in Different Workspaces

cessfully detect as much as 96% of the obstacle area, and the watershed transformation method is more robust when it comes to successfully detecting obstacles.

Chapter 4

MAPPING USING QR CODES

This chapter will introduce another method for mapping the crane workspace, which is based on QR codes. The algorithm requires the knowledge of the dimensions of the obstacles likely to appear in the map. These obstacles are labeled with QR codes, and a database is created to store the information of the known obstacles. The camera mounted on the crane trolley takes pictures of the workspace. The QR codes in the pictures taken by the camera are read, and a map is generated by matching the identified QR codes to the database. Unless all the obstacles in the workspace are labeled with a QR code, this mapping technique shows only a fraction of the total number of obstacles. The QR code-based map is combined with the previously described machine-vision-based map to ensure that maximum number of obstacles are shown in the map.

4.1 QR Code Overview

The Quick Response code (QR code) is a kind of two dimensional barcode. Barcodes are used all over the world for encoding information of the items onto which they are affixed. The QR code was first developed for the automotive industry in Japan [72]. Now, it is very popular worldwide because of its greater data storage capacity and quick readability. The datatypes QR codes support include binary, numeric, alphanumeric and kanji. Figure 4.1 shows an example QR code, encoded with the C.R.A.W.LAB web address (http://www.ucs.louisiana.edu/~jev9637/).

Figure 4.1: QR code for the URL of CRAWLAB

4.1.1 Encoding and Decoding

There are several standards for encoding data in QR codes. A QR code consists of black dots, called modules, on a white background. These modules create a unique pattern for the data encoded in the QR codes. For decoding, an imaging device, typically a camera, and a processor are used. The imaging device reads the QR codes, and the processor extracts the data from the pattern present on the QR codes [73]. The processor first locates the position blocks, shown in Figure 4.2, at the corners of the QR coded image. It also locates a fourth square, known as an alignment block, near the other corner to correct distortions. Then, the small modules are converted to binary numbers and validated with an error correcting code.

4.1.2 Storage

The storage capacity of a QR code depends on the encoded datatype, version, and error correction level. The possible datatypes are numeric, alphanumeric, binary and kanji. There are 40 available versions of QR codes. Version 1 is a 21x21 matrix, which can store the lowest amount of data. With each higher version, four additional rows

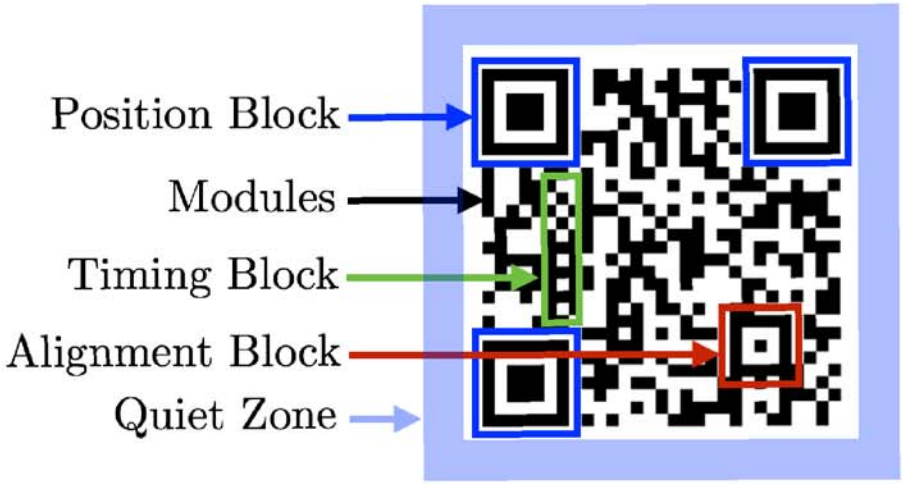


Figure 4.2: Design of a QR Code

Table 4.1: Different Levels of Error Correction in QR codes

Level L	up to 7 % damage
Level M	up to 15% damage
Level Q	up to 25% damage
Level H	up to 30% damage

and columns are added. Version 2 is a 25x25 matrix, version 3 is a 29x29 matrix, and so forth. Version 40 is a 117x117 matrix; it has a maximum storage capacity of 1,264 characters of ASCII text.

4.1.3 Error Correction

QR codes are robust. They can sustain damage and remain readable even if a part of the code is unreadable. This is made possible by Reed-Solomon Error Correction [74]. Backup data are added to make sure that the QR code is readable even if part of it is damaged. The error correction levels are summarized in Table 4.1. It shows that the QR codes with correction level L can survive upto 7% damage, and the QR codes with correction level H can survive up to 30% damage and be still readable.

For higher correction levels, the QR code has to accommodate backup data as well as the original data, which means more rows and columns of modules are required. As

a result, the QR code becomes denser. This is shown in Figure 4.3. It shows that the modules become smaller and denser as the error correction level increases. Two modules at the bottom left corner of a QR code shows the error correction level used in that QR code. Higher correction levels are convenient for industrial use where it is challenging to keep the QR codes clean and undamaged. However, higher correction levels reduce the data storage capacity of the QR codes.

It is also desired to have higher correction levels for the mapping technique proposed in this thesis. Since a portion of the image taken from the crane trolley is blocked by the crane payload, a higher correction rate provides a better chance to retrieve the data even if part of the QR code is obscured by the payload.

4.2 Mapping Algorithm

The advantages offered by QR codes facilitate a method for workspace mapping. For known obstacles, this method could be more reliable than the algorithm presented in the previous chapter, because the exact dimensions of the obstacles are already in the

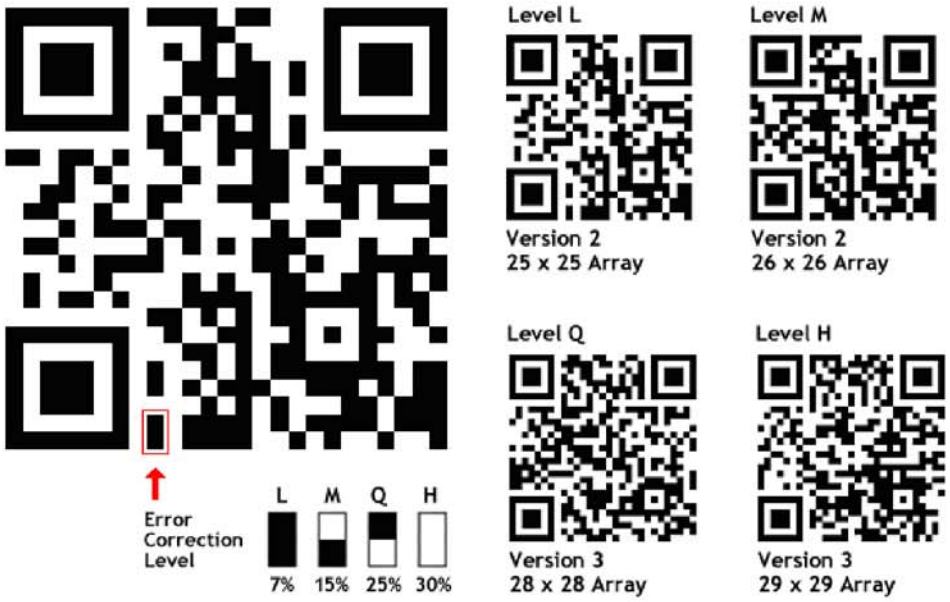


Figure 4.3: QR Code Error Correction Levels [3]

database and the number of undetected obstacles can be counted. The database file should be updated whenever an object is added or removed from the workspace.

Figure 4.4 shows the key steps of mapping using QR codes. In order for this algorithm to work, first a database of all the known obstacles is created. Each obstacle listed in the database is labeled with a QR code, the position of which with respect to the obstacle is listed in the database.

The latest stitched image of the workspace is taken as the input. All the QR codes in the stitched image are decoded, and the data are recorded. The recorded data includes a keyword for the obstacle each QR code is attached to and the position and orientation of that QR code. Next, for each QR code found in the image, the database is searched for the matching obstacle using the keyword. When a match is found, the dimensions of the obstacle and its relative position from the QR code is read from the database and from the obstacle height, the conversion factor from the physical units to pixels is calculated.

Next, the obstacle position and orientation is calculated from the QR code position and orientation, and using the conversion factor, the data are converted into pixel units. Finally, the obstacle is drawn on the map. The process continues until all the obstacles labeled with QR codes read from the stitched image are drawn. Figure 4.5 shows an example workspace, a complete map is shown in Figure 4.6.

4.2.1 Creating Obstacle Database

Since the mapping technique using QR codes works only for known obstacles, a database is necessary for the algorithm, which is created in text format where obstacles with their types and dimensions are listed. The position of each QR code relative to the obstacle onto which it is affixed is also listed. Taking precise measurements of the obstacles and putting them in correct order is imperative for the algorithm to work properly.

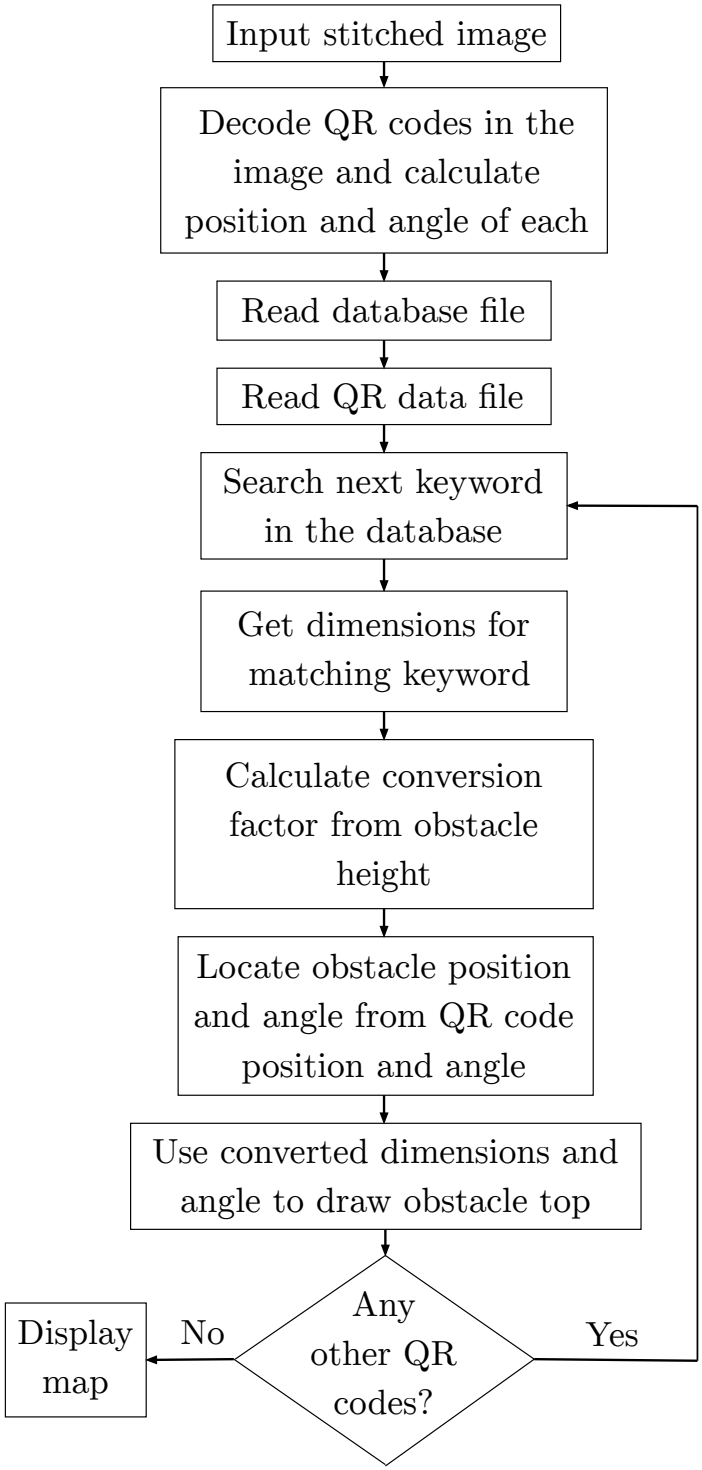


Figure 4.4: Flow Chart of Mapping Algorithm Using QR code

Figure 4.5: Workspace with QR Code Labeled Obstacles

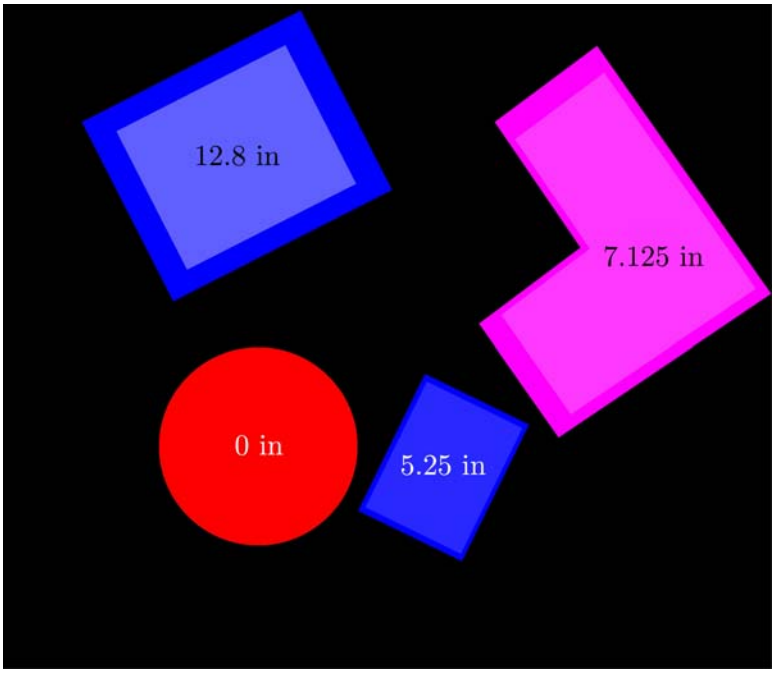


Figure 4.6: Resultant Map of the Workspace

Every obstacle in the database file is assigned with a code, which is also encoded in the QR code of the corresponding obstacle. This code is the keyword for a QR code

Table 4.2: Data Entry System for Circular Obstacles

tank1	r_x	radius	height	circle
c1	6	11	7	

to match with the database file. For example, the keywords for circular obstacles are `c1`, `c2`, ...,`cn`, and for rectangular obstacles, they are `r1`, `r2`, ...,`rn`. The database is updated with every new obstacle added to the workspace. For every obstacle, the data is listed in the following order: obstacle type (circle/rectangle/polygon), obstacle code (example: `c1`, `r3`, `p2` etc.), and dimensions.

For circular obstacles, the dimensions listed in the database are the relative distance of the QR code from the center, the radius, and the height of the circular obstacle. Since rectangular obstacles are common, and the labeling and data entry is easier than that of polygonal obstacles, they are treated separately. Moreover, polygons close to rectangular shape can be simplified as rectangles. For rectangular obstacles, the dimensions listed are the relative distance of the QR code from the lower left vertex, and the length, width, and height of the obstacle. In the case of polygonal obstacles, the number of sides of the polygon is included in the database. The position of each corner relative to the QR code center is listed in polar coordinate, followed by the height of the obstacle. All dimensions are specified in inches.

An example database entry for circular obstacles is shown in Table 4.2. Here, `tank1` is a circular object, and `c1` is the unique keyword for `tank1`. The QR code is placed at a radial distance of 6 units from the center, perpendicular to the radius, and the left side facing the center, as shown in Figure 4.7. The tank has a radius of 11 units, and a height of 7 units.

An example data entry for rectangular obstacles is shown in Table 4.3. Here, `machine1` is a rectangular object, and `r1` is the keyword for `machine1`. The QR code center is

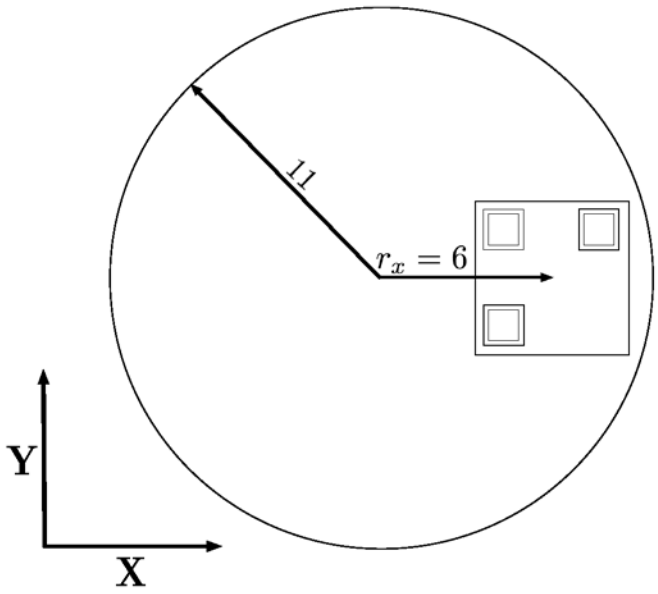


Figure 4.7: Labeling of a Cylindrical Obstacle

Table 4.3: Data Entry System for Rectangular Obstacles

machine1	r_x	r_y	length	width	height	rectangle
r1	10	9	21	17.25	15	

placed at a distance of (r_x, r_y) units from the origin, which in this case is the lower left vertex of the rectangle relative to the QR code. The longer side of the rectangle is chosen as the X-axis. The top, left, right and bottom of the QR code are identified from the position of the position blocks. The corner of the QR code without a position block is the lower right corner of the QR code. The QR code should be parallel to the edges and the lower left corner towards the origin, as shown in Figure 4.8. The length, width and height of `machine1` are 21, 17.25 and 15 units, respectively.

For all other polygons, including triangles, the data entry method is different, as shown in Table 4.4. Here, `machine2` is a polygon with 6 sides. The positions or the vertices relative to the QR code center are listed in polar coordinates, after selecting a convenient horizontal and vertical axis. In the example shown in Figure 4.9, point `pt1` is

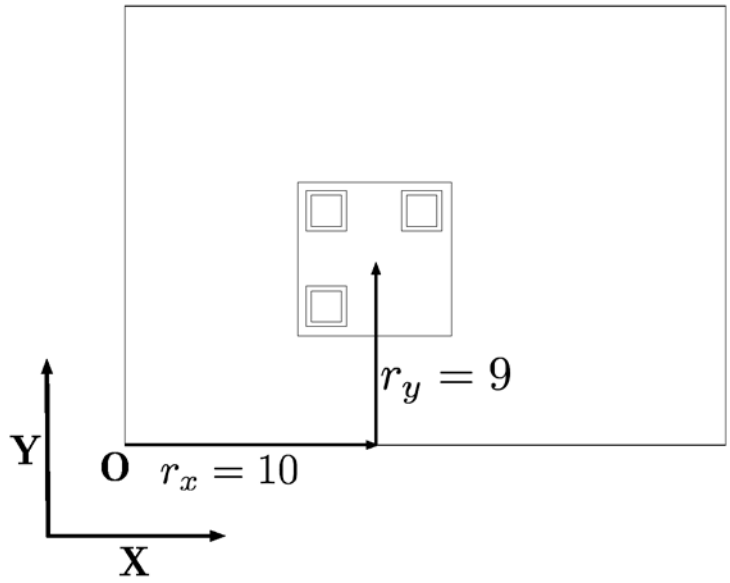


Figure 4.8: Labeling of a Rectangular Obstacle

Table 4.4: Data Entry System for Polygonal Obstacles

machine2	sides	pt1	pt2	pt3	pt4	pt5	pt6	height	polygon
p1	6	40 0°	30 5°	30 135°	40 180°	30 225°	0 315°	18	

40 units and 0 degrees from the QR code center, point `pt2` is 30 units and 45 degrees from the QR code center and so forth.

4.2.2 Encoding QR Codes

Every obstacle in the database is labeled with a QR code encoded with the unique keyword for it. Numerous tools are available for generating QR codes. A Python QR code package named qrcode 5.1 is used in this research [75]. While encoding, the version of the QR code, the level of error correction and the box size can be specified. Since the keyword is short, the level of error correction does not affect the data storage capacity or the size of the QR code, which allows any level of error correction to be chosen. However, since the payload blocks a part of the image, it is recommended to use the highest level of error correction for maximum chance of data retrieval from the QR

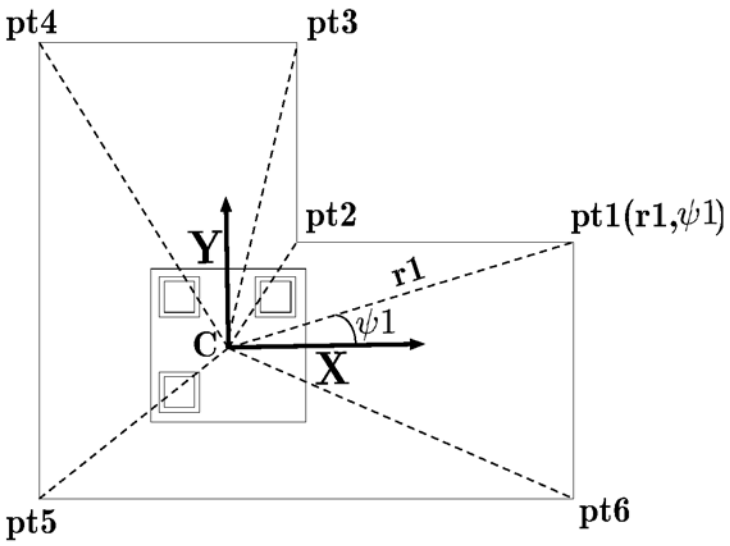


Figure 4.9: Labeling of a Polygonal Obstacle

codes.

QR code version 1 is used for this thesis, because it consists of the least number of rows and columns, allowing the size of the QR code to be large and readable from a long distance. The storage capacity of version 1 is enough for the keyword encoded in it, along with the maximum level of error correction. The box size parameter controls how many pixels each "box" of the QR code is. A higher box size is desired for a good-resolution QR code, allowing it to be printed at large sizes.

4.2.3 Decoding QR Codes

Every time an image of the entire workspace is generated, all the QR codes of the image are read using a QR code decoding library. In this research, the open-source ZBAR library [60] is used for decoding the QR codes. The decoding result, which is the keyword, is written to a text file, followed by the angle with respect to the horizontal and the center of the QR code. The center and angle information are used to calculate the position and orientation of the obstacles. A flow chart of the decoding process is shown in Figure 4.10. Figure 4.11 shows an example image with two obstacles labeled with

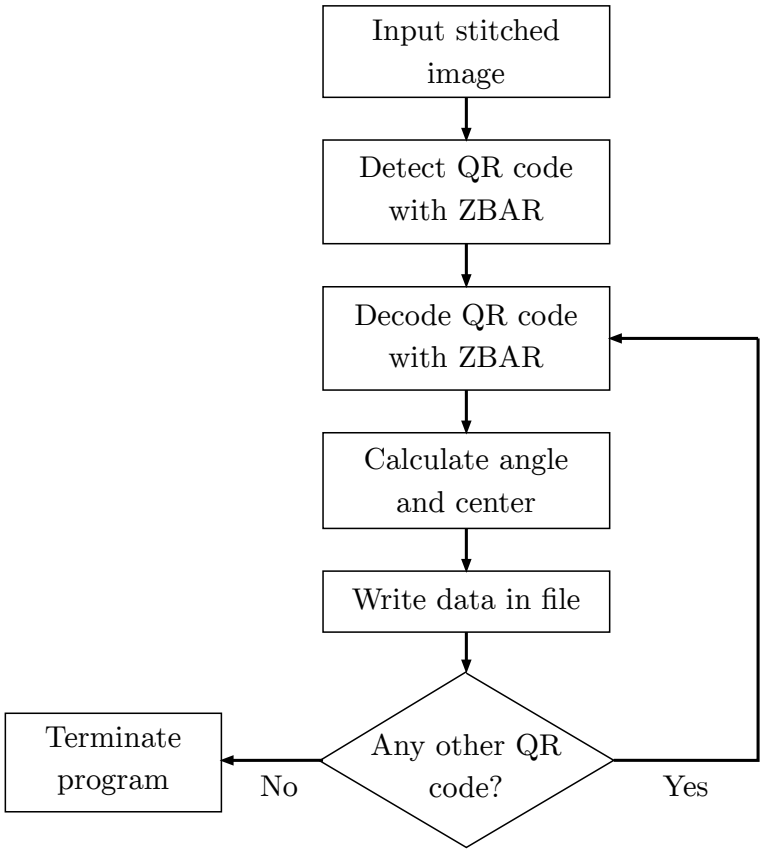


Figure 4.10: Flow Chart of the QR code Decoding Process

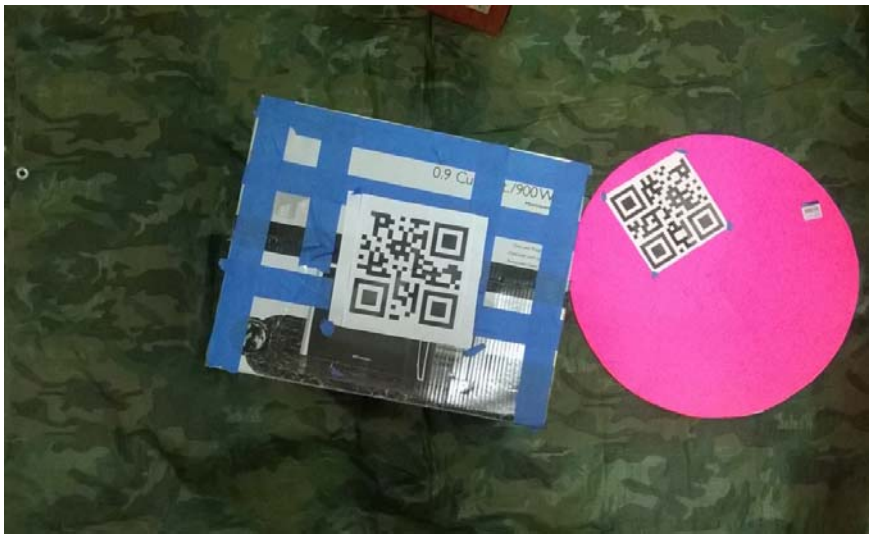


Figure 4.11: Example Image to be Decoded

QR codes. The decoding results are shown in Table 4.5.

Table 4.5: Result of the Decoded QR Codes from the Example Image

c1	237°	820	256
r1	189°	501	326

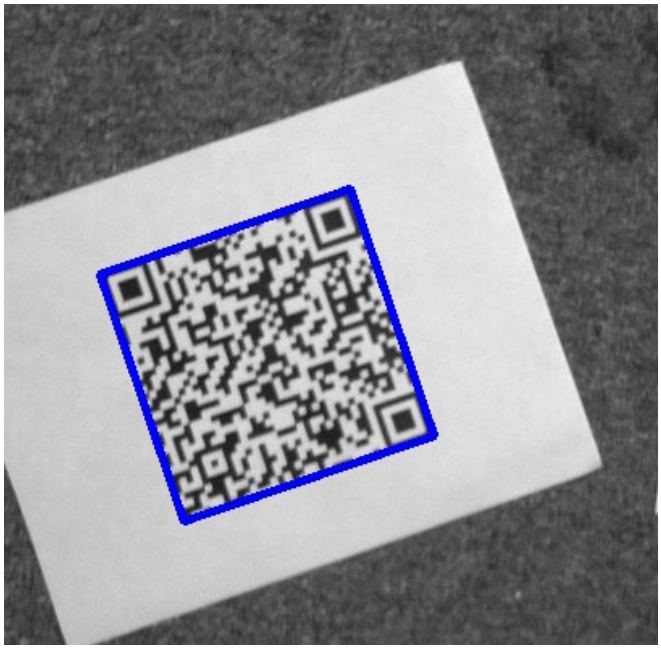


Figure 4.12: Minimum Enclosing Rectangle for finding the QR code center

In Table 4.5, `c1` is the keyword encoded in the QR code on `tank1`. The QR code's position is (820, 256) pixels, and it is at an angle of 237 degrees relative to the positive X-axis. Similarly, `r1` is the keyword encoded in the QR code on `machine1` and the QR code's position is (501, 326) pixels, and it is at an angle of 189 degrees relative to the positive X-axis.

4.2.4 Calculating the Center of QR Codes

For every QR code detected, a minimum rectangle enclosing the QR code is calculated, as shown in Figure 4.12. The four vertices of the rectangle are averaged to find the approximate center of the QR code.

4.2.5 Calculating the Angle of QR Codes

To calculate the QR code angle, the position blocks are first detected. The canny edge detection algorithm [76] is applied to detect all the edges of the image containing the QR code. Then, a contour finding algorithm is used to detect all the contours with hierarchy in the image. The position markers contain five nested contours, as shown in Figure 4.13. This distinguishes the position markers from the other modules of the QR code.

After the three position markers have been identified from the number of nested contours they contain, the relative position of them with respect to each other is determined, and the markers are named as the top, right, and bottom markers. This can be done by using a triangle ABC shown in Figure 4.14, formed by connecting the center of each of the three contours of the three position blocks. The vertex not involved in the largest side is out-lying, and it can be named as the top marker. In this case, C is the top marker. In order determine the right and the bottom marker from the remaining two, the slope of the straight line AB they form, and the distance of AB from C is calculated.

* if *slope* and *distance* are positive, A is BOTTOM and B is RIGHT

* if *slope* in negative and *distance* is positive, A is RIGHT and B is BOTTOM

* if *slope* in positive and *distance* is negative, A is RIGHT and B is BOTTOM

* if *slope* and *distance* are negative, A is BOTTOM and B is RIGHT

Once the top, right, and bottom markers are determined, the angle of the straight line connecting the top and right marker with respect to horizontal is the angle of the QR code with respect to horizontal.

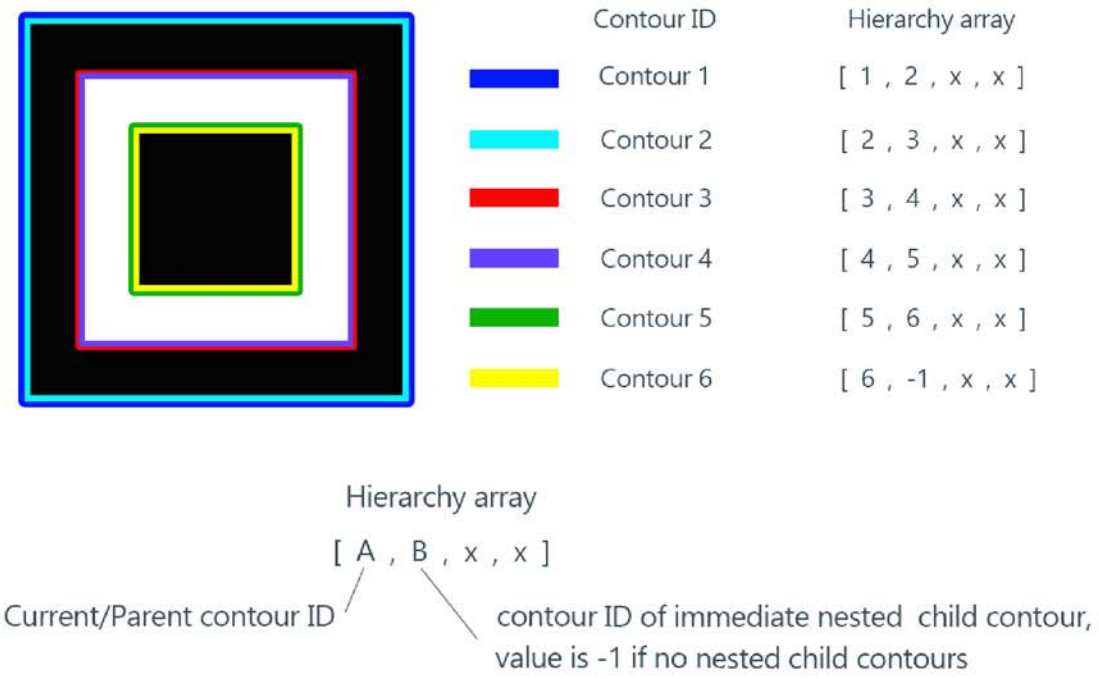


Figure 4.13: Detection of Position Marker from Number of Nested Contours [4]

4.2.6 Generating Maps from QR Code Data

To generate the map from the QR code data, first an empty map of the same size as the stitched image is created. Then, the text file where all the data for the decoded QR codes are saved is read. When a keyword is found, the database is searched for the keyword. Every keyword corresponds to a unique obstacle. When the keyword is found in the database, the shape and dimensions of the corresponding obstacle are read from the database. The center position and angle of the QR code are read from the data associated with the keyword in the QR data file. Then the conversion factor from inches to pixels is calculated from the height of the obstacle and used to draw the top face of the obstacle. If the obstacle height is not uniform, the highest point is assumed to be the obstacle height, and the conversion factor is calculated with that height. Then the obstacle position and orientation is calculated from the QR code position and orientation.

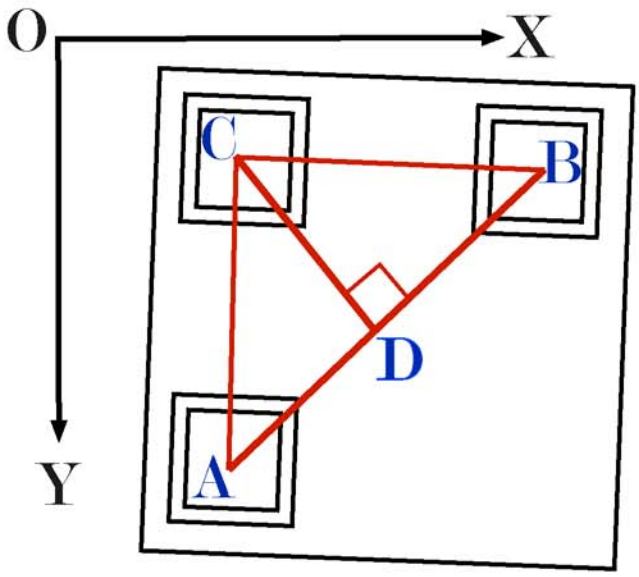


Figure 4.14: Detection of Top, Right and Bottom Markers for Angle Calculation

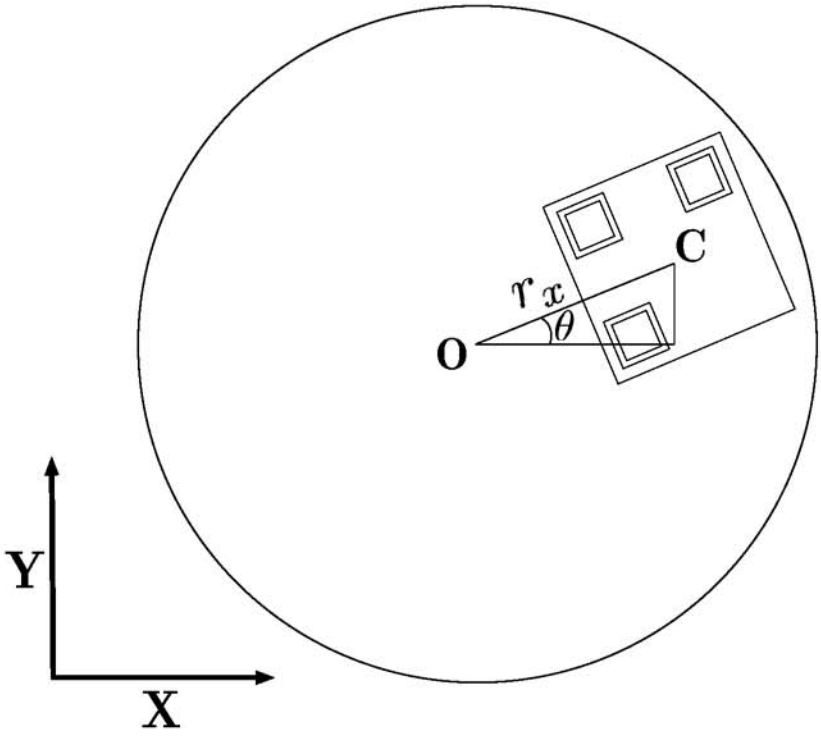


Figure 4.15: Determination of Cylindrical Obstacle Center From QR Code Position

Figure 4.15 shows how the center of the cylindrical obstacle is calculated from the QR code position and orientation. If the QR code center $C(x, y)$ is at a distance r_x from

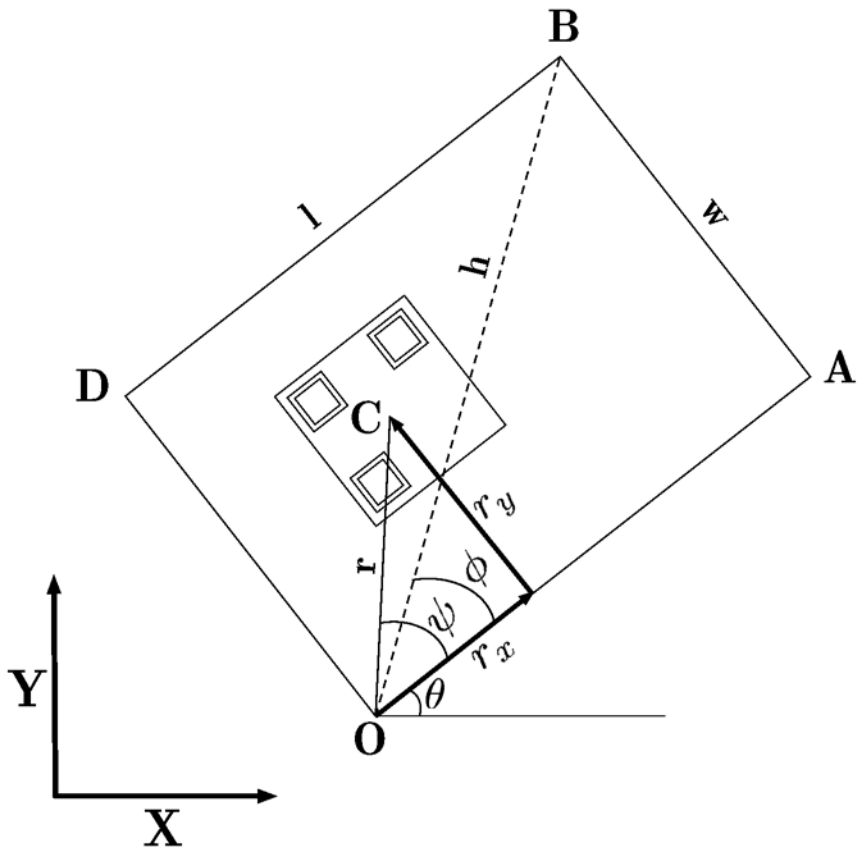


Figure 4.16: Determination of Rectangular Obstacle Vertices from QR Code Data

the center of the obstacle and perpendicular to the radius, and at an angle θ with the horizontal axis, then the obstacle center O is $(x - r_x \cos\theta,\ y - r_x \sin\theta)$. Once the position of the center of the obstacle is calculated, the obstacle can be drawn with the radius data read from the database file. The QR code can also be positioned at the center of the obstacle. In that case, r_x is zero.

To draw rectangular obstacles, all the vertices are calculated from the center position and orientation of the QR code. If the center of the QR code is $C(x, y)$ and it is at an angle of θ with the horizontal axis, then the vertices can be calculated as follows:

Vertex O:

$$x_o, y_o = x - r\cos(\psi + \theta),\ y - r\sin(\psi + \theta) \tag{4.1}$$

Vertex A:

$$x_a, y_a = x_o + l\cos\theta,\ y_o + l\sin\theta \tag{4.2}$$

Vertex B:

$$x_b, y_b = x_o + h\cos(\theta+\phi),\ y_o + h\sin(\theta+\phi) \tag{4.3}$$

Vertex D:

$$x_d, y_d = x_o - w\sin\theta,\ y_o + w\cos\theta \tag{4.4}$$

where,

$$r^2 = r_x^2 + r_y^2 \tag{4.5}$$

$$\psi = \tan^{-1}(r_y/r_x) \tag{4.6}$$

$$h^2 = l^2 + l^2 \tag{4.7}$$

and

$$\phi = \tan^{-1}(w/l) \tag{4.8}$$

Since in polygonal obstacles, the vertices are defined in polar coordinates relative to the QR code center, once the position of the QR code Center $C(x_o, y_o)$ and the QR code angle θ with respect to positive horizontal axis are read from the QR code data, the vertices can be calculated from the following formula:

$$(x, y)_i = x_o + r_i\cos(\psi_i+\theta),\ y_o + r_i\sin(\psi_i+\theta) \tag{4.9}$$

where r_i is the distance of i^{th} vertex from the QR code center, and ψ_i is the angle of the vertex with respect to the horizontal axis.

After all the vertices are found for the rectangle or polygon, it can be drawn by connecting the vertices with straight lines. Figure 4.18 shows an example workspace, a

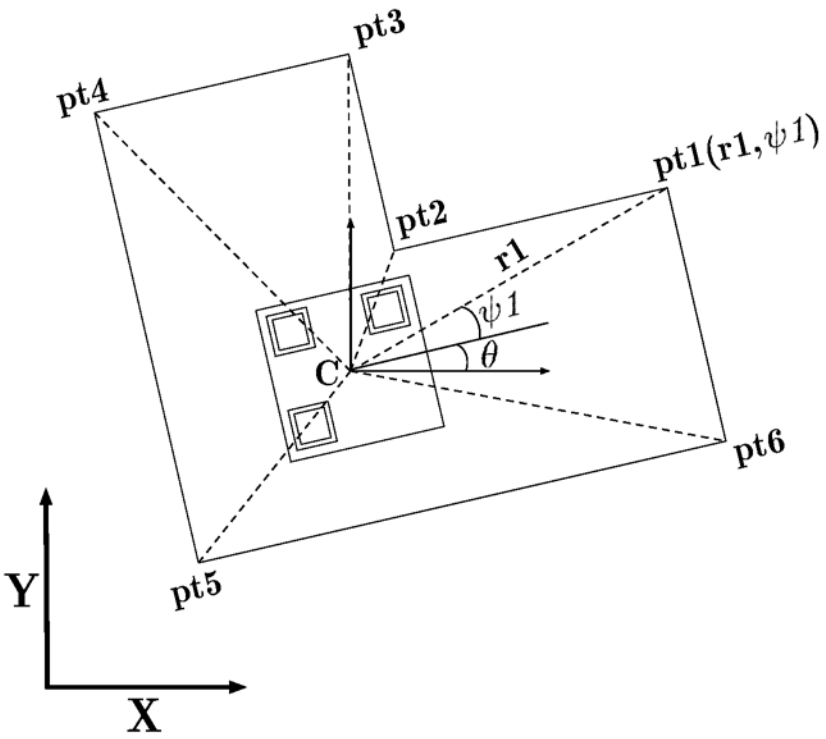


Figure 4.17: Determination of Polygonal Obstacle Vertices from QR Code Data

complete map is shown in Figure 4.19. Since the map is directly drawn from the dimensions specified in the database and the position and orientation of the obstacles, the accuracy of the map depends on the precision of measurements of the dimensions of the obstacles and the accuracy of placement of the QR codes. The distortion of the images captured by the camera, and the distortion of the stitched image also have a significant effect on the mapping accuracy.

4.2.7 Adding the Depth Information of the Obstacles

The map shown in Figure 4.19 shows only the top face of the obstacles. It is difficult to know the height and relative size of the obstacles from this map. It can be improved by adding the depth information of the obstacles. The height can be written in plain text on each obstacle, as shown in Figure 4.20. One way that a sense of depth of the obstacles can be given is by drawing the base of the obstacles. The obstacle bases in the map change size and color depending on their height. The base becomes smaller

Figure 4.18: Workspace with QR Code Labeled Obstacles

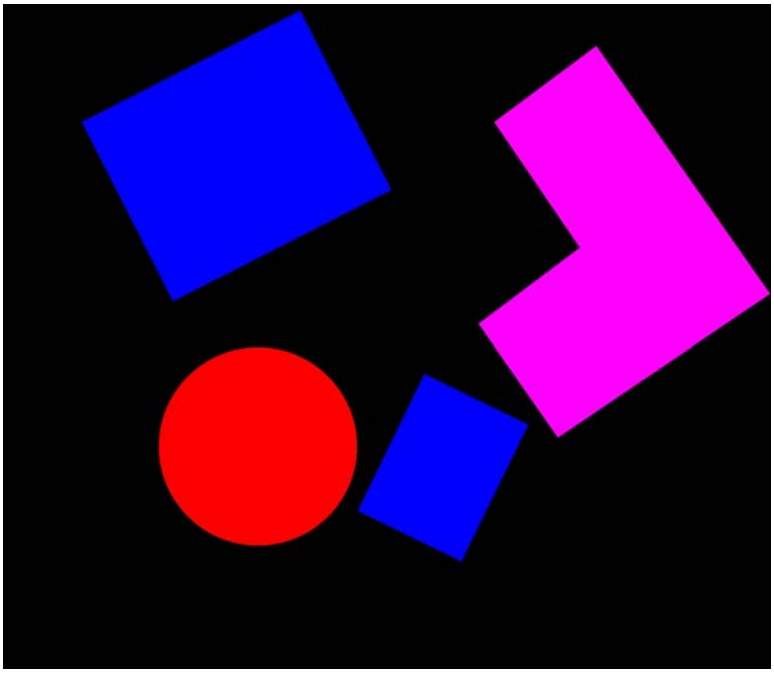


Figure 4.19: Resultant Map of the Workspace

and less bright with increasing height. The contrast and relative size between the top and the base give a sense of height of the obstacles.

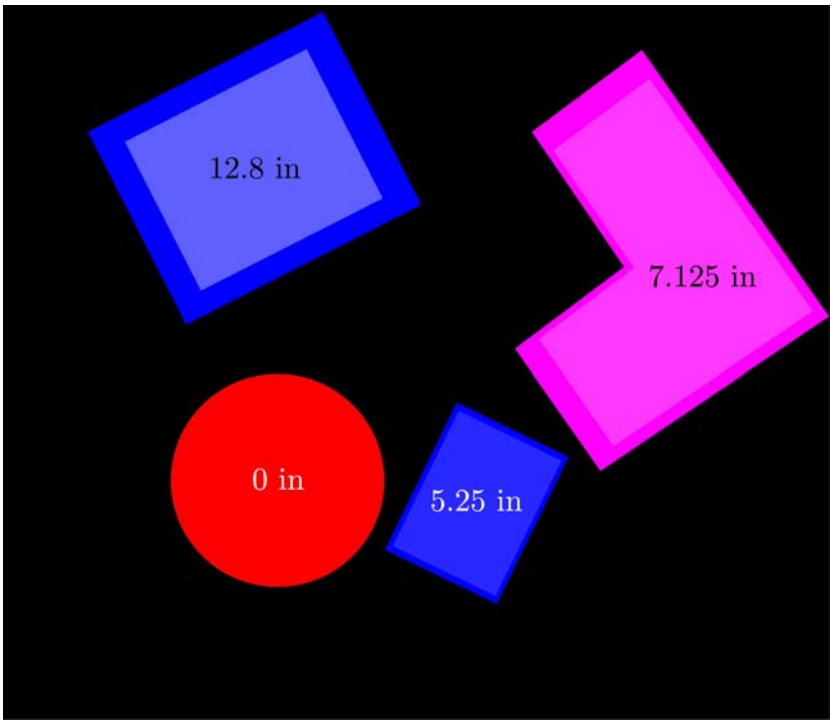


Figure 4.20: Workspace Map with Height Information

4.3 Combination with Machine-Vision-Based Mapping

The proposed QR-code-based method can be combined with the pure machine vision method introduced in Chapter III. A simple technique to combine these two maps is by drawing the QR code map directly over the machine-vision-based map. An example of this technique is shown in Figure 4.21, where the white areas behind the obstacles represent the machine-vision-based map. In this example, the QR code attached on the rectangular obstacle was slightly misplaced, so it shows slight discrepancy from the machine-vision-based map.

4.4 Performance Evaluation of QR Code-Based Mapping Algorithm

The performance of the mapping algorithm using QR codes was tested in different workspaces. Stitched images of these workspaces are shown in Figure 4.22. These workspaces have different backgrounds and different obstacle positions and spacing. The obstacles used for testing are of polygonal, rectangular, and circular shapes. The

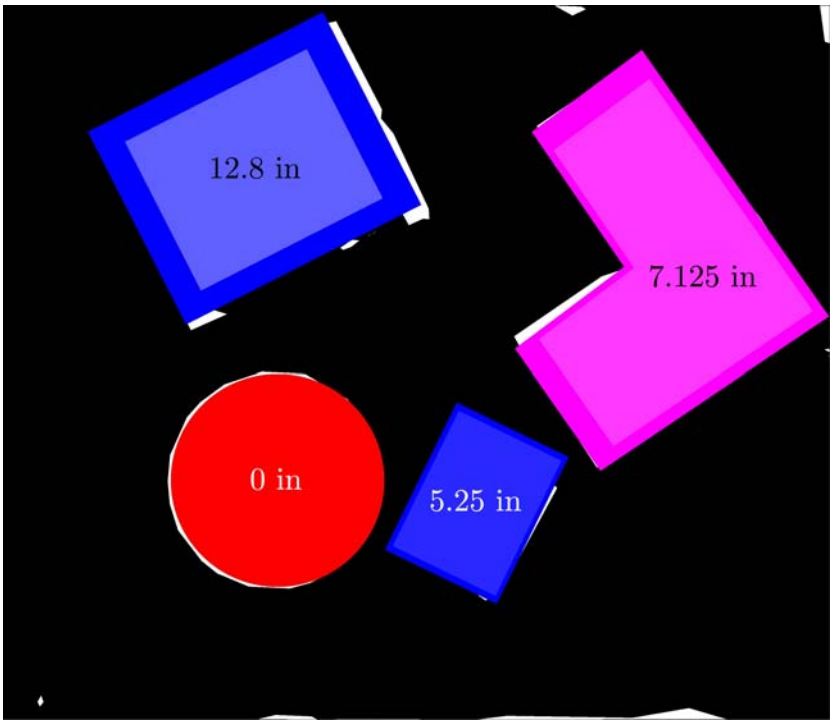


Figure 4.21: Workspace Map with Two Mapping Techniques Together

QR code maps are compared against manual maps drawn from the measurements of the workspaces. The coordinate of each vertex of each obstacle and the center in case of circular obstacles from an origin is measured, and the data are converted from inches to pixels. Then, the converted data are used to manually draw the top face of the obstacles.

Using this data, the total percentage of matched pixels, percentage of matched foreground pixels, percentage of missed obstacle pixels, and the percentage of falsely detected obstacle pixels are calculated. The percentage match plot shows what percentage of pixels of the manually-generated map match with the corresponding pixels of the QR code-based map, and the percentage foreground match plot shows what percentage of foreground pixels of the manually-generated map match with the corresponding pixels of the QR code-based map. These two plots indicate how good the map represents the actual workspace. The percentage missed obstacle pixels plot shows

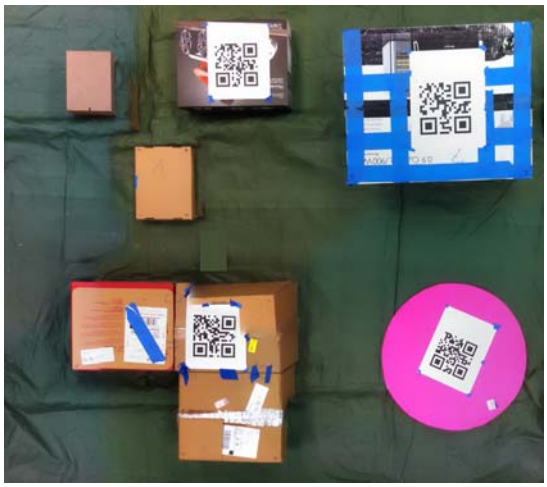
(a) Workspace 1 (b) Workspace 2 (c) Workspace 3

(d) Workspace 4 (e) Workspace 5 (f) Workspace 6

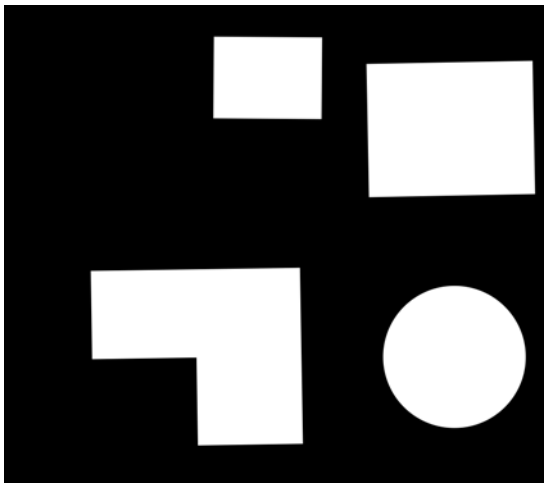
(g) Workspace 7

Figure 4.22: Workspaces Used for Evaluating QR Code Mapping Performance

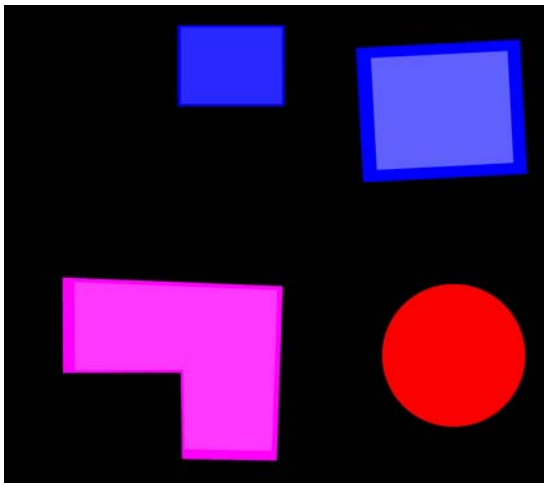
what percentage of foreground pixels in the manually-generated map are detected as a part of the background in the QR code-based map. The percentage falsely detected obstacles plot shows what percentage of background pixels in the manually-generated map are detected as a part of the foreground in the QR code-based map. Figure 4.23 shows an example workspace the corresponding manually-generated map and QR code map to be compared. Only the obstacles labeled with QR codes are considered for comparison. Some part of workspaces 6 and 7 are masked, because they are part of the



(a) Example Workspace



(b) Manually Drawn Map



(c) Map Using QR Code

Figure 4.23: Example Images for Evaluation

crane, not the workspace. The stitched image are of different sizes due to individual image resolution and stitching distortion differences. However, since the performance is evaluated as a percentage of total image pixels and total foreground pixels, the size variation doesn't affect the evaluation.

Figure 4.24 shows the percentage pixel match between the manually-generated map from the measurements of the workspace and the maps generated using QR codes. The results show as high as a 93% match between the manually-generated map and the QR code-based map. Figure 4.25 shows the percentage match of obstacle pixels of the QR

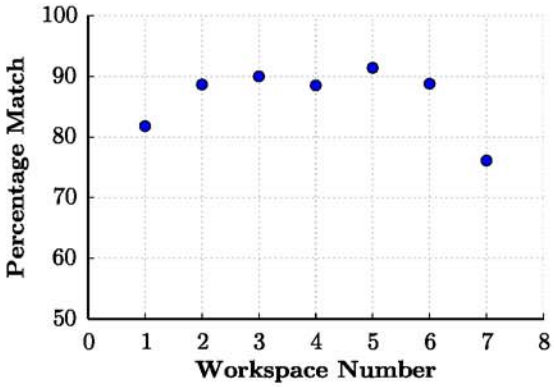


Figure 4.24: Percentage Match Between Actual Workspace and QR Code Map

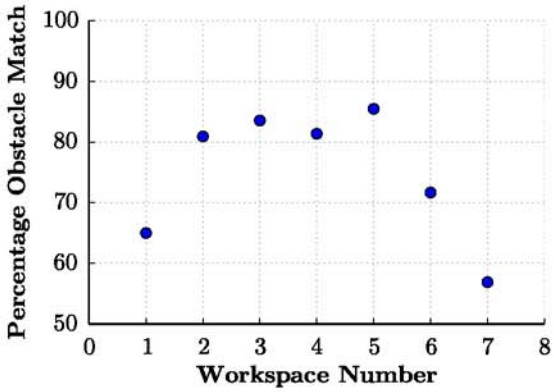


Figure 4.25: Percentage Obstacle Pixels Matched in QR Code Map

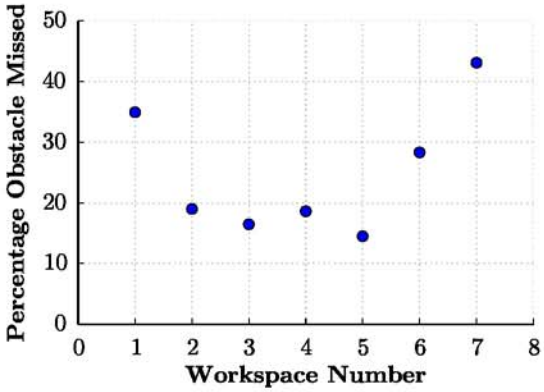


Figure 4.26: Percentage Obstacle Pixels missed in QR Code Map

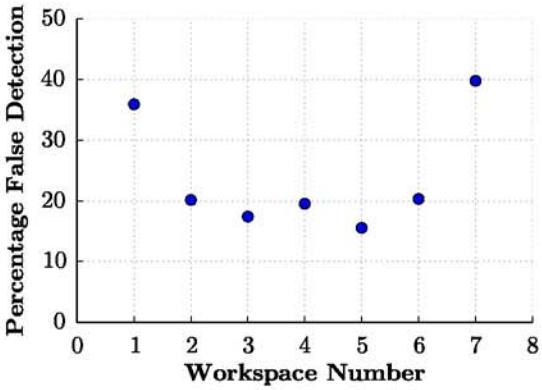


Figure 4.27: Percentage of Falsely Detected Obstacles in QR Code Map

code map compared to the manually-generated map. It shows the exact same pattern as Figure 4.24, which means the percentage match of obstacle pixels is almost proportional to the percentage match in total map. Figure 4.26 shows the percentage missed, and Figure 4.27 shows the percentage falsely detected obstacle pixels of the QR code map compared to the manually-generated map. From these two figures, it is clear that the percentage of missed obstacle pixels is proportional to the percentage of falsely detected obstacle pixels. This suggests that, if a map misses a high percentage of obstacles, then it also detects a high percentage of false obstacles. In all four figures it is noticeable that workspace 1 and workspace 7 have the highest amount of error. In these two cases there is a high distortion in stitching, resulting in a significant amount of de-

viation of the stitched image from the actual workspaces. Due to the distortion, the center and angle detection of the QR codes result in some errors, affecting the accuracy of the map.

4.5 Comparison of Mapping Techniques

The pure machine vision map, the QR code-based map, and a combination of these two maps are compared against the maps drawn from the physical measurements of the obstacles for the same workspaces shown in Figure 4.22. Obstacles both with and without QR codes are Considered for comparison in these workspaces. The percentage pixel match between actual map and manually-generated map is shown in Figure 4.28. The percentage match is close for the pure vision map, QR code-based map, and combination map in most cases, and there is no particular trend. However, Figure 4.29 shows that the percentage foreground match is higher for the combination map than the individual pure machine vision map and QR code map. The pure machine vision map shows slightly better match than the QR code map. The reason for that is that the obstacles without QR codes are not displayed in the QR code-based map, therefore only part of the actual workspace is shown in this map. This is also reflected on Figure 4.30 which compares the percentage obstacle pixels missed for the three kinds of map. This figure shows that the QR code-based map misses the most obstacle pixels. However, the previous section shows that the percentage missed pixels is reduced if the obstacles not labeled with QR codes are not considered for comparison. This figure also shows that the combination map misses fewer pixels that belong to an obstacle than the pure machine vision map and QR code-based map individually, which means the combination map is more reliable. However, Figure 4.31 shows that the combination map detects highest number of pixels as part of an obstacle where in reality they are not. It also shows that the pure vision map has the tendency of falsely detecting pixels as obstacles than the QR code-based map.

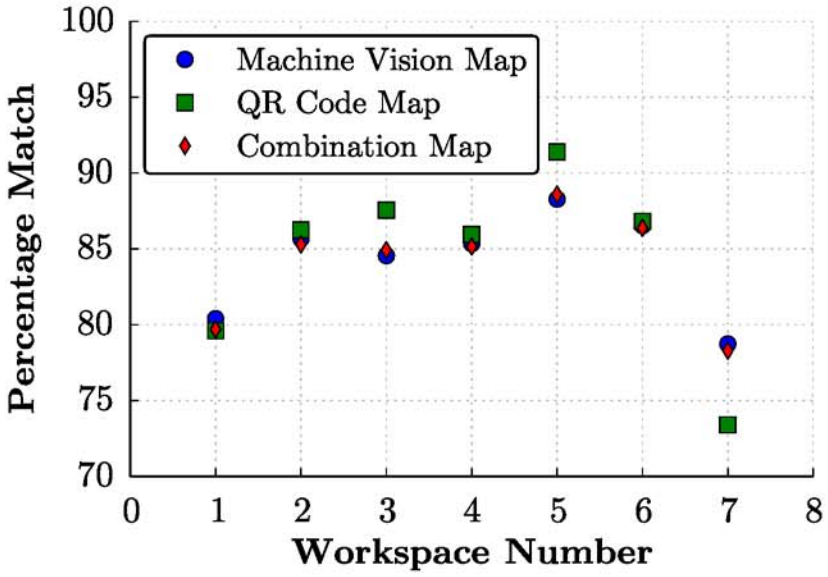


Figure 4.28: Percentage Pixel Match Between Actual Workspace and Generated Maps

For all three mapping techniques, errors are high for workspaces 1 and 7 because of the high distortion in stitched images. Another source of error is the assumption that each obstacle is viewed right from the top of the obstacles. The manual maps are drawn accordingly, by centering the obstacle top with the bottom. However, in reality that is not the case. The obstacles are viewed from an angle by the camera; therefore, the obstacle tops are skewed at different directions from the base.

It can be concluded from the analysis that, though the combination map has the tendency of falsely detecting greater number of pixels as obstacles, it detects the highest percentage of obstacle pixels and misses the lowest percentage of obstacle pixels. Falsely detecting obstacles is safer than missing obstacles in terms of safety in crane operation. Therefore, the combination map is more reliable, if not the most accurate, than the individual maps.

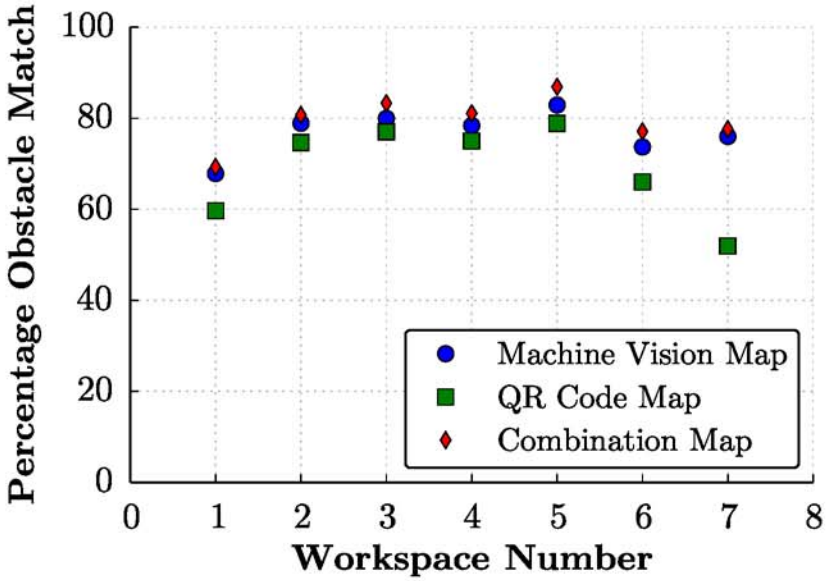


Figure 4.29: Percentage Obstacle Pixels Matched Between the Workspace and Generated Maps

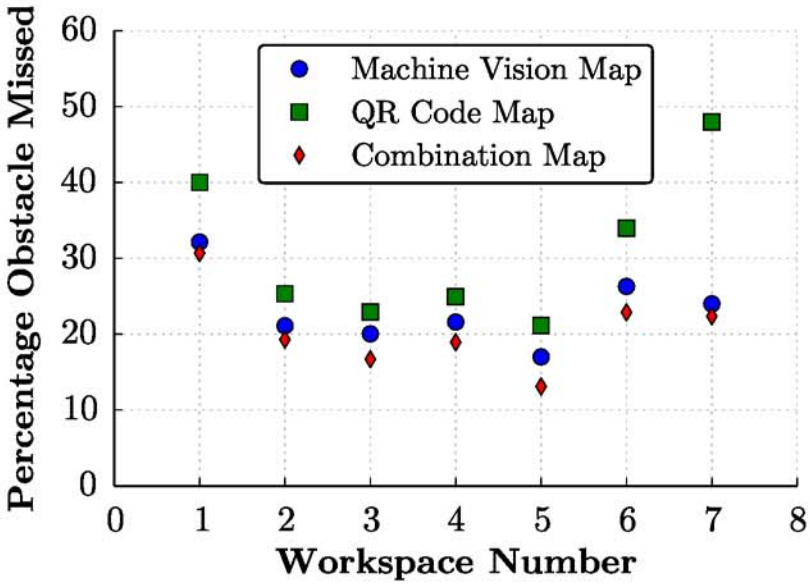


Figure 4.30: Percentage Obstacle Pixels Missed in Generated Maps

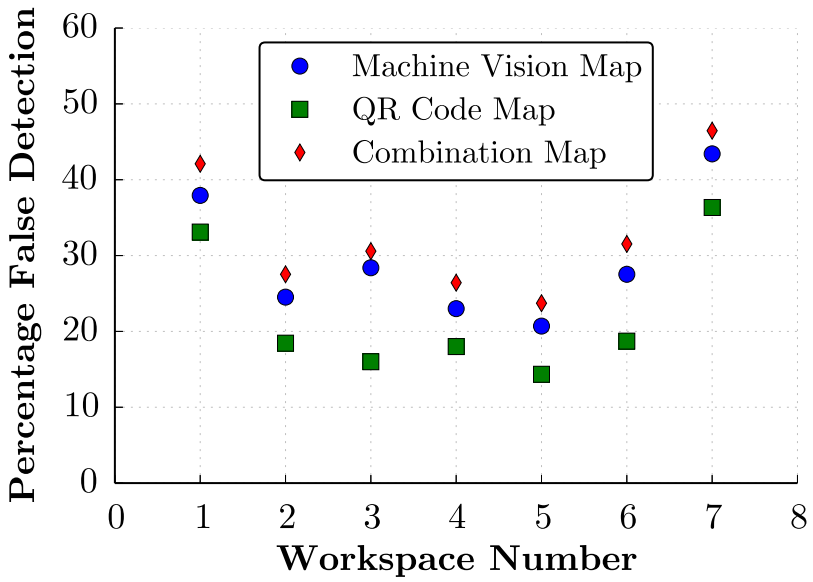


Figure 4.31: Percentage of Falsely Detected Obstacles in Generated Maps

4.6 Chapter Conclusion

In this chapter, an algorithm for mapping crane workspace using QR codes was introduced. All the steps of the mapping algorithm, including labeling of the obstacles with QR codes, creating database, encoding and decoding of the QR codes, angle and center calculation, and drawing the map were explained. The height of the obstacles was included in the drawn map. Then, the resultant map was integrated with the machine-vision-based map. The mapping performance of the QR code-based map was compared with the machine-vision-based map and the combination map. The efficiency of the QR code-based mapping algorithm was found to be as high as 93%. However, precise positioning of the QR codes and accurate measurements of the dimensions of the obstacles are important for optimum performance. If the QR codes are precisely positioned and clearly readable, and the measurements are accurate, the QR code-based mapping algorithm is preferable. However, it only works with QR code-labeled obstacles of known dimensions and therefore should be accompanied by the machine-vision-based mapping algorithm for a complete map of the crane workspace, unless all the

obstacles in the workspaces are labeled with a QR code.

Chapter 5

CONCLUSION

5.1 Summary and Contributions

In this thesis, first a novel approach of mapping the workspace of overhead cranes using pictures from a single camera was presented. Different image processing techniques used to accomplish the mapping, including image stitching, template matching, images segmentation, histogram calculation were discussed. The idea of displaying the older maps in the current map to show the previous obstacle positions as an indication of the likelihood of finding the obstacle at those positions again in the future was introduced. A memory factor was also introduced, which determines when to forget an old map. Two segmentation algorithms for obstacle detection, simple grayscale thresholding and watershed transform were compared. While the watershed transformation is more successful in detecting obstacles, it is also computationally expensive and the threshold values required to create the marker are difficult to determine.

In addition, a method of crane workspace mapping using QR codes was presented. The method for creating database, the labeling of the obstacles with QR codes, the encoding and decoding of the QR codes and the drawing of the obstacles on the map were discussed. The height information was included with the map to give it a 3D appearance, and the combination of the two mapping techniques were discussed. Then the performance of the mapping algorithm using QR codes was compared with the machine-vision-based map for different workspaces using a manually-generated map.

Image segmentation methods have applications in a number of different fields, such as

object detection and tracking. However, it has not been used in mapping crane workspaces before. This thesis shows the enormous potential of image segmentation methods to be used in mapping workspaces. Although image stitching techniques have been used before in aerial mapping, this is the first time the possibility of using image stitching together with image segmentation to formulate a crane workspace mapping algorithm has been explored.

Although QR codes were initially invented for storing product information, other applications of them have been introduced, such as using them as landmarks for localization of robots. This thesis presents an unique idea of using QR codes for mapping crane workspaces. The results obtained from this thesis demonstrate that QR codes have the potential of being widely used in mapping crane workspaces.

5.2 Future Work

The work in this thesis enables some advanced crane workspace mapping. More advanced image segmentation algorithms can be used to improve the mapping performance. Significant computation power can be saved by replacing the stitching with augmenting individual segmented images side-to-side. Instead of reading the QR codes from the stitched image, they could be read directly by the camera, and the obstacles could be drawn by locating the QR code in the camera coordinate combined with the position of the camera itself in the world coordinate.

The obstacle positions obtained from the map can be used for automatic obstacle avoidance by creating virtual boundaries around the obstacles, which in turn will make possible partial or fully-automated crane operation. This map can also be used to optimize the path for the crane, which will increase productivity by reducing travel distance and time.

BIBLIOGRAPHY

[1] [Online]. Available: http://www.cautructinnghia.com/

[2] [Online]. Available: http://www.cognex.com/products/machine-vision/in-sight-7000-series-integrated-vision-systems/

[3] Qrstuff. [Online]. Available:http://www.qrstuff.com/blog/2011/12/14/qr-code-error-correction

[4] Dsynflo. [Online]. Available:http://dsynflo.blogspot.com/2014/10/opencv-qr-code-detection-and-extraction.html

[5] B. of Labor Statistics. (2008, July) Crane-related occupational fatalities.

[6] Crane inspection and certification bureau. [Online]. Available:http://www.cicb.com/blog/posts/with-crane-related-injuries-on-the-rise-don-t-become-another-statistic.html

[7] O. J. M. Smith, *Feedback Control Systems*. New York: McGraw-Hill Book Co., Inc., 1958.

[8] N. C. Singer and W. P. Seering, "Preshaping command inputs to reduce system vibration," *Journal of Dynamic Systems, Measurement, and Control*, vol. Volume 112, pp. 76–82, March 1990.

[9] W. Singhose, N. Singer, and W. Seering, "Time-optimal negative input shapers," *Journal of Dynamic Systems, Measurement, and Control,*, vol. 119, pp. 198–205, June 1997.

[10] N. Singer, W. Singhose, and E. Kriikku, "An input shaping controller enabling cranes to move without sway," in *ANS 7th Topical Meeting on Robotics and Remote Systems*, vol. 1, Augusta, GA, 1997, pp. 225–31.

[11] K. Sorensen, W. Singhose, and S. Dickerson, "A controller enabling precise positioning and sway reduction in bridge and gantry cranes," *Control Engineering Practice*, vol. 15, no. 7, pp. 825–837, July 2007.

[12] A. Khalid, W. Singhose, J. Huey, J. Lawrence, and D. Frakes, "Study of operator behavior, learning, and performance using an input-shaped bridge crane," in *Proceedings of the 2004 IEEE International Conference on Control Applications*, vol. 1, September 2004, pp. 759–764.

[13] D. Kim and W. Singhose, "Performance studies of human operators driving double-pendulum bridge cranes," *Control Engineering Practice*, vol. 18, no. 6, pp. 567–576, June 2010.

[14] Camotion, inc. [Online]. Available: http://www.camotion.com

[15] [Online]. Available: http://www.konecranes.com/equipment/overhead-cranes/smart-features

[16] C. Kim, C. T. Haas, K. A. Liapi, and C. H. Caldas, "Human-assisted obstacle avoidance system using 3d workspace modeling for construction equipment operation," *Journal of computing in civil engineering*, vol. 20, no. 3, pp. 177–186, May 2006.

[17] D. Murray and C. Jennings, "Stereo vision based mapping and navigation for mobile robots," in *Proceedings of the IEEE International Conference on Robotics And Automation*, vol. 2, April 1997, pp. 1694–1699.

[18] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *EEE Transactions on Robotics*, vol. 30, no. 1, February 2014.

[19] C. Stiller, J. Hipp, C. Rossig, and A. Ewald, "Multisensor obstacle detection and tracking," *Image and Vision Computing*, vol. 18, no. 5, pp. 389–396, April 2000.

[20] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, November 2007, pp. 225–234.

[21] ——, "Parallel tracking and mapping on a camera phone," in *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality*, October 2009, pp. 83–86.

[22] S. Thrun and J. Leonard, *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.

[23] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 229-241, June 2001.

[24] A. Shapira, Y. Rosenfeld, and I. Mizrahi, "Vision system for tower cranes," *Journal of Construction Engineering and Management*, vol. 134, no. 5, pp. 320–332, May 2008.

[25] J. Yang, P. Vela, J. Teizer, and Z. Shi, "Vision-based tower crane tracking for understanding construction activity," *Journal of Computing in Civil Engineering*, vol. 28, no. 1, pp. 103–112, January 2014.

[26] Y. Yoshida and K. Tsuzuki, "Visual tracking and control of a moving overhead crane load," *9th IEEE International Workshop on Advanced Motion Control*, pp. 630–635, 2006.

[27] T. Miyoshi, K. Tsuchida, and K. Terashima, "Obstacle avoidance control for overhead crane with rotary motion of load," in *SICE Annual Conference in Fukui*, August 2003.

[28] S. Nagai, A. Kaneshige, and S. Ueki, "Three-dimensional obstacle avoidance online path-planning method for autonomous mobile overhead crane," in *International Conference on Mechatronics and Automation (ICMA)*, August 2011, pp. 1497–1502.

[29] C. Yuan Been, C. Oscal T-C *et al.*, "Image segmentation method using thresholds automatically determined from picture contents," *EURASIP Journal on Image and Video Processing*, 2009.

[30] O. J. Tobias and R. Seara, "Image segmentation by histogram thresholding using fuzzy sets," *Image Processing, IEEE Transactions on*, vol. 11, no. 12, pp. 1457–1465, Dec 2002.

[31] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using kalman-filtering," in *Proceedings of International Conference on recent Advances in Mechatronics*. Citeseer, 1995, pp. 193–199.

[32] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 11, pp. 1101–1113, 1993.

[33] A. E. Savakis, "Adaptive document image thresholding using foreground and background clustering," in *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*. IEEE, 1998, pp. 785–789.

[34] T. Uemura, G. Koutaki, and K. Uchimura, "Image segmentation based on edge detection using boundary code," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 10, pp. 6073–6083, 2011.

[35] R. Adams and L. Bischof, "Seeded region growing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 6, pp. 641–647, Jun 1994.

[36] F. Y. Shih and S. Cheng, "Automatic seeded region growing for color image segmentation," *Image and Vision Computing*, vol. 23, no. 10, pp. 877–886, Sepetember 2005.

[37] A. Mehnert and P. Jackway, "An improved seeded region growing algorithm," *Pattern Recognition Letters*, vol. 18, no. 10, pp. 1065–1071, 1997.

[38] S. Beucher, *SCANNING MICROSCOPY-SUPPLEMENT*, 1992.

[39] S. Beucher and C. Lantuéjoul, "Use of watersheds in contour detection," 1976.

[40] A. Hanbury, "Image segmentation by region based and watershed algorithms," *Wiley Encyclopedia of Computer Science and Engineering*, 2008.

[41] M. Gonzalez and V. Ballarin, "Automatic marker determination algorithm for watershed segmentation using clustering," *Latin American applied research*, vol. 39, no. 3, pp. 225–229, 2009.

[42] I. Pratikakis, I. Vanhamel, H. Sahli, B. Gatos, and S. Perantonis, "Unsupervised watershed-driven region-based image retrieval," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 153, no. 3, pp. 313–322, June 2006.

[43] M. Meng, L. Fan, and L. Liu, "A comparative evaluation of foreground/background sketch-based mesh segmentation algorithms," *Computers & Graphics*, vol. 35, no. 3, pp. 650–660, 2011.

[44] J. Pont-Tuset and F. Marques, "Measures and meta-measures for the supervised evaluation of image segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, IEEE. IEEE, 2013, pp. 2131–2138.

[45] M. Borsotti, P. Campadelli, and R. Schettini, "Quantitative evaluation of color image segmentation results," *Pattern recognition letters*, vol. 19, no. 8, pp. 741–747, 1998.

[46] Y. J. Zhang, "Evaluation and comparison of different segmentation algorithms," *Pattern recognition letters*, vol. 18, no. 10, pp. 963–974, 1997.

[47] ——, "A survey on evaluation methods for image segmentation," *Pattern recognition*, vol. 29, no. 8, pp. 1335–1346, 1996.

[48] H. Kobayashi, "A new proposal for self-localization of mobile robot by self-contained 2d barcode landmark," in *SICE Annual Conference*. Akita University, Akita, Japan: SICE, August 20-23 2012.

[49] G. Lin and X. Chen, "A robot indoor position and orientation method based on 2d barcode landmark," *Journal of Computers*, vol. 6, no. 6, June 2011.

[50] T. Suriyon, H. Keisuke, and B. Choompol, "Development of guide robot by using qr code recognition," in *The Second TSME International Conference on Mechanical Engineering*, vol. 21, 2011.

[51] J. tung Wang, C.-N. Shyi, T.-W. Hou, and C. Fong, "Design and implementation of augmented reality system collaborating with qr code," in *Computer Symposium (ICS), 2010 International*, Dec 2010, pp. 414–418.

[52] H.-W. Liu and Y. Yan, "Recognition and decoding of qr code," *Jisuanji Gongcheng yu Sheji(Computer Engineering and Design)*, vol. 26, no. 6, pp. 1560–1562, 2005.

[53] L. Zhao-lai, H. Ting-lei, W. Rui, and Z. Xiao-yan, "A method of image analysis for qr code recognition," in *Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on*. IEEE, 2010, pp. 250–253.

[54] Y. Kato, D. Deguchi, T. Takahashi, I. Ide, and H. Murase, "Low resolution qr-code recognition by applying super-resolution using the property of qr-codes," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, Sept 2011, pp. 992–996.

[55] A. Sun, Y. Sun, and C. Liu, "The qr-code reorganization in illegible snapshots taken by mobile phones," in *Computational Science and its Applications, 2007. ICCSA 2007. International Conference on.* IEEE, 2007, pp. 532–538.

[56] Y. Liu, J. Yang, and M. Liu, "Recognition of qr code with mobile phones," in *Control and Decision Conference, 2008. CCDC 2008. Chinese.* IEEE, 2008, pp. 203–206.

[57] Y.-H. Chang, C.-H. Chu, and M.-S. Chen, "A general scheme for extracting qr code from a non-uniform background in camera phones and applications," in *Multimedia, 2007. ISM 2007. Ninth IEEE International Symposium on.* IEEE, 2007, pp. 123–130.

[58] L. F. Belussi and N. S. Hirata, "Fast qr code detection in arbitrarily acquired images," in *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on.* IEEE, 2011, pp. 281–288.

[59] [Online]. Available: http://opencv.org/

[60] Zbar. [Online]. Available: http://zbar.sourceforge.net/

[61] M. S. Rahman and J. Vaughan, "Simple near-realtime crane workspace mapping using machine vision," in *Proceedings of Dynamic Systems and Control Conference.* San Antonio, TX: ASME, October 2014.

[62] A. Saalfeld, "Topologically consistent line simplification with the douglas-peucker algorithm," *Cartography and Geographic Information Science*, vol. 26, no. 1, pp. 7–18, 1999.

[63] F. Jurie and M. Dhome, "A simple and efficient template matching algorithm," in *Proceedings of the Eight IEEE International Conference on Computer Vision*, vol. 2, 2001, pp. 544–549.

[64] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007.

[65] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

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

[67] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *Vision algorithms: theory and practice*. Springer, 2000, pp. 298–372.

[68] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1101–1113, August 2002.

[69] S. Zhu and A. Yuille, "Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, pp. 884–900, September 1996.

[70] Q. Huang, B. Dom, D. Steele, J. Ashley, and W. . Niblack, "Foreground/background segmentation of color images by integration of multiple cues," *International Conference on Image Processing*, vol. 1, pp. 23–26, October 1995.

[71] L. J. Belard and W. Mourou, "Image segmentation: A watershed transformation algorithm," *Image Analysis and Stereology*, vol. 28, no. 2, pp. 93–102, 2011.

[72] J. F. Barrera, A. Mira, and R. Torroba, "Optical encryption and qr codes: secure and noise-free information retrieval," *Optics express*, vol. 21, no. 5, pp. 5373–5378, 2013.

[73] Denso. Qr code essentials.

[74] M. Sudan, "Decoding of reed solomon codes beyond the error-correction bound," *Journal of complexity*, vol. 13, no. 1, pp. 180–193, 1997.

[75] qrcode 5.1. [Online]. Available: https://pypi.python.org/pypi/qrcode

[76] J. F. Canny, "Finding edges and lines in images," *Massachusetts Inst. of Tech. Report*, vol. 1, 1983.

Rahman, Mohammad Sazzad. Bachelor of Science, Bangladesh University of Engineering and Technology, Spring 2011; Master of Science, University of Louisiana at Lafayette, Spring 2015
Major: Mechanical Engineering
Title of Thesis: Machine Vision Techniques for Crane Workspace Mapping
Committee Chair: Dr. Joshua Vaughan
Pages in Thesis: 96; Words in Abstract: 242

ABSTRACT

Cranes are used worldwide for transportation and material handling in a variety of industries and facilities, including manufacturing industries, shipyards, and warehouses. Safety and efficiency in crane operations are a concern, since these issues are closely related to productivity. One of the reasons for crane-related accidents is mistakes by the operator, some of which can be attributed to the limitations of the operator?s field of view, depth perception, and knowledge of the workspace. These limitations are exacerbated by the dynamic environment of the workspace. One possible solution to these problems could be aiding the operator with a dynamic map of the workspace that shows the position of obstacles within it. In this thesis, two methods for mapping the crane workspace in near-realtime using computer vision are introduced. Several computer vision algorithms are integrated, and new techniques are introduced to generate a machine-vision-based map. A QR code-based mapping algorithm is also formulated. The algorithms can work independently. However, they can also be integrated, and the results show that a combination of these two mapping techniques produce the best results. The success of the pure machine-vision-based map and the QR code-based map depend on successful segmentation of color regions and detection of the QR codes, respectively. The combination of the two algorithms is a novel approach that ensures maximum obstacle detection. The algorithms produce a workspace map that can help the crane operator drive the crane more safely and efficiently.

BIOGRAPHICAL SKETCH

Mohammad Sazzad Rahman was born and grew up in Chittagong, at the south-eastern part of Bangladesh. He is the second child of his mother, Hasina Begum. He attended Bangladesh University of Engineering and Technology, Dhaka where he earned a Bachelor of Science in Mechanical Engineering. He began his graduate studies in Mechanical Engineering at the University of Louisiana at Lafayette in Fall 2013. His research interests are Controls and Robotics.