January 2017

Generation Of An Accurate, Metric Spatial Database Of A Large Multi Storied Building

Ahmad Jarjis Hasan

GENERATION OF AN ACCURATE, METRIC SPATIAL DATABASE OF A LARGE MULTI STORIED BUILDING

by

Ahmad Jarjis Hasan
Bachelor of Science, Khulna University of Engineering & Technology

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Master of Science

Grand Forks, North Dakota
June
2017

This thesis, submitted by Ahmad Jarjis Hasan in partial fulfillment of the requirements for the Degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

_____
Dr. Jeremiah Neubert, chair

_____
Dr. William Semke

_____
Dr. Cai Xia Yang

This thesis is being submitted by the appointed advisory committee as having met all of the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

_____
Grant McGimpsey,
Dean of the Graduate School

_____
Date

PERMISSION

Title Generation of an Accurate, Metric Spatial Database of a Large Multistoried Building

Department Mechanical Engineering

Degree Master of Science

In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work or, in his absence, by the chairperson of the department or the dean of the Graduate School. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my thesis.

Ahmad Jarjis Hasan
June 26th , 2017

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I express my gratitude to the Department of Mechanical Engineering of University of North Dakota, for giving me the opportunity to perform research and pursue my master degree here. My solemn gratitude to Dr. Jeremiah Neubert who has been a mentor and guide to me. While being my supervisor in this academic period, he has enriched and trained me with vast knowledge and competitive skills.

I am very much grateful to the committee members, Dr. Willaim Semke and Dr. Cai Xia Yang for their continuous support and advice.

I am also very grateful to Dr. Ashraf Qadir for guiding me and giving valuable suggestions throughout my research. Thanks to Ian Nordeng and Akkas Uddin Haque for reviewing the work and for helping with the data acquisition of the construction site.

To my parents – my inspiration behind all my good works

ABSTRACT

This thesis presents the development of a novel method to generate an accurate, metric spatial database of a large multi storied building during construction. The algorithm uses the 3D CAD model of the building and the video of the structure captured by an Unmanned Aircraft System (UAS). The spatial database is then used to perform several inspection procedures such as, metric data analysis, spatial query for images, visualization through 3D textured model. The video is processed using a simultaneous localization and mapping (SLAM) system. SLAM generates a sparse 3D map of the environment. Our algorithm registers the 3D map with the 3D CAD model to generate the accurate metric spatial database. The user can click on the desired part of the CAD model for inspection and the image of that part will be shown by using the spatial indexing between the CAD model and the spatially distributed images. The image returned by the spatial query can be used to extract metric information. The spatial database is also used to generate a 3D textured model which provides a visual as-built documentation. The metric data calculation and textured model reconstruction methods have been compared to the state of the art Pix4D software (Latest Release (Version 3.1)). The proposed method has a mean squared error (MSE) of 31.9 cm^2 and standard deviation of 4.28 cm where Pix4D had a higher MSE of 45.6 cm^2 and standard deviation of 4.91 cm. Using statistical t-test and ANOVA tests we have shown that we are statistically 99% confident that the proposed algorithm has performed better than Pix4D.

CHAPTER I

INTRODUCTION

Generation of as-built documentation requires engineers to search through large amount of video of the construction site for a specific region of interest. This is time-consuming and a very inefficient method. The objective of the proposed system is to reduce the amount of labor and time required to collect important construction site information, and enable engineers to generate as-built documentation of building elements. The proposed system provides a credible solution to that problem by aligning a SLAM generated 3D map to the 3D CAD model generating an accurate metric 3D spatial database. The registration facilitates spatial query for the images from the spatial database. This is done by allowing the user to click on the region of interest on the 3D CAD model.

The system takes a previously generated 3D CAD model and the images collected with the UAS which may be processed by a variety of keyframe based SLAM [1,2,3] systems as input. The SLAM system is used to estimate camera poses and generate a 3D point cloud map of the structure. Each map point in the point cloud stores the 3D pose in world coordinate m^w, its index, and a reference to the index of the source keyframe where it was first detected. Tagging the index of the 3D point to the index of the keyframe provides the spatial index of the database. The proposed system takes the point cloud map and aligns it to the 3D CAD model to update the geometry of the spatial database. This allows the user to perform a spatial query through the 3D CAD model by use of mouse clicks on the model to search for the desired images. The proposed system also provides a metric data analysis tool to analyze that queried image. Finally, a textured 3D model of the structure is generated to serve as an overall visual as-built documentation of the structure.

Building inspection for quality assurance often involves analyzing images and detecting any anomaly or defect in the construction. For a multi-storied large building, it is necessary that the images are collected from a close range for proper visualization and inspection. Using a UAS is one way that allows users to collect high resolution building images with better fidelity. It is necessary, as ground based images would not be able to provide the required fidelity to perform accurate building inspections. This work acknowledges the necessity and increasing use of small UAS in construction process and opens a new door for computer vision based systems to become more intimate with this advancement. There have been more than a million small UAS sold in the United States over the past few years per news reports [4, 5]. As the use of small UAS grows the need for cost-effective methods for accessing and processing data will grow. The goal of this project is to address that need by creating novel technologies to facilitate the use of small UAS to inspect structures as they are built.

Our contributions are listed below:

Accurate Metric Spatial Database Generation: The proposed system generates an accurate metric spatial database by taking the 3D CAD model of the building and a video of the building as an input. The video is processed by a SLAM system to generate a 3D map. The SLAM generated map is registered to the 3D CAD model to produce the spatial database. The spatial database is used for inspection purposes. The spatial database has metric units which enables users to extract metric information from the database.

Spatial Query for Images: The generated spatial database allows a user to perform an efficient spatial query for images by clicking on the 3D CAD model. The user may easily visualize the 3D CAD model, making our implementation convenient for anyone locate images of a particular

feature. The user simply clicks on the 3D model to find the relevant image without having to search through time intensive videos.

Metric Data Analysis: Our system accurately calculates the distance between any two places of the structure, and gives a comparison between the 3D CAD model distance and the as-built structure. A 3D CAD model- image correspondence is used to ensure the robustness of the calculation.

Visualization through 3D textured model: The proposed system also generates a 3D textured model with high-resolution images to create a virtual reality of the scene. The virtual reality is visualized with an OpenGL window implemented with a moving camera to allow the user to move around the scene. The 3D textured model enables engineers or facility mangers visually assess the building and track construction progress.

To understand the need of engineers, our team worked with construction engineers who helped with a detailed checklist of specific requirements for inspection. The system was developed to solve the problems identified in the checklist.

Contractors are already using small UAS to gather information about their worksite and inspect structures. In [6, 7] small UASs are used to construct a detailed 3D map of work sites. Others have used small UAS to inspect existing structures [7]. In [6] they used a UAV from 75 meters elevation and only used 64 images which is not enough to get accurate measurement. To have a usable spatial query for images is not possible because of the low number of images too. Their approach gives an elevation map using orthophotos which is 2.5D. The height map is scale consistent but is not absolute. In [8] they outline the potential applications of UAS in the new construction such as monitoring the build process, creating “as-built” documentation and automated defect detection. However, unfortunately they do not mention how much error do they have and mainly explain

how to use the software. They used a laser scanner to collect the point cloud which is registered to the ground truth model. As they used laser scanner, the do not have a visual as built documentation and cannot do spatial query through images nor can the extract metric data from images. Laser scanners have been a popular tool for construction quality control as in [9, 10] and tracking various components [11,12]. They have also been used for progress monitoring in [13,14]. Researchers are currently working on creating the algorithms needed to exploit this potential. One such system presented in [15] was used to aid in the creation of "as-built" documentation. Their work generated reasonable output, but the dimensions had more than 5% error.

The D4AR modeling [16] uses an unordered collection of images of the structure to generate the underlying geometric model by using a Structure-from-Motion (SfM). They solve the similarity transform between the model and 3D point cloud found from the SfM using minimum user inputs to transform the SfM coordinate system to the 3D CAD model's coordinate system to allow the aligning the SfM photos to the CAD model. They do not perform a spatial query for images too and do not extract measurement data. There are popular methods being used to create 3D mesh models by using images. Photo Tourism [17] has used unordered photo collection to create a mesh model. The system has not been tested on a large-scale structure. There are other limitations too such as, they mentioned that their system does not guarantee production of metric scene reconstruction. The system also does not consider lens distortion which generates more errors. In [18] Debevec et al proposed a modeling and rendering pipeline for architectural structures from photographs. The purpose of that research was visualization and does not handle metric data calculation. Sinha et. al [19] took such an approach for scene reconstruction. They mention that, their automatic view selection did not work all the time and users had to manually assign additional source images which is a significant disadvantage. This system is also used for visualization

purposes and does not extract images from spatial data and do not provide metric data output. In [20] Xu et al. proposed a photo-inspired model driven method for 3D object modeling. Their algorithm is shown to work on small models. Another disadvantage of their system is, the user must pick a base image from a set of images and manually segment the desired object out fro the first frame. This is almost an impossible task for a large-scale structure. Such methods have been used in [21] too.

A different approach was taken by [22]. Instead of creating 3D mesh model using images, they rely on an existing semantic 3D CAD model known as Building Information Model (BIM). This modeling is widely available nowadays to facilitate easier construction as it provides prior detailed information about the building or structure to be constructed [22].

However, none of the methods above does an image query for analysis of as-built structure. L. Klein et al. [15] is an effective step towards metric data analysis of as-built structures.

Methods described in [16, 17, 18, 19, 20, 21] use images to create underlying geometry. Such methods are not useful for construction sites because it only provides visual information of an already built structure. Construction sites require a pre-designed CAD model to be able to have a ground truth for the construction and compare it to the as-built structure to detect anomalies. Considering this, our method uses a 3D CAD model designed with AutoCAD Revit as [22].

In this work, out design considerations are governed by the need of the engineers on a construction site. The checklist provided by the construction engineers specifically pointed that, the need for an accurate metric data analysis is a priority. Another important tool required in that checklist was the need for a user-friendly tool for searching for originally taken images through the 3D CAD model. Our system enables the user to search for images through spatially indexing images onto the model. Metric data analysis and crack detection might be performed on the queried

image through our provided tools. Moreover, our visualization tool enables the user to move freely throughout the scene providing a better visualization experience.

CHAPTER II

GENERATION OF AN ACCURATE, METRIC SPATIAL DATABASE OF A LARGE MULTI STORIED BUILDING

2.1 System Overview

Figure 2.1 outlines the system. First the visual data is read frame by frame from a video as sequential images. The user specifies the region of interest for inspection from the model. The system leverages user input to register the first keyframe to the 3D CAD model. The proposed system prompts the user to specify a four-point correspondence between the model and the 1st keyframe at the beginning to aid the registration process. This registration aids in finding the pose of the UAS in model coordinate by solving a Perspective-Three-Point (p3p) problem. The visual input is then processed by a SLAM system. The SLAM system generates a scalable point cloud map from the visual input. The SLAM generated 3D map has a point cloud which is refined with a random sample consensus (RANSAC) based plane-fitting algorithm. A similarity transform is applied to the 3D map to transform the map to the 3D CAD model coordinate system. All the images are then aligned with the 3D mesh model to update the geometry of the spatial database. The updated spatial database uses the spatial index to run a query for images of the desired region which is specified by the user. The image returned from the spatial query may further be used to evaluate distance in the as-built structure. For accurate metric calculation, the proposed system, runs an automated window detection and updates scale of the queried area by taking window size information from the 3D CAD model.

To generate the 3D textured model, the proposed system finds images on the planes and stitches them together to texture map those images to the specified zone of the textured model. The metric

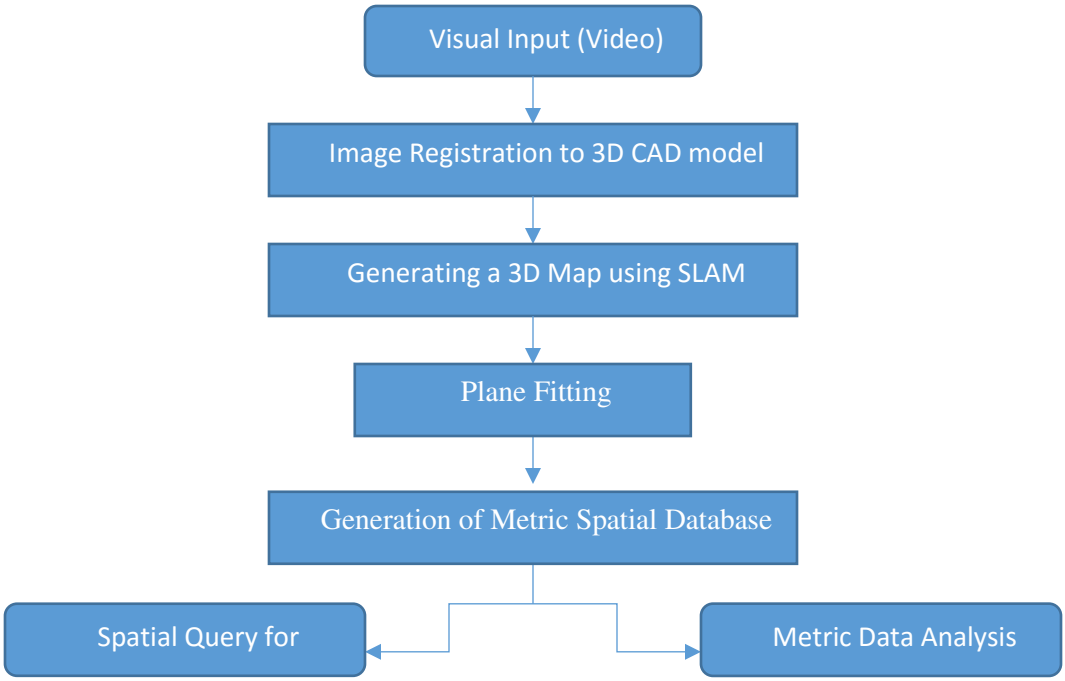

Figure 2.1: The process flow chart for system overview.

information is mainly intended to find dimensions of various entities in the as-built structure. The user must click on the different corners of the model to inspect the size of that specific entity i.e. windows, doors, the length of the column etc. After clicking on the model, the corresponding image of that area will be seen, and the user will specify the desired dimension with mouse clicks. A user guide has been provided in the APPENDIX, for the convenience of the users.

2.2 Methodology

The proposed system requires eight manual clicks from the user as inputs for creating the initial correspondence between the CAD model and the first image. In this section, the methodology of the system has been described. The image registration process, generation of metric spatial database, and spatial query for images has been explained in detail. A plane fitting is performed on the registered point cloud to refine the spatial database which facilitates further processing such as 3D textured reconstruction, metric information extraction, and crack detection.

(a) (b)

Figure 2.2: Image registration to 3D CAD model (a) Selected four corners of the window shown in red from CAD model (b) Corresponding four corners selected in the first keyframe of the red marked window.

2.2.1 Image Registration to 3D CAD Model

The location and orientations of the camera are arbitrary in the camera or SLAM coordinate system. In this case the CAD model provides the ground truth and a global coordinate system. To facilitate alignment of SLAM and 3D CAD model coordinate systems our system uses an input from the user. The user must provide four 3D-2D point correspondences between the 3D CAD model and the first keyframe. The approach described in [23] is used to solve the p3p problem to determine the position and orientation of the camera while capturing the first keyframe in the model coordinate system. The solution described in [23] for the p3p algorithm provides up to four solutions which are disambiguated by using a fourth point. The user clicks on four corners of an entity such as a window from the model. That entity must be seen in the first keyframe so that the user can then click on corresponding four corners of the same entity from the image in a sequential order. Figure 2.2 shows a representation of the process.

2.2.2 Generating a 3D Map using SLAM:

In a large-scale application, such as this one, range sensors such as RGB-D cameras, stereo cameras or laser range finders are not very effective because of limited range. To tackle this

problem a monocular Simultaneous Localization and Mapping system has been used. However, the monocular SLAM system comes with its own challenges. The main issue with monocular SLAM systems is that the metric scale cannot be measured with a single camera. To recover scale additional metric information must be provided by another sensor or by the user. The SLAM system continuously updates the 6 degrees of freedom camera pose while creating a keyframe based parallel tracking and mapping framework. The generated 3D point cloud map consists of a collection of feature points M in the world coordinate system W. Each map point stores the 3D world coordinate m^w, its index, a reference to the source keyframe where it was first detected along with the indices of the detector and descriptor in the source frame. Each map point also stores a list of keyframes, where each keyframe stores the corresponding vector of keypoints and their descriptors computed from all levels of image pyramid. Camera pose associated with each keyframe is represented as a coordinate system T_W^C where C and W are camera and world coordinate systems respectively. Each keyframe also stores the list of indices of the map points that are visible in the keyframe along with their corresponding image positions.

2.2.3 Plane Fitting

Typically, buildings are composed of planes, so we extract planes from the 3D point map created by the SLAM system. A RANSAC [24] based plane fitting method was developed that uses a voting scheme for assigning points in the 3D point cloud to individual planes. For a set of 3D data points $\{P_i(x_i, y_i, z_i); i = 1, \ldots, N\}$, where N is total number of 3D points in the point cloud, the plane equation has been defined as,

$$ax + by + cz + d = 0 \tag{2.1}$$

where, a, b, c are slope parameters and d is the distance of the plane from the origin.

Planes are extracted by randomly constructing different planes from point cloud data. The

minimum number of points needed to determine a plane are three. Three random points are sampled from the point cloud and checked for collinearity. If we define those three points as $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$. If those points are non- coincidental and creates a triangle of area zero that proves those points are collinear. So, the condition of collinearity is,

$$x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) = 0. \quad (2.2)$$

However, it is highly unlikely to be exactly zero. If the value is less than 0.001, it is considered to be collinear from an implementation perspective. If the points selected are collinear or coincident, new points are considered for hypothesis proposal. This process is repeated for N number of times. The number N is selected in a way to ensure probability p of finding at least one set of random samples is without an outlier. p is chosen to be 0.99. Let i is the probability of any selected point out of m number of points is an inlier of the hypothesis and $o = 1 - i$ is the probability of finding an outlier gives,

$$1 - p = (1 - i^m)^N. \quad (2.3)$$

Rearranging the above equation to solve for N gives,

$$N = \frac{\log(1 - p)}{\log(1 - 1 - o^m)} \quad (2.4)$$

The resulting candidate planes are scored against all points in the cloud to validate the candidate plane. The scoring process is based on votes from the point cloud. In a candidate plane, the points that falls onto that plane, votes for the plane. The total vote for a candidate plane is the score of that plane. The quality of a plane is determined by its score. After a predefined number of trials, the candidate plane having the highest score is validated as a plane. Points voting for the valid plane are tagged to that plane and removed from the plane fitting consideration. These points form that plane and not considered to vote for other plane hypotheses. The procedure is then repeated on the rest of the point cloud to find subsequent planes. The planes are finally made robust using

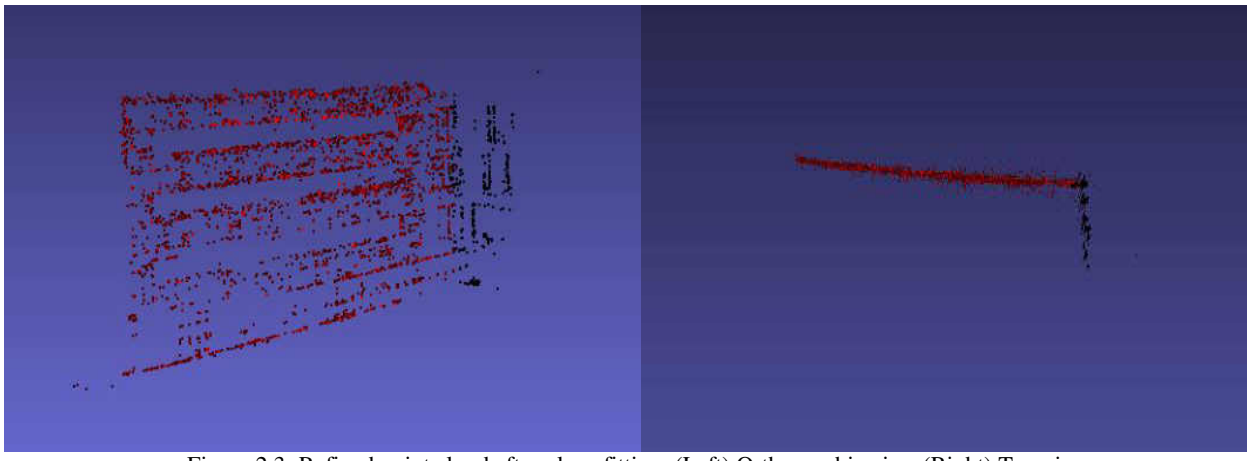

Figure 2.3: Refined point cloud after plane fitting: (Left) Orthographic view (Right) Top view

least-square constrains. After fitting the planes, the point cloud is refined. All the points not included in any plane are considered as outliers and removed from the cloud. Figure 2.3 shows the output point cloud after plane fitting.

2.2.4 Generation of Metric Spatial Database

Spatial databases are used to represent data in space. The space can be a geographic space, a man-made building space or even a user defined coordinate. A spatial data system must at least be able to query for a data from a large collection of objects from a particular area without searching through the whole dataset. A spatial query is A statement or logical expression that selects geographic features based on location or spatial relationship. Spatial indexing is a way to organize the space and the objects in that space so that only part of the space and related objects answer to a spatial query. Generation of a metric spatial database will be explained in this section.

The camera pose while capturing the first keyframe is set to the identity in SLAM coordinate system. The 3D CAD model coordinate system is set while designing it in AutoCAD Revit. As the proposed system takes a sequential photo collection, the user defined entity as stated in 2.2.1

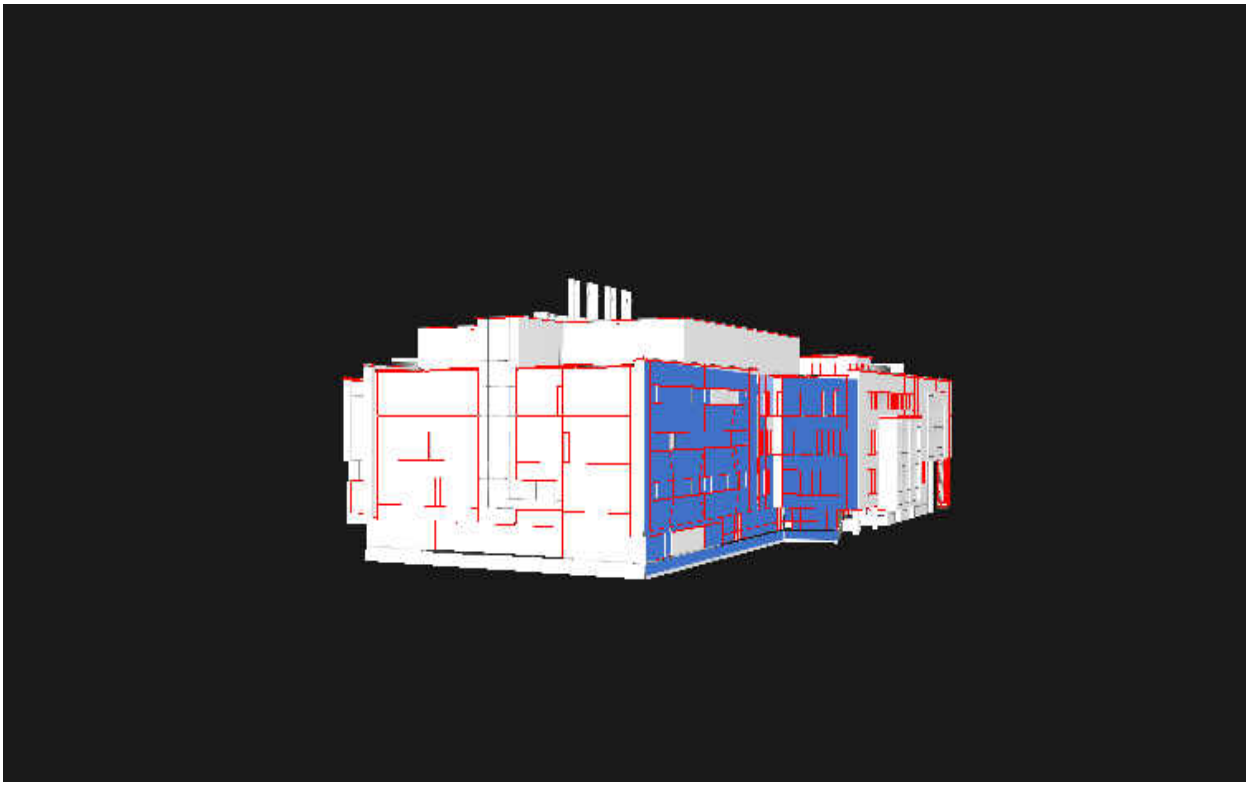

Figure 2.4: Selected region for inspection in CAD model

should appear in the second keyframe. The keyframes are generated by the SLAM system. In this implementation, there were 12,000 images from which 4654 keyframes were generated by the SLAM. The proposed system creates small 11×11 patches around the clicked points of the first keyframe. Then a normalized cross-correlation based matcher finds the corresponding four corners in the second keyframe providing a 2D-2D correspondence between both keyframes. This also creates a 3D-2D correspondence of the 3D CAD model to the second keyframe. These correspondences are used to compute the scale factor and transformation from the world coordinate system to the model coordinate system.

The proposed system uses the user-provided 3D-2D correspondence between the 3D CAD model and the first two keyframes to find the scale factor. The pose of UAS for first two keyframes according to the world coordinate, P_{W1} and P_{W2} are found from the SLAM generated 3D map, as the map stores the pose of the camera during generating the point cloud. Corresponding 3D poses

for the model coordinate system P_{Model1} and P_{Model2} are found by solving the p3p problem as described in section 2.2.1. The location of the four user-defined 2D points in the image plane of the first keyframe $p_{11}, p_{12}, p_{13}, p_{14}$ and the location of the four corresponding 2D points in second keyframe $p_{21}, p_{22}, p_{23}$and p_{24} are projected to the camera plane. Assuming a calibrated linear camera, the intrinsic parameter matrix, K is known. The principal point, C in pixels and the focal length, f in pixels, are used to find the projected point,

$$p_{proj} = {}^{1}/_{f} \times (p_{ij} - C), \tag{2.5}$$

where, p_{ij} is the point being mapped on the camera plane. The projected point is in the homogeneous coordinate system and expressed as, $p_{proj} = [x\ y\ 1]^T$.The transformation matrix between pose of second keyframe to first keyframe is,

$$T_2^1 = T_W^1 (T_W^2)^{-1}, \tag{2.6}$$

where T_W^1 and T_W^2 denotes first and second camera poses in world coordinate system. A simple linear triangulation method is applied to find 3D location, P_{ij}, where i is either World or Model and j =1, 2, 3, 4. If M and M' are homogeneous camera projection matrices for first and second keyframes respectively then, $p_{proj1} = M \times P_{ij}$, where p_{proj1} homogeneous image point is seen from the first image and P_{ij} is 3D world point in homogenous coordinate system. Similarly, the corresponding image point in second keyframe is denoted as $p_{proj2} = M' \times P_{ij.}$ In practice this is found by evaluating the last column of T_2^1. These equations are combined into a form $A \times P_{ij} = 0$, this equation is linear in P_{ij}. After eliminating the homogeneous scale factor by a cross product $p_{ij} \times P_{ij} = 0$ the equation of form $A \times P_{ij} = 0$ can be composed where,

$$A = \begin{bmatrix} xM^{3T} - M^{1T} \\ yM^{3T} - M^{2T} \\ x'M'^{1T} - M'^{1T} \\ y'M'^{3T} - M'^{2T} \end{bmatrix}. \tag{2.7}$$

Solving the previous equation gives 3D coordinate of the image point [25]. In this manner, all four 3D points corresponding to the four corners of the entity from the images are generated as P_{ij}. For the world coordinate the 3D points are $P_{W1}, P_{W2}, P_{W3},$ and P_{W4} and for the model coordinate system the generated points are $P_{M1}, P_{M2}, P_{M3},$ and P_{M4}. Then the average distance of the 3D points is calculated as d_{wolrd} and d_{model} for world and model coordinate system respectively. Finally, the scale factor,

$$s = \frac{d_{model}}{d_{wolrd}}, \tag{2.8}$$

is calculated.

The pose of first keyframe in world and model coordinate are, T_W^1 and, T_M^1 which are used to find the transformation between world to model coordinate,

$$T_W^M = (T_W^1)^{-1} T_1^M. \tag{2.9}$$

If A and B are two matrices, they are similar if $B = P^{-1}AP,$ where P is the similarity transform. In this case the similarity transform is,

$$T = \begin{bmatrix} R & \vec{t} \\ 0\ 0\ 0 & s \end{bmatrix}, \tag{2.10}$$

is applied to the point cloud as,

$$P_M = T \times P_W, \tag{2.11}$$

where P_M and P_W are points on model and world coordinates respectively, to align the point cloud to the model coordinate system. With the completion of the registration process, a spatial databse has been generated.

2.2.5 Spatial Query for Images

As the source keyframes are tagged along with each point in the cloud, a spatial index of images is used for efficient spatial image query. The entries in the spatial index depend on the vertices location in the model coordinate system.

The user must click on the area or entity on the visualization window to run the spatial query for image of that particular area. A ray casting method is used to select the 3D location of the mouse click in the model. For ray casting, a 3D ray is projected from the mouse into the CAD model space through the OpenGL visualization window. If the ray intersects with any object, the 3D location of the intersection in CAD coordinate system is calculated. The calculations will be done in CAD coordinate which is the global coordinate in this case. So, the origin of the ray is the 3D location of the OpenGL camera expressed as O. Our ray casting will be done on world coordinate, so in this case ray origin O is in the x, y, z position in the world coordinate system. If the direction normal of the ray is expressed as $\widehat{D}$ then any point on the ray can be expressed as

$$R(t) = O + \widehat{D} * t, \tag{2.12}$$

where t is the distance of the point from origin O. If there are multiple planes the nearest plane to the camera is considered.

For selecting a point of the 3D CAD model the user has to simply click on the model with mouse

pointer. The pixel location has x_mouse and y_mouse coordinates and the range of x_mouse is from 0 to width of the viewport and y_mouse has a range of height to 0. Then the x_mouse, y_mouse coordinate is transformed into normalized device coordinate. In normalized device coordinate the range of $x, y\ and\ z$ will be from -1 to 1 and the center of the coordinate system will be the center of the viewport. In normalized device coordinat e the new $x, y\ and\ z$ will be as follows,

$$x = \frac{2 * x_{mouse}}{width - 1} \tag{2.13}$$

$$y = 1 - \frac{2 * y_{mouse}}{height} \tag{2.14}$$

$$z = -1. \tag{2.15}$$

In OpenGL coordinate system y_mouse was downwards and the direction is switched upwards in normalized device coordinate. z is assumed to be -1 because the ray is supposed to point forwards which is negative in OpenGL coordinate system. Now, the x, y, z is in Euclidian plane. To convert it to projective plane the coordinate system must be in homogeneous coordinate system. To do that we added a w to get a 4D vector. To get the ray in camera coordinate system we multiply the inverse of projective matrix with the ray in homogeneous coordinate. To express the ray in world coordinate we multiply the ray in camera coordinate with the inverse of the view matrix and finally normalize the ray vector.

The CAD model is mostly planar. We parameterize the CAD model as various planes and the intersection between the plane and ray would provide the intersecting 3D location in the CAD coordinate.

The selected 3D point of the model is snapped to the nearest vertex, so that the user does not have to click precisely. The nearest point of the point cloud to the selected vertex is selected as the

entry to spatial index for query. As the SLAM stores all the source keyframes of each of the points in the point cloud, the associated source keyframe to entry point is returned.

2.2.6 Metric Data Analysis

The proposed system assumes the user wants distance from one vertex to another to check the dimensions of different entities. By using the spatial query for an image, the user finds the image of the entity to be inspected.

First the image is converted into a grayscale image. Then a canny edge detection is done to find edges on the image. Canny uses a Gaussian blur at the beginning of the algorithm, which seems to eliminate noise and aids in the correct classification of most images. In this implementation, a 5×5 Gaussian filter was used to with σ = 1.4. The filter was applied on image A to find filtered image,

$$B = \frac{1}{159}\begin{vmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{vmatrix} * A \tag{3.16}$$

where, (*) denotes convolution operation.

Then it determines the intensity gradient for the image by applying a pair of convolutional masks in both the x and y directions in the form

$$\mathrm{Gx} = [-1\ 0\ 1\ -2\ 0\ 2\ -1\ 0\ 1] \tag{3.17}$$

$$\text{and } \mathrm{Gy} = [-1\ -2\ -1\ 0\ 0\ 0\ 1\ 2\ 1]. \tag{3.18}$$

Then the gradient strength,

$$\mathrm{G} = \sqrt{{G_x}^2 + {G_y}^2}, \tag{3.19}$$

alongside the gradient direction,

$$\theta = \text{atan2}(G_y, G_x) \tag{3.20}$$

is calculated and rounded to either 0, 45, 90, or 135 degrees. Non-maximum suppression is applied to remove all pixels not considered as the edge leaving only thin candidate line edges. The final step involves hysteresis that uses two upper and lower thresholds. If a pixel gradient is higher than the upper threshold it is accepted as an edge, below the lower threshold it is rejected as an edge, and between the two thresholds it is accepted as an edge if it is connected to a pixel above the upper threshold but removed otherwise. Here, the thresholds were set as 100 for the lower and 200 for the higher threshold. With the edge pixels found, any pixel above a zero-intensity value were set to the maximum 255 to provide a binary image with the background pure black and all edges to be considered pure white. Each white pixel was then stored in a vector of points to be considered

in the line detection algorithm [27]. Figure 2.5 shows one output of canny edge detection.

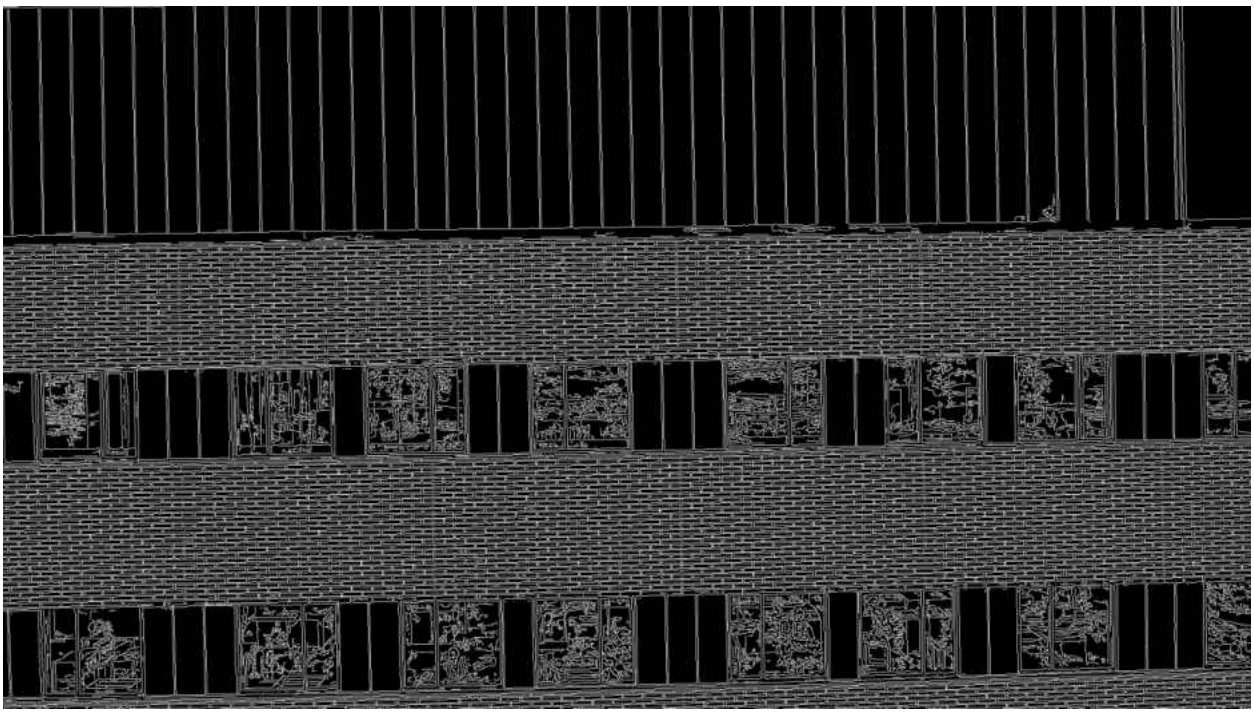

Figure 2.5: Canny edge detection

Contours are curves joining continuous points having same color or intensity. Contours are found by using OpenCV's implementation of [27] on the binary image. This method stores all the contour points. If there are two subsequent points (x_1, y_1) and (x_2, y_2) of the contour. Those points will either be vertical, horizontal or diagonal neighbors. Figure 2.6 show detected contours on Figure 2.5 overlaid on the source image.

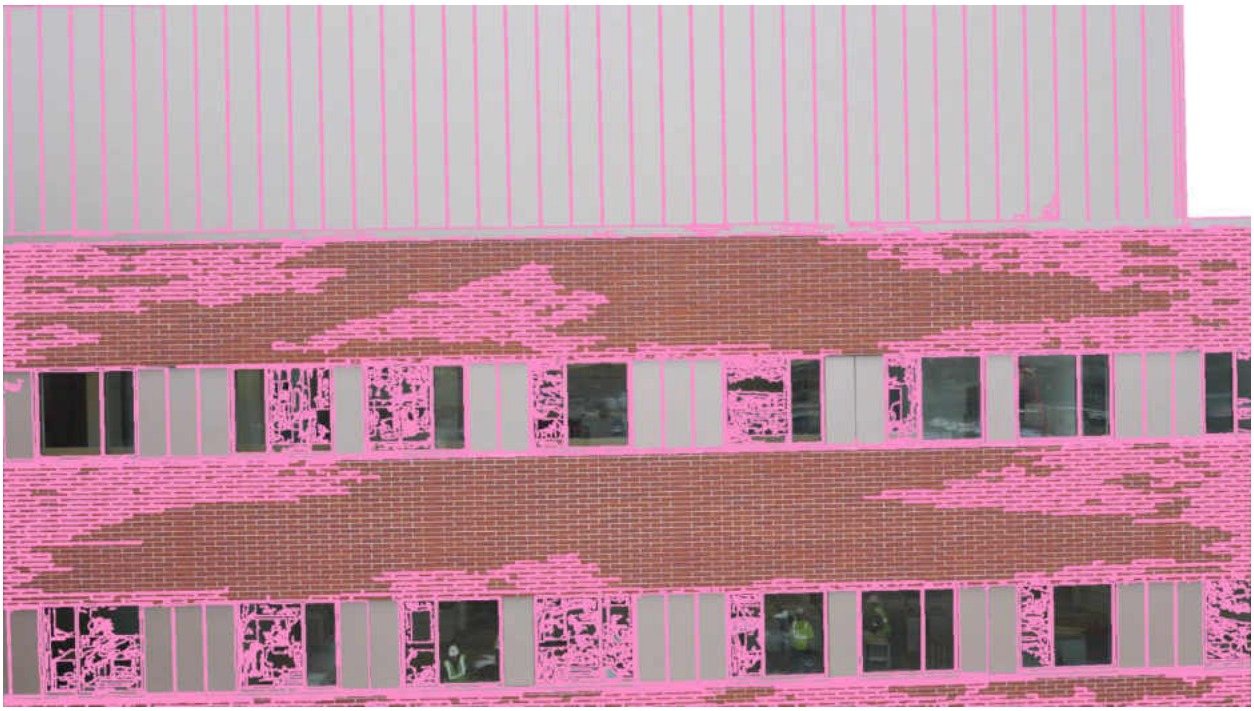

Figure 2.6: Contour detection

After finding all the contours, OpenCV's implementation of Doughlas-Peucker [28] algorithm has been used to find rectangular shapes from the contours found. This algorithm approximates polygonal curves with the specified vertices. In this implementation, input curves were the detected contours found by using the findContours() function. The epsilon value was selected to be the length of the contour which was found by arcLength() function of OpenCV. The boolean value was to true to detect closed contours only to determine the rectangles. The approxCurve was returned which has a member function size() which returns how many corners are there is the returned polygon. If it is equal to 4 it was detected as a rectangle. Figure 2.7 shows detected rectangles on figure 2.6. It detects even very small rectangles which do not represent any meaningful features. To get better output,

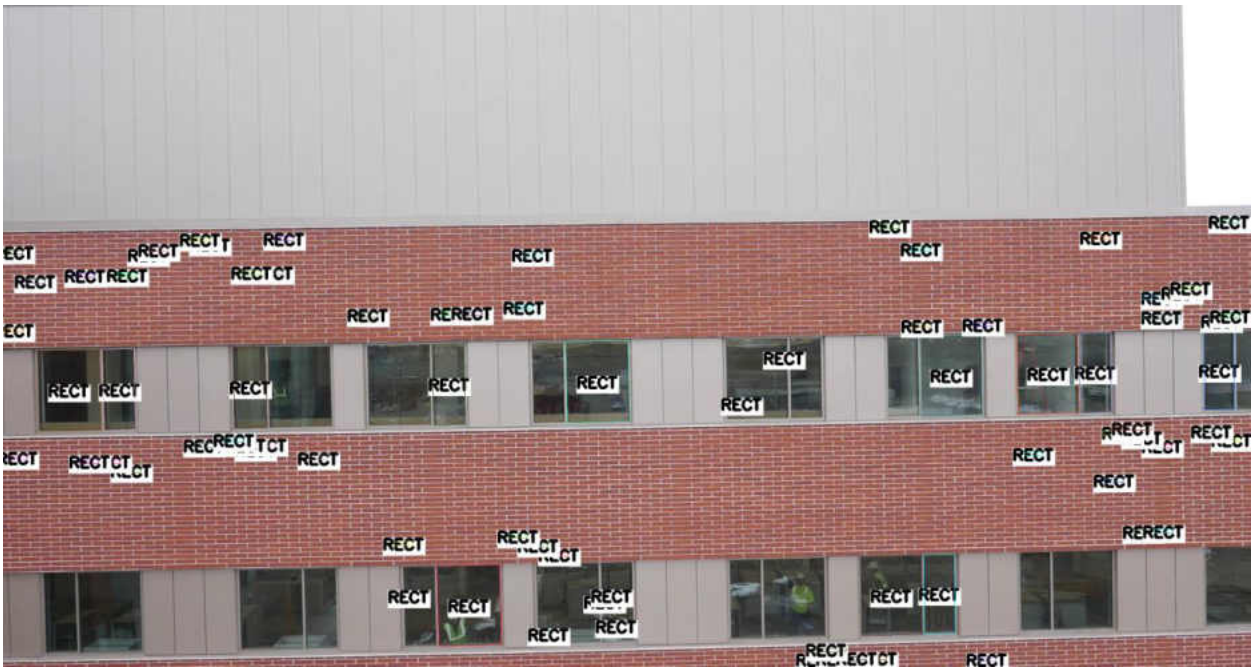

Figure 2.7: Detected rectangles using Doughlas-Peucker algorithm

the rectangular contour area is calculated by using OpenCV's contourArea() function to make sure that the area is big enough to be considered as a window. Figure 2.8 shows windows detected in a queried image by this method. The actual height of the window is taken during

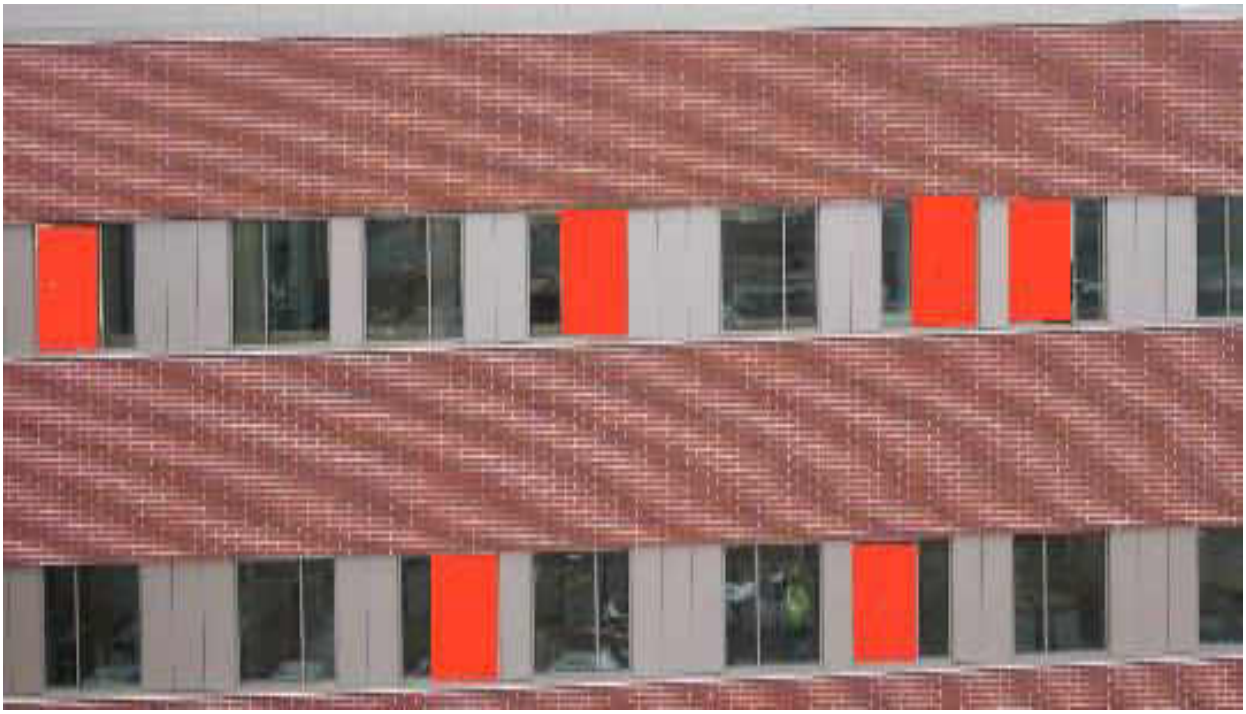

Figure 2.8: Detected windows in an image

Figure 2.9: The magnifier for accurate clicking

alignment of the spatial database to the 3D CAD model. The pixel distance of the height of the window then sets a scale factor. The user clicks on two points of the image to find the 3D distance and the nearest windows scale factor is used to find the 3D distance. A magnifier has been implemented to aid the accuracy of clicking on the image. To implement the magnifier, OpenCV's pyrUp() function has been used. Figure 2.9 shows the magnifier.

2.2.7 Textured Model Generation

All the keyframes for each plane are then stored in a vector to be stitched together. OpenCV's image stitching is very slow and due to large image size in this dataset the default OpenCV's stitcher could not stitch more than 6 images. However, OpenCV has a stitching_detailed.cpp implemented which was modified as [29] which uses OpenMP to speed up the stitching process. That also had some limitations. It was not able to stitch more than 22 images. Finally, OpenPano

was used which is also used by Autostitch. OpenPano is an open-source C++ implementation of [30] which successfully stitches all the images supplied. To get high quality images, it requires images to be rectified before giving them as input. Before running the SLAM all the images are rectified after being loaded, so that problem does not occur in our case. Also, the exposure parameters are constant for better blending. Multiple parameters have been used to get better output from OpenPano. First, stitching mode has been set to camera estimation mode which gives better output as it performs pairwise matching. As the images are collected from a video, we assume the input images are ordered. To avoid any irregularities from the stitching process, the output image was cropped. In OpenPano, to stitch images, SIFT features were extracted from the images to be stitched. Then using a k-dimentional tree 10 nearest neighbors for each feature were found. Images that have the most feature matches to each image are selected. A pairwise matching was done among those images using RANSAC to solve for homography. Then a probabilistic model was used to verify these matches. Connected components in image matches are then found. A bundle adjustment to solve for the rotation and focal length of the camera is performed for each connected component. The final panorama is rendered using multi-band blending. Because of a high number of key-points our system takes in every tenth image from the keyframe vector and stitches them together. During the image registration to the 3D mesh model the user specifies the required area to be inspected. That process takes in the boundaries for each plane in the specified region. That boundary is used to texture map the stitched image to its corresponding plane. A photorealistic 3D texture model is created in this manner.

CHAPTER III

DATA COLLECTION PROCESS

Figure 3.1 provides images of data being collected with a UAS. The UAS used in this application was the DJI Spreading Wings S1000+ model. One of the most important features of the S1000+ is a low gimbal mounting bracket which enables a wide range of possible shooting angles and camera motions [30].

It was fully compatible with the Zenmuse Z15 camera gimbals from DJI which stabilizes the camera to the desired orientation during flight. A Canon EOS 5D Mark III camera has been used to capture video of the scene. The camera has a 22.3 Megapixel CMOS n sensor. High-Definition video with a resolution of 1920×1040 was collected at 30 frames per second.

Figure 3.1: Random Images during data collection

The SLAM requires a calibrated camera system. Calibration images were taken before every flight. Figure 3.2 shows the calibration image collection at the project site. The 3D CAD model used was loaded in the system using Open Asset Import Library (Assimp) and OpenGL.

Camera calibration is a process to estimate the parameters of a lens and image sensor of a camera. These parameters can be used to correct distortions in an image, to measure the size of an object in world or to locate the camera in the world.

Figure 3.2: Calibration Image collection

Camera parameters include,

i) Intrinsic camera parameters

ii) Extrinsic camera parameters

iii) Distortion Coefficients.

Estimation of camera parameters requires 3D-2D correspondence of an object. Generally, a camera calibration pattern (Checker Board Pattern) is used to find those correspondences. The camera model that has been used in this work is based on Pinhole Camera model. The camera parameters are expressed by a 4×3 matrix known as a Camera Matrix. The camera matrix maps

the 3D world scene in a 2D image plane. The extrinsic parameters represent camera poses during each image capture in 3D world.

Calibration of Canon EOS 5D Mark III camera was done using OpenCV's camera calibration code that is provided within OpenCV's source library [33]. This camera calibration code provides several algorithms to compute the intrinsic and distortion properties found in pinhole cameras. Distortion is typically caused by symmetry found in the lens used with the camera [33]. OpenCV takes into account both radial and tangential distortion that can be found in cameras. Radial distortions are classified as barrel, pincushion, or a combination of the two which cause straight lines to no longer appear straight in distorted images. In barrel distortion, lines appear to bend away from the center of the image in a "fish-eye" effect whereas pincushion distortion causes lines to bend towards the camera. The following formulas provides the equations used in OpenCV for correction of the radial distortion [33]

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (1)$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (2)$$

This states that for a pixel at location (x, y) coordinates in the input image, the corrected output image without distortion will be the new $(x_{corrected}, y_{corrected})$ coordinates [33].

Tangential distortion is caused by the camera lenses not being perfectly parallel to the imaging plane. This is corrected as

$$x_{corrected} = x + 2\big(\, p_1\, xy + \, p_2(r^2 + 2x^2)\big) \quad (3)$$

$$y_{corrected} = y + 2(\, p_1\, (r^2 + 2y^2) + \, p_2 xy) \quad (4)$$

The five parameters listed are provided by OpenCV once the camera calibration is finished and presented as a one row matrix with 5 columns in an .xml file as

$$Distortion_{coefficients} = (k_1 k_2 p_1 p_1 k_3) \quad (5)$$

Intrinsic parameters include parameters inherent to the camera itself. These include the focal length of the camera and the image offset. The focal length is the distance from the center of a lens and its focus and represents how strongly the system converges the light onto the optical sensor. Image offset which represents the optical center of the camera in pixel coordinates. OpenCV creates a camera matrix of these unknown parameters as the 3-by-3 matrix below

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_y \\ 0 & f_y & c_x \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{6}$$

Where f_x and f_y represent the focal lengths in the x and y directions, c_y and c_x represent the image offset, and w is simply used to homography the coordinate system. Solving all the unknown parameters mentioned previously is accomplished through the calibration.

OpenCV solves these unknowns through a series of several geometric equations using a known calibration object or pattern [33]. The patterns supported by OpenCV's algorithm include chessboards, circular grids, and asymmetrical grid patterns. OpenCV first locates the corners or circles of the pattern and determines where that plane lies in 3D space in relation to the camera. Knowing the 3D location of the object provides 6 parameters for rotation and translation.

OpenCV solves the distortion and camera parameters separately, focusing first on the distortion parameters [33]. Due to distortion parameters being tied to the 2D geometry of how the image is warped, a known calibration pattern can provide the information necessary to solve these parameters. Technically a simple three corner pattern providing 6 pieces of information would provide enough information to solve the distortion parameters but more are needed for robustness.

Intrinsic parameters are also solved through use of the calibration pattern. For each view of the

calibration pattern 6 unknown values for the extrinsic parameters are developed due to the rotation and translation of the object along with the previously mention 4 intrinsic parameters. To solve for these 10 unknowns, a pattern of N corners or circles can be used to provide 2NK constraints, where K is the number of images, due to each point having an x and y position. Solving this series of equations then requires that $2NK \geq 6K+4$. This does require multiple images though due to one image only providing 4 corners worth of information since that is all that is needed to describe a plane. So, a pattern with enough point and multiple images can be used to solve the intrinsic parameters [33].

CHAPTER IV

EXPERIMENTAL RESULTS AND DISCUSSION

In this section, the proposed algorithm is compared against the latest version of the state of the art Pix4D mapper (version 3.1) [31]. The primary focus of this research is to incorporate a BIM to access the point cloud which provides us a prior knowledge on how the building was intended to be constructed. The CAD model provides entry to the spatial database eliminating the need to perform a dense reconstruction as Pix4D. Despite not having a dense reconstruction the proposed algorithm outperformed Pix4D in metric data calculation and textured 3D model generation.

Spatial Query:

The point cloud generated by the SLAM have been aligned to the 3D CAD model by a similarity transform. Each point has a keyframe associated with it, which creates the spatial index of images. The user clicks on the desired entity. The nearest point from the point cloud to the vertex is selected, working as an entry to the spatial index for query and efficiently searches for the spatial image data. Figure 4.1 shows some of the outputs using the spatial query.

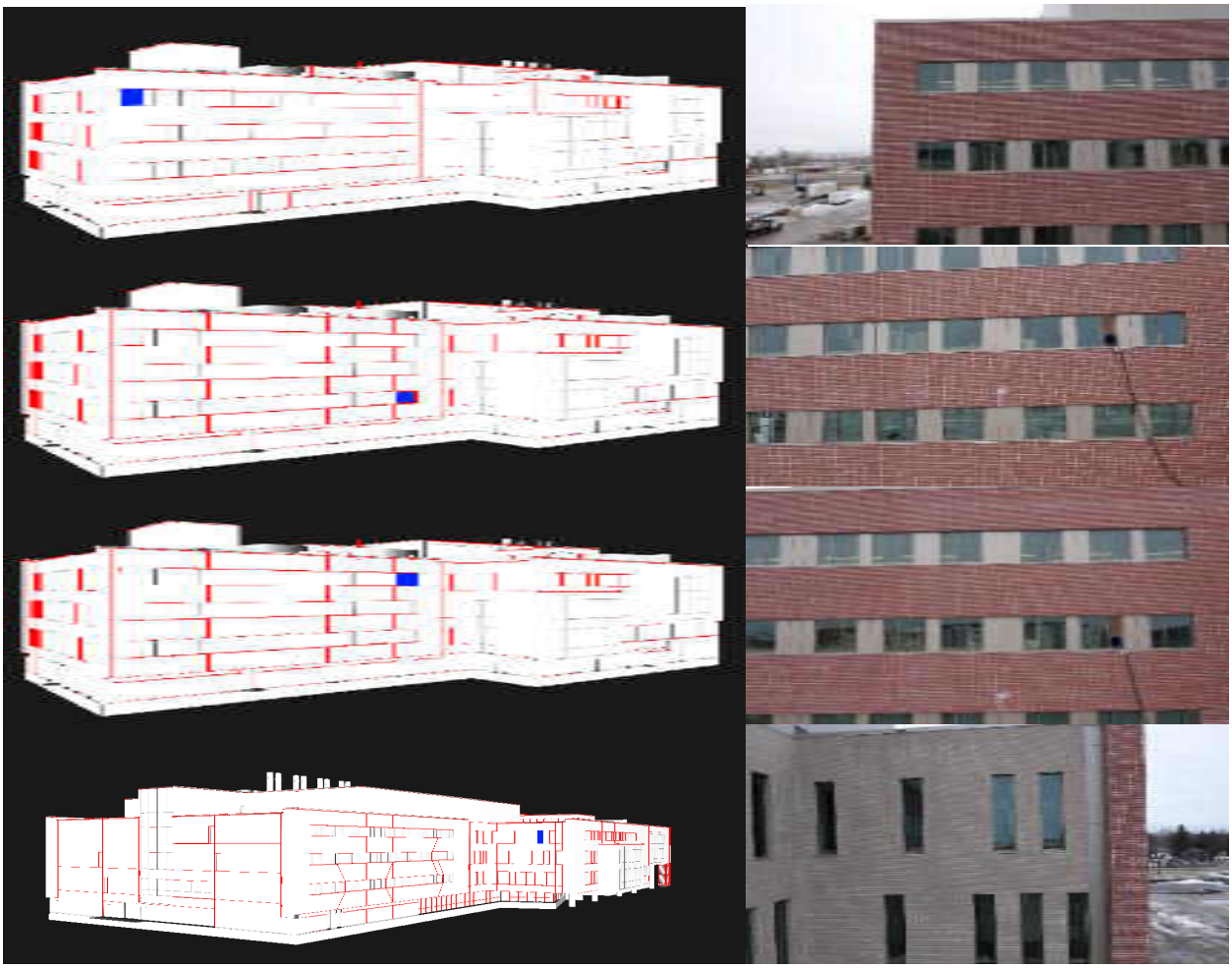

Figure 4.1. Spatial query: The window to be inspected in the model is marked with blue square and the image of that region found through the spatial query.

After a query the user may choose to check dimensions of various entities which is the metric data analysis in our system.

Metric Data Analysis:

The proposed algorithm's dimensional calculations have been compared to Pix4D's calculations. The proposed system showed significant improvements over Pix4D in metric data extraction. However, Pix4D requires geotagged images to assign scale and orientation. It is not always possible to have geotag information with images, so this algorithm was developed keeping this in mind and geotags was not included. As the proposed system does not require geotagged images, the scale was provided manually to Pix4D. To apply a scale constraint into Pix4D, the recommended process is to click on both vertices from the dense point cloud that Pix4D generates and provide the accurate metric distance. Moreover, it is recommended by Pix4D to correct the vertices in at least two of the corresponding images. The proposed system applies scale by using two consecutive images.

As a direct comparison, a scale was applied to the Pix4D model using the dense point cloud and two images. Pix4D does recommend more scale constraints as it will provide better accuracy. This is true for both Pix4D, as well as the proposed algorithm. Figure 4.2 below shows the process to apply scale using Pix4D mapper.

Figure 4.2. Applying scale constraint on Pix4D mapper.

After applying scale constraints, the height and width of 15 windows providing 60 total distances for each method have been tested. For verification, these same distances were manually measured. The actual width and height were 2.012 meters(m) (6.6 feet) and 1.829 meters(m) (6 feet). The CAD model provides these distances as 1.829 meters for both width and height. The scale was applied based on the height of the windows, as provided by the CAD model, on both the proposed model and Pix4D. The proposed system and Pix4D both determined the building was not constructed per the CAD model dimensions.

The proposed model resulted in a mean squared error (MSE) of 31.9 cm^2 whereas Pix4D mapper's MSE was of 45.6 cm^2. For Pix4D's width calculation, it has a standard deviation of 4.92 cm, as opposed to the proposed system's standard deviation of 4.28 cm. For height calculation, Pix4D's standard deviation resulted in 4.17 cm, where the proposed algorithm provided a standard deviation of 3.27 cm. Pix4D had a combined standard deviation of 6.45 cm where the proposed

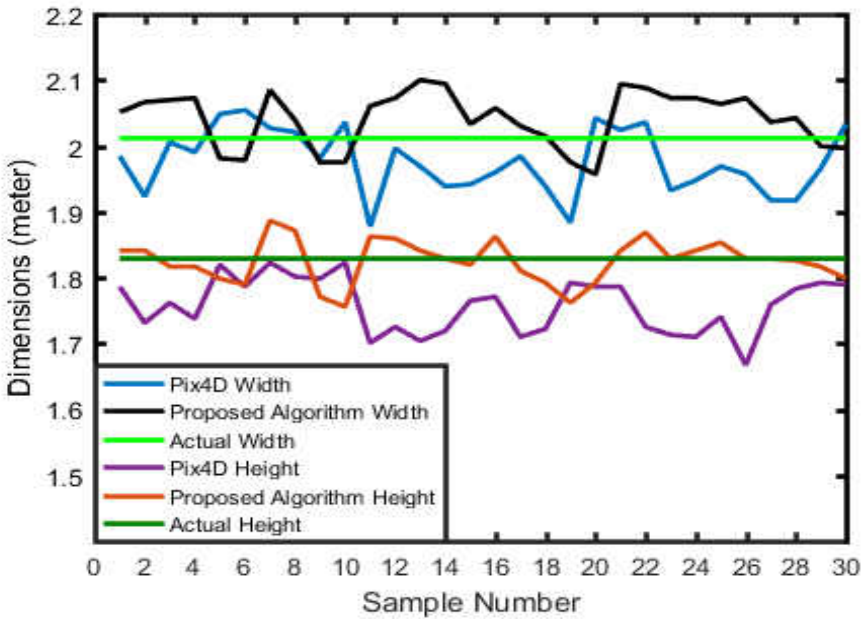

Figure 4.3. Dimensions calculated with Pix4D and Proposed Algorithm along with their actual values.

algorithms combined standard deviation was 5.39 cm. Figure 4.3 provides the width and height calculations along with the actual.

A t-test was performed on the width and height averages to verify the statistical significance of the calculated errors. The t-test is a statistical hypothesis test where test statistics follow a student's t-distribution if the null hypothesis is true. In a one sample t-test, a value is specified as null hypothesis and the mean is tested to verify if it matches with the null hypothesis. In case of a two-sample t-test, the null hypothesis assumes the means of two populations are the same. The first step in a statistical hypothesis test is deciding on the significance level known as $\propto$. It is the probability of rejecting the null hypothesis even when it is true. The p-value is calculated and compared to $\propto$ to accept or reject the null hypothesis. The p-value is the probability of obtaining a result which is equal or better than actually observed result under the null hypothesis. If X is a continuous random variable where x is an observed instance, p-value is probability of $X > x$ (right

tail event) or $X < x$ (left tail event) or both (two tail event) under the null hypothesis. The null hypothesis is rejected if p-value is less than a pre-defined $\propto$.

Tables 4.1 and 4.2 show the resulting t-tests from the two-sample width and height data assuming unequal variances. In both cases, an $\propto$ of 0.01 was used. This shows we are 99% confident that the proposed method is statistically better than Pix4D. The null hypotheses for both cases were set as there are no significant difference.

Table 4.1. t-test results on two sample widths assuming unequal variances

	Pix4D Width (meters)	Proposed Algorithm Width (meters)
Mean	1.97673	2.04084
t Stat	-5.38338	
t Critical 2 tail	2.66487	
P two-tail	1.4E-06	

Table 4.2. t-test results on two sample heights assuming unequal variances

	Pix4D Height (meters)	Proposed Algorithm Height (meters)
Mean	1.75758	1.82494
t Stat	-6.95956	
t Critical 2 tail	2.66822	
P two-tail	4.4E-09	

Due to the p_{value} for both resulted in a much lower value than the $\propto$ value, the null hypothesis can be rejected. We can therefore state that the mean average in the proposed algorithm were significantly closer to the actual values than Pix4D.

An analysis of variance (ANOVA) was performed on the variances of the data analyses to verify the statistical significance of the sample variables. Table 4.3 provides the ANOVA output.

Table 4.3. Two-factor ANOVA with replication results on both width and height with $\propto = 0.01$

Source of Variation	F	P-value	F Critical
Sample	73.39466	5.25E-14	6.858521
Width/Height	803.691	5.6E-54	6.858521
Interaction	0.044844	0.832588	6.858521

Pix4D and the proposed algorithm are represented as the two samples. The p_{value} $(sample)$ was lower than $\propto$. The $F_{Crirtical}$ $(sample)$ is lower than $F(sample)$ showing both samples have significant differences. The second row in Table 4.3 represents the effect of both samples on width and height calculations and the $p_{value}(width/height)$ is much less than $\propto$. Therefore, we can reject the null hypothesis and conclude both systems are significantly different on the width and height calculations. The p_{value} $(interaction)$ calculated a value greater than $\propto$, showing we can conclude the interaction between width and height have no significant difference, meaning the effect of width or height does not depend on one another.

We can therefore assume the proposed algorithm provides significantly more accurate results than Pix4D data analysis.

3D Textured Model Generation:

Another tool developed was for visualization through 3D textured model. To visualize the 3D texture model, the OpenGL library was used with a moving camera implementation. This allows the user to roam through the 3D textured model to look for any visual anomalies. We have compared the proposed system to Pix4D mapper's textured reconstruction. For a fair comparison,

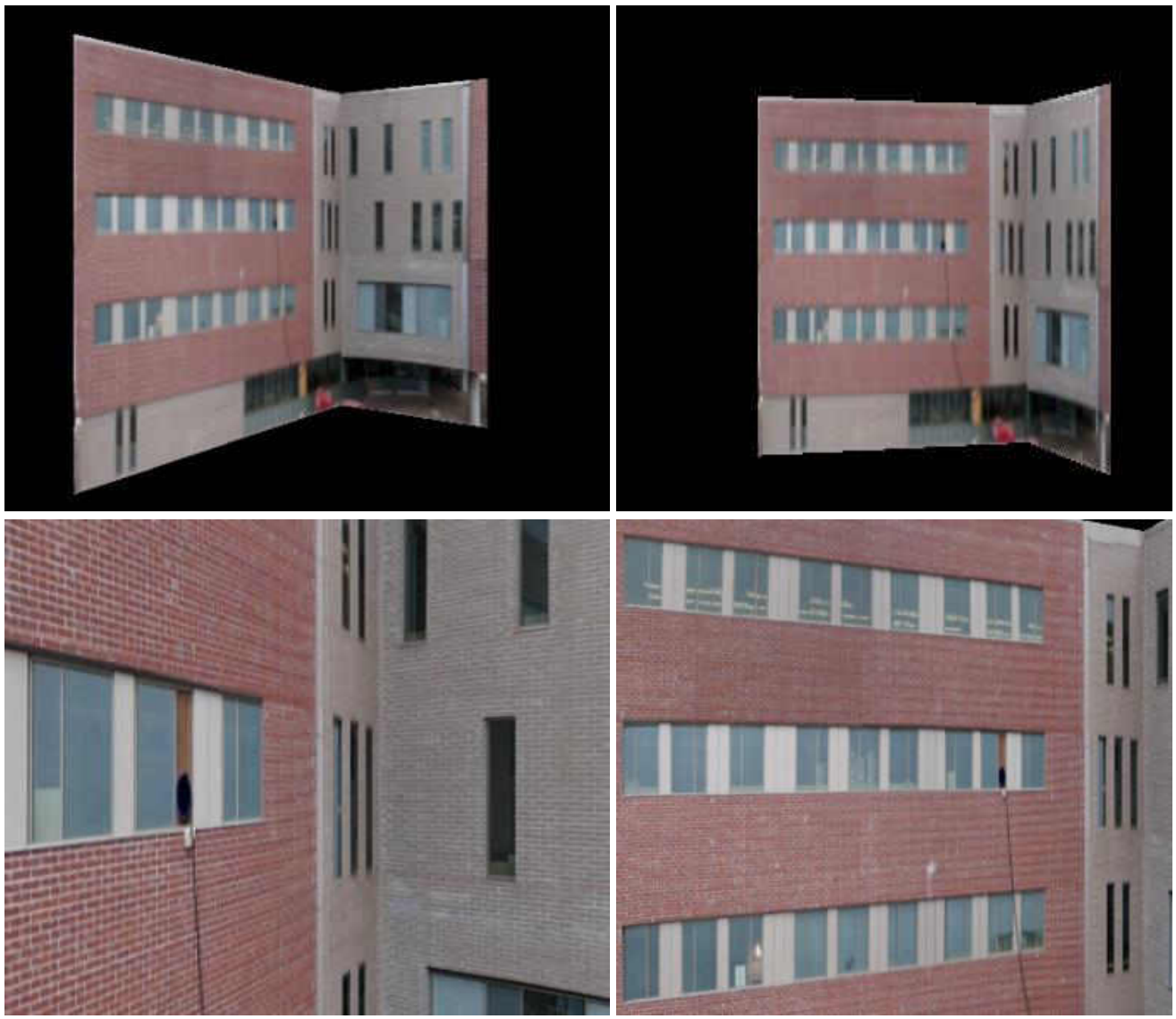

Figure 4.4: 3D texture model with mipmapping from various camera positions

we have selected the high-resolution texture mapping option for Pix4D. Figure 4.4 shows some of the images taken from the proposed systems 3D textured modelling.

(a) (b)

Figure 4.5. Texture Mapping (a) with linear filtering (b) with mipmap filtering

Figure 4.5(a) shows the texture map using linear filtering. Thus, it has visible artifacts due to aliasing. A mipmap filter has been used to remove the artifacts (Figure 4.5(b)).

Figure 4.6. Textured model comparison: (Left): Pix4D's textured model (Right): Proposed Algorithm's textured model

Figure 4.6 shows the differences between the proposed system and Pix4D's 3D texture models.

Pix4D's textured reconstruction results in clearly visible holes and artifacts. Windows are not realistic and provide visible anomalies. Straight lines are distorted resulting in wavy lines, unusable for detailed visual inspections. The proposed method does not have any of these artifacts resulting in a photorealistic rendition.

Moreover, the visualization and movement of the camera is recorded in a video and so that it can be played back in a virtual reality mode to be visualized using a VR headset. It makes the user experience more immersive.

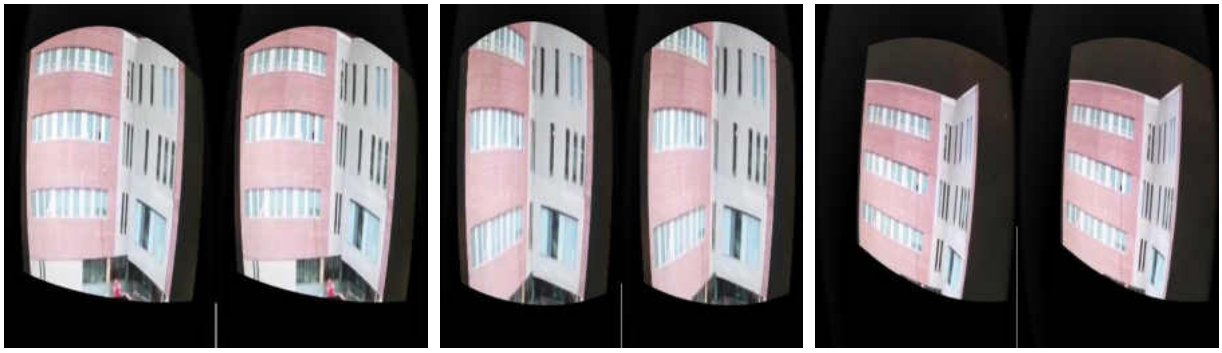

Figure 4.7. Visualization through VR headset

Figure 4.7 shows the parallax going on during visualization

Limitations:

The SLAM systems use loop closure for better pose estimation. However, as our dataset comprises a very high amount of repetitive brick patterns the loop closure had to be turned off. Figure 4.8 shows some images where brick patterns are too repetitive that loop closure provides wrong output.

Figure 4.8: Repetitive brick pattern throughout the dataset.

Another major problem was very large reflective surfaces various sections of the building. The reflections are not static as SLAM assumes and that's why SLAM fails to generate a reliable point cloud at that region.

Figure 4.9 shows some of the very large specular surfaces where reliable feature finding was too difficult and SLAM failed to continue. Taking the video from a greater distance could be a possible solution as it would capture more area where reliable static features could have been found. An attempt was made but was not possible because of the limitation of the flying region.

Figure 4.9: Large Lambertian surfaces in the building under inspection

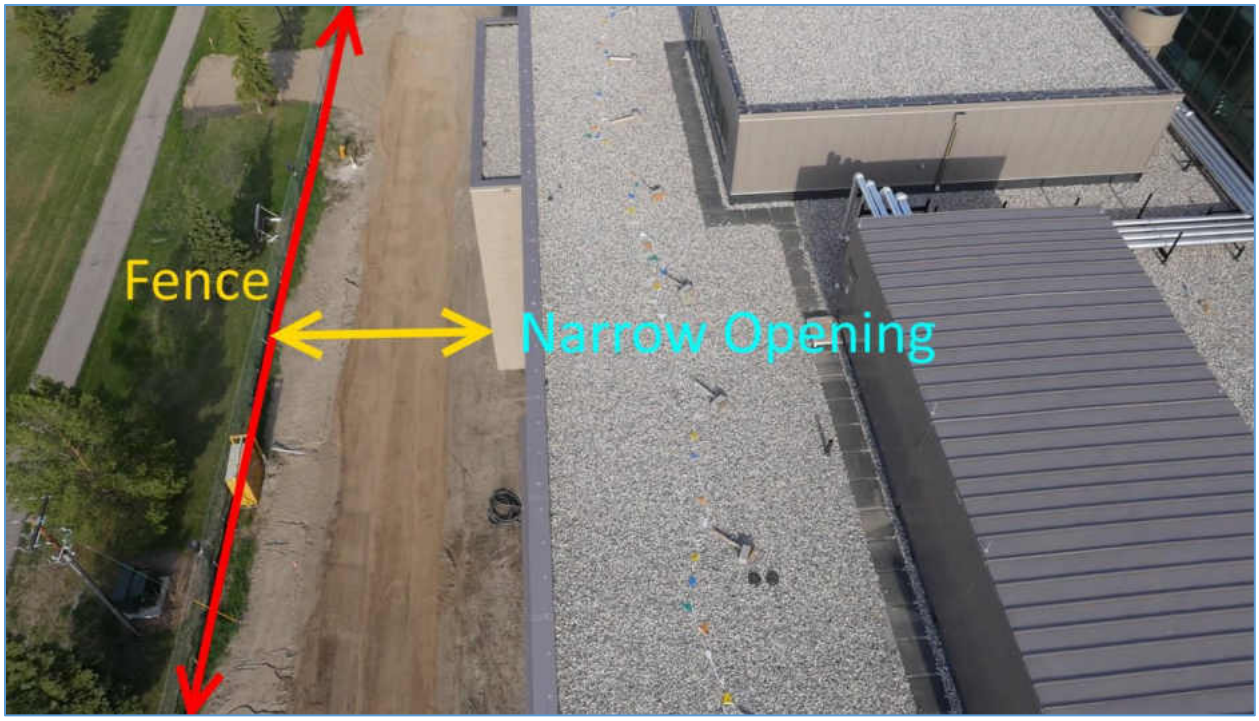

Figure 4.10: Narrow road from building to the fence

Figure 4.10 shows the reason of failure in the attempt to go further away from the building to take images. We were not allowed to fly beyond the fence due to FAA regulations and the problematic images were taken from the furthest position possible.

CHAPTER V

CONCLUSION AND FUTURE WORK

Conclusion:

This system has successfully demonstrated spatial query into the spatial database through the provided 3D CAD model. The spatial query for images will significantly improve the inspection process because of easy and fast access to image data. The images have been used to extract metric data. The proposed system has a mean squared error of 31.9 cm^2 and a standard deviation of 4.28 cm for width calculation and 4.17 cm for height calculation. The metric data extraction method has outperformed the state of the art Pix4D mapper. Statistical tests have been performed to verify that the above outputs are statistically significant. Two t-tests followed by an analysis of variance (ANOVA) was done on the data collected by our system and Pix4D mapper. Although traditional t-tests or ANOVA uses an α value of 0.05 to ensure 95% confidence, α value of 0.01 has been to be 99% confident statistically that this system has generated better results in terms of metric data calculation. The proposed algorithm is the only one using a spatial query for images to extract metric data. 3D CAD models have been used, which is a unique system as other systems uses different sources to work as the prior such as, GPS data for finding pose of the UAV or geotagged images. Introducing 3D CAD model to provide the prior information about dimensions is definitely a valuable addition to the inspection process because in all modern construction the CAD model is almost always available and it can be used without the aid of extra sensors. However, other sensors can be used and the data extracted by using the CAD model can be used to refit the output to get a better result. Using a monocular camera system also played a crucial part as the system was not bound by a fixed baseline of a stereo camera and could operate with a larger depth

range. Metric data analysis along with visualization through a 3D textured model have been performed. A comparison with the state of the art Pix4D mapper was given where the proposed system has been proven to significantly improve the metric data analysis and provided better and photorealistic 3D textured model. The 3D textured model was generated to provide an overall view of the structure under inspection. An OpenGL camera has been implemented to enable the user to move around the structure and visualize the as-built model. The viewport is recorded in a playable video format and can be visualized using a VR headset. The textured reconstruction is photorealistic and compared to Pix4D produces a lot less anomalies. The proposed system is easy to use and does not require the user to have any previous knowledge of visualization, rendering or CAD software.

Future works:

Detecting all the windows would make the metric data calculation better and robust. Deep convolutional neural networks could be possible solution to detect all the windows in each frame. Another interesting step towards further research could be automatic scheduling monitoring and temporal navigation using the 3D CAD model. Creating a temporal database would allow the user visit through time and access as-build documentation. Combining spatial and temporal database will enable users to inspect the construction using the CAD model. The user will be able to click on the CAD model and the spatial query will return all the images with associated timestamp of that interest area. Real time implementation of the algorithm could be another future work to make inspection real time. Optimizing the system to implement on an on-board environment or mobile devices will let users inspect the building using their phone or any mobile device.

APPENDIX

User Guide

The user first needs to specify locations of the folders containing the sequential images and the 3D CAD model. The executable is located in the build folder. To run the program the user must type the following command in the terminal.

```
. /main  path/to/3D/CAD/Model  path/to/image/folder
```

Then the 3D CAD model will load and displayed along with the first image of the image sequence. The user must specify the planes which are to be inspected by clicking once anywhere on the plane for each plane. After selecting the desired planes the user should select any window or specific entity that is also seen in the 3D CAD model from the first image. User must click on the four corners of that window both in the images and the CAD model and then press "ENTER".

The SLAM will run and generate a 3D point cloud and the spatial database will automatically be generated. It will also generate the stitched images and a text file named 'plane_boundaries.txt' in the 'Output' folder.

After the spatial database is generated, the user can click anywhere in the inspection area to see the relevant image of that location. After the image is shown, the user can click on different places on the image to check the metric distance. To visualize any other area, the user must click "ESC" first and then click again on the CAD model. User can repeat this process as many times as they want.

To get the textured visualization, the user should run the following command in the terminal

```
./visualizeCa
```

This command will use the 'plane_boundaries.txt' and the stitched images from the 'Output' folder and load the textured visualization. Pressing 'q' during both program executions will

terminate the program.

Spatial Database

Spatial database means data related to space. The space in this case can be a geographic space, a man-made building space, a user defined coordinate space or even a layout of a VLSI design for example. The need for managing a large number of data present in the space has gotten a lot of interest with the advent of relational database systems. A spatial data system must at least be able to query for a data from a large collection of objects from a particular area without searching through the whole dataset.

Modeling of Spatial Database

Spatial databases can be of many types. The following are the components who represent a spatial database:

(i) Data in space: Distinct entities are arranged in space with their own geometric description.

(ii) Space: The space where the data is stored must be defined.

Spatial Query

A statement or logical expression that selects geographic features based on location or spatial relationship. For example, a spatial query might find which points are contained within a polygon or set of polygons, find features within a specified distance of a feature, or find features that are adjacent to each other. There can be two main types of operations for manipulating sets of database elements with spatial attributes. Such as:

(i) Spatial Selection

(ii) Spatial Join

Spatial Selection

Spatial selection is an operation that returns a set of elements fulfilling the predicates from the spatial database. Finding all state parks in North Dakota in a map is an example of spatial selection.

Spatial Join

Spatial join begins with a spatial selection and comparing different domains of elements in that selected space. For example, if a person is looking for fishing opportunities in state parks of North Dakota. A spatial selection can bring up all the state parks in North Dakota. Then considering fishing opportunities in those state parks is a spatial join. State park is a domain and fishing is another but in this case, they share the same spatial predicates. A spatial query must be supported by spatial indexing.

Spatial Indexing

Spatial indexing is mainly a support for spatial selection, spatial join or finding objects in space closest to the query value. Spatial indexing is a way to organize the space and the objects in that space so that only part of the space and related objects answer to a spatial query [32].

Camera Calibration

Camera calibration is a process to estimate the parameters of a lens and image sensor of a camera. These parameters can be used to correct distortions in an image, to measure the size of an object in world or to locate the camera in the world.

Camera parameters include,

iv) Intrinsic camera parameters

v) Extrinsic camera parameters

vi) Distortion Coefficients.

Estimation of camera parameters requires 3D-2D correspondence of an object. Generally, a camera calibration pattern (Checker Board Pattern) is used to find those correspondences. The camera model that has been used in this work is based on Pinhole Camera model. The camera parameters are expressed by a 4×3 matrix known as a Camera Matrix. The camera matrix maps the 3D world scene in a 2D image plane. The extrinsic parameters represent camera poses during each image capture in 3D world.

Calibration of Canon EOS 5D Mark III camera was done using OpenCV's camera calibration code that is provided within OpenCV's source library [33]. This camera calibration code provides several algorithms to compute the intrinsic and distortion properties found in pinhole cameras. Distortion is typically caused by symmetry found in the lens used with the camera [33]. OpenCV takes into account both radial and tangential distortion that can be found in cameras. Radial distortions are classified as barrel, pincushion, or a combination of the two which cause straight lines to no longer appear straight in distorted images. In barrel distortion, lines appear to bend away from the center of the image in a "fish-eye" effect whereas pincushion distortion causes lines to bend towards the camera. The following formulas provides the equations used in OpenCV for

correction of the radial distortion [33]

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (1)$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (2)$$

This states that for a pixel at location (x, y) coordinates in the input image, the corrected output image without distortion will be the new $(x_{corrected}, y_{corrected})$ coordinates [33].

Tangential distortion is caused by the camera lenses not being perfectly parallel to the imaging plane. This is corrected as [27]

$$x_{corrected} = x + 2\big(\, p_1\, xy + \, p_2(r^2 + 2x^2)\big) \quad (3)$$

$$y_{corrected} = y + 2(\, p_1\, (r^2 + 2y^2) + \, p_2 xy) \quad (4)$$

The five parameters listed are provided by OpenCV once the camera calibration is finished and presented as a one row matrix with 5 columns in an .xml file as

$$Distortion_{coefficients} = (k_1 k_2 p_1 p_1 k_3) \quad (5)$$

Intrinsic parameters include parameters inherent to the camera itself. These include the focal length of the camera and the image offset. The focal length is the distance from the center of a lens and its focus and represents how strongly the system converges the light onto the optical sensor. Image offset which represents the optical center of the camera in pixel coordinates. OpenCV creates a camera matrix of these unknown parameters as the 3-by-3 matrix below

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_y \\ 0 & f_y & c_x \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (6)$$

Where f_x and f_y represent the focal lengths in the x and y directions, c_y and c_x represent the image offset, and w is simply used to homography the coordinate system. Solving all the unknown parameters mentioned previously is accomplished through the calibration.

OpenCV solves these unknowns through a series of several geometric equations using a known

calibration object or pattern [33]. The patterns supported by OpenCV's algorithm include chessboards, circular grids, and asymmetrical grid patterns. OpenCV first locates the corners or circles of the pattern and determines where that plane lies in 3D space in relation to the camera. Knowing the 3D location of the object provides 6 parameters for rotation and translation.

OpenCV solves the distortion and camera parameters separately, focusing first on the distortion parameters [33]. Due to distortion parameters being tied to the 2D geometry of how the image is warped, a known calibration pattern can provide the information necessary to solve these parameters. Technically a simple three corner pattern providing 6 pieces of information would provide enough information to solve the distortion parameters but more are needed for robustness.

Intrinsic parameters are also solved through use of the calibration pattern. For each view of the calibration pattern 6 unknown values for the extrinsic parameters are developed due to the rotation and translation of the object along with the previously mention 4 intrinsic parameters. To solve for these 10 unknowns, a pattern of N corners or circles can be used to provide 2NK constraints, where K is the number of images, due to each point having an x and y position. Solving this series of equations then requires that $2NK \geq 6K+4$. This does require multiple images though due to one image only providing 4 corners worth of information since that is all that is needed to describe a plane. So, a pattern with enough point and multiple images can be used to solve the intrinsic parameters [33].

Solution of Perspective Three-Point (P3P) Problem

Solution of P3P problem is used to estimate the position and orientation of an object relative to the position and orientation of the camera or vice versa.

With four given 3D/2D correspondences where points in 2D are defined in image coordinate system and 3D points are given in model coordinate system ($A \leftrightarrow u$, $B \leftrightarrow v$, $C \leftrightarrow w$, $D \leftrightarrow z$), three points (A, B, C) are used to solve P3P equations. The solution gives up to 4 possible sets of distances. All the solutions are converted into pose configurations (up to four) and the fourth

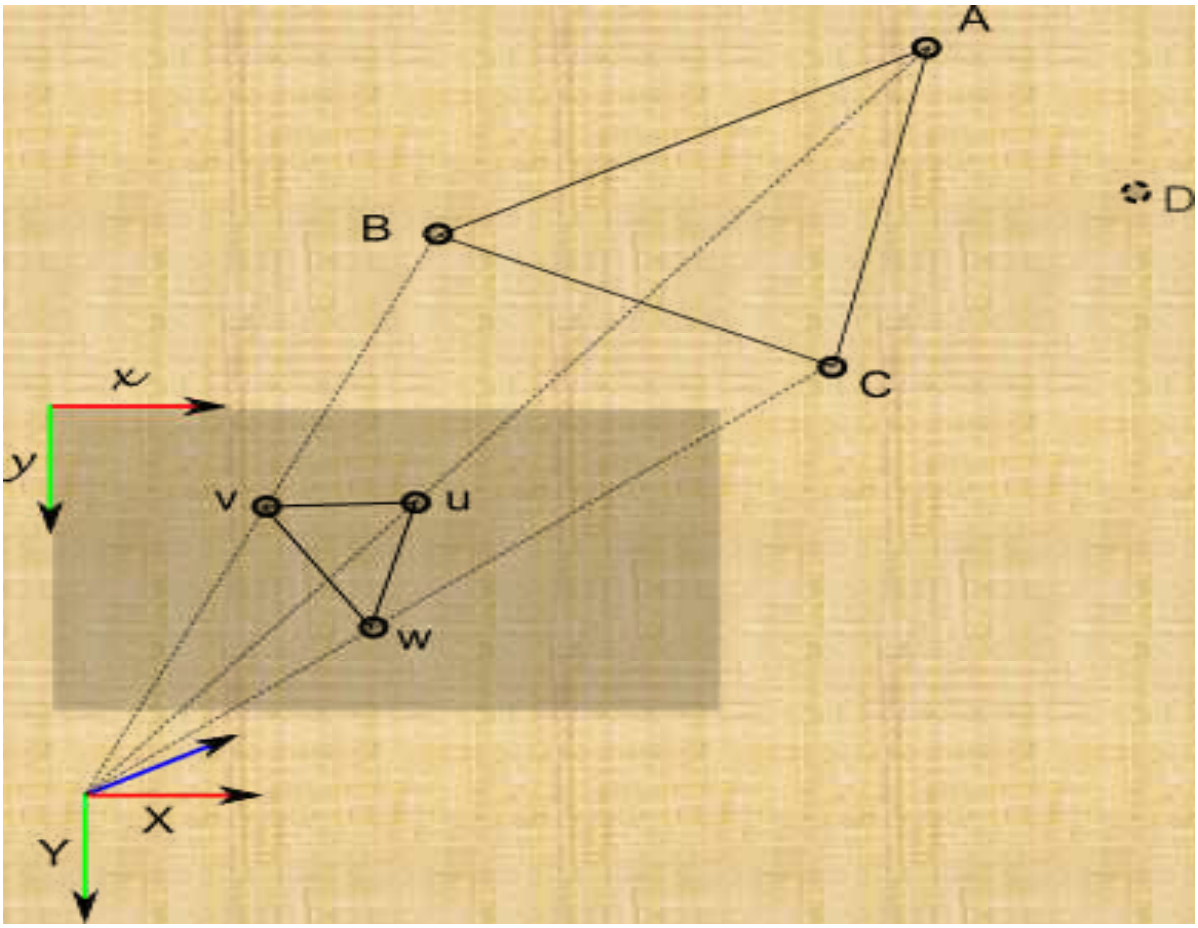

Figure I: Image Projection: A, B, C projected in u, v, w [34]

point D is used for disambiguation [23].

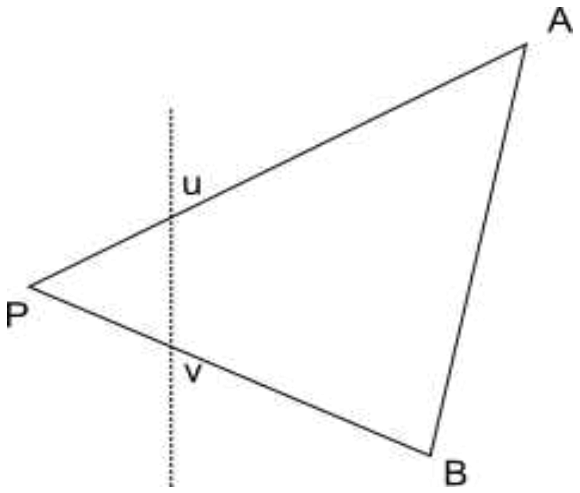

Figure II: Law of cosines in 2D (u & v denote A & B projections on the image plane) [34]

The main principle behind this algorithm is the law of cosines [28]. Using angles, it expresses the length of each sides of a triangle. If a triangle *ABC* is the triangle which being used to solve the P3P problem and P is the camera optical center, and *u, v* are projections of *A* and *B* respectively, triangle *ABP* gives,

$$PA^2 + PB^2 - 2\times PA\times PB\times cos\alpha_{u,v} = AB^2. \quad (7)$$

So, applying this law of cosines the P3P equation system becomes,

$$PB^2 + PC^2 - 2\times PB\times PC\times cos\alpha_{v,w} - BC^2 = 0 \quad (8)$$

$$PA^2 + PC^2 - 2\times PA\times PC\times cos\alpha_{u,w} - AC^2 = 0 \quad (9)$$

$$PA^2 + PB^2 - 2\times PA\times PB\times cos\alpha_{u,v} - AB^2 = 0. \quad (10)$$

Dividing these equations with PC^2 and letting $y = PB/PC$ and $x = PA/PC$,

$$y^2 + 1 - 2\times y\times cos\alpha_{v,w} - \frac{BC^2}{PC^2} = 0 \quad (11)$$

$$x^2 + 1 - 2\times x\times cos\alpha_{u,w} - \frac{AC^2}{PC^2} = 0 \quad (12)$$

$$x^2 + y^2 - 2 \times x \times y \times cos\alpha_{u,v} - \frac{AB^2}{PC^2} = 0 \quad (13)$$

are obtained. Finally, introducing $v = \frac{AB^2}{PC^2}$, $av = \frac{BC^2}{PC^2}$ and $bv = \frac{AC^2}{PC^2}$ we get,

$$y^2 + 1 - 2 \times y \times cos\alpha_{v,w} - av = 0 \quad (14)$$

$$y^2 + 1 - 2 \times x \times cos\alpha_{u,w} - bv = 0 \quad (15)$$

$$x^2 + y^2 - 2 \times x \times y \times cos\alpha_{u,v} - v = 0. \quad (16)$$

Replacing

$$v = x^2 + y^2 - 2 \times x \times y \times cos\alpha_{u,v}$$

in first two equations, a simplified version of P3P equation system is obtained as,

$$(1 - a)y^2 - ax^2 - cos\alpha_{v,w}y + 2acos\alpha_{u,v}xy + 1 = 0 \quad (17)$$

$$(1 - a)y^2 - ax^2 - cos\alpha_{v,w}y + 2acos\alpha_{u,v}xy + 1 = 0 \quad (18)$$

Above equations can be solved by using the Wu Ritt's zero decomposition method [35].

Bibliography

[1] Qadir, A.” A Large Scale Inertial Aided Visual Simultaneous Localization and Mapping (SLAM) System for Small Mobile Platforms,” Ph.D. dissertation, Dept. Mech. Eng., Univ. of North Dakota, Grand Forks, ND, 2016.

[2] Klein, G., & Murray, D. (2007, November). Parallel tracking and mapping for small AR workspaces. In Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on (pp. 225-234). IEEE.

[3] Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics, 31(5), 1147-1163.

[4] Lowy, J. "Drone sightings up dramatically (update)," 2014. [Online]. Available: http://phys.org/news/2014-11-drone-sightings.html. dramatically/SrrcKzjnphejy4Zvf3OaEI/story.html.

[5] Jansen, B. "FAA: Drone sightings on pace to quadruple this year," USA TODAY, 2015. [Online]. Available: http://www.usatoday.com/story/news/2015/08/13/d rone-sightings-faa-newark/31614151/.

[6] Siebert, S., & Teizer, J. (2014). Mobile 3D mapping for surveying earthwork projects using an Unmanned Aerial Vehicle (UAV) system. Automation in Construction, 41, 1-14.

[7] Eschmann, C., Kuo, C. M., Kuo, C. H., & Boller, C. (2012, July). Unmanned aircraft systems for remote building inspection and monitoring. In 6th European workshop on structural health monitoring (pp. 1-8).

[8] Yue, K., Huber, D., Akinci, B., & Krishnamurti, R. (2006, June). The ASDMCon project: The challenge of detecting defects on construction sites. In 3D Data Processing, Visualization, and Transmission, Third International Symposium on (pp. 1048-1055). IEEE.

[9] Akinci B., Boukamp F., Gordon C., Huber D., Lyons C., and Park K. (2006). A formalism for utilization of sensor systems and integrated project models for active construction quality control. Automation in Construction, Vol. 15, No. 2, 124-138.

[10] Jaselskis E., Cackler E., Walters R., Zhang J., and Kaewmoracharoen M. (2006). Using scanning lasers for realtime pavement thickness measurement. CTRE Project 05-205, National Concrete Pavement Technology Center, Iowa State University.

[11] Bosche F. and Haas C.T. (2008). Automated retrieval of 3D CAD model objects in construction range images. Journal of Automation in Construction, Vol. 17, No. 4, 499-512.

[12] Teizer J., Kim C., Haas C., Liapi K., and Caldas C. (2005). Framework for real-time three-dimensional modelling of infrastructure. Geology and Properties of Earth Materials 2005, Transportation Research Board Natl

Research Council, Washington, 177-186.

[13] El-Omari S. and Moselhi O. (2008). Integrating 3D laser scanning and photogrammetry for progress measurement of construction work. Journal of Automation in Construction. Vol. 18, No.1, 1-9.

[14] Su Y., Hashash Y., and Liu L.Y. (2006). Integration of construction as-built data via laser scanning with geotechnical monitoring of urban excavation. Journal of Construction Engineering and Management, Vol. 132, No. 12, 1234-1241.

[15] Klein, L., Li, N., & Becerik-Gerber, B. (2012). Imaged-based verification of as-built documentation of operational buildings. Automation in Construction, 21, 161-171.

[16] Golparvar-Fard, M., Peña-Mora, F., & Savarese, S. (2011). Integrated sequential as-built and as-planned representation with D 4 AR tools in support of decision-making tasks in the AEC/FM industry. Journal of Construction Engineering and Management, 137(12), 1099-1116.

[17] Snavely, N., Seitz, S. M., & Szeliski, R. (2006, July). Photo tourism: exploring photo collections in 3D. In ACM transactions on graphics (TOG) (Vol. 25, No. 3, pp. 835-846). ACM.

[18] Debevec, P. E., Taylor, C. J., & Malik, J. (1996, August). Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (pp. 11-20). ACM.

[19] Sinha, S. N., Steedly, D., Szeliski, R., Agrawala, M., & Pollefeys, M. (2008, December). Interactive 3D architectural modeling from unordered photo collections. In ACM Transactions on Graphics (TOG) (Vol. 27, No. 5, p. 159). ACM.

[20] Xu, K., Zheng, H., Zhang, H., Cohen-Or, D., Liu, L., & Xiong, Y. (2011, August). Photo-inspired modeldriven 3D object modeling. In ACM Transactions on Graphics (TOG) (Vol. 30, No. 4, p. 80). ACM.

[21] Colburn, A., Agarwala, A., Hertzmann, A., Curless, B., & Cohen, M. F. (2013). Image-based remodeling. IEEE transactions on visualization and computer graphics, 19(1), 56-66.

[22] Karsch, K., Golparvar-Fard, M., & Forsyth, D. (2014). ConstructAide: analyzing and visualizing construction sites through photographs and building models. ACM Transactions on Graphics (TOG), 33(6), 176.

[23] Kneip, L., Scaramuzza, D., & Siegwart, R. (2011, June). A novel parametrization of the perspective three-point problem for a direct computation of absolute camera position and orientation. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on (pp. 2969- 2976). IEEE.

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

[25] Hartley, R., Zisserman, A., Multiple View Geometry in Computer Vision, Cambridge University Press, New York, NY, 2003

[26] Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. JOSA A, 4(4), 629-642.

[27] Suzuki, S. (1985). Topological structural analysis of digitized binary images by border following. Computer vision, graphics, and image processing, 30(1), 32-46.

[28] Douglas, D. H., & Peucker, T. K. (2011). Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. Classics in Cartography: Reflections on Influential Articles from Cartographica, 15-28.

[29] Brown, M., & Lowe, D. G. (2007). Automatic panoramic image stitching using invariant features. International journal of computer vision, 74(1), 59-73.

[30] D. A. R. Reserved, "Spreading wings S1000+," 2016. [Online]. Available: http://www.dji.com/spreading-wings-s1000-plus.

[31] "Generate 2D and 3D Information, Purely from Images with Pix4D." Pix4D. N.p., n.d. Web. 09 Apr. 2017 retrived from. https://pix4d.com/

[32] Güting, Ralf Hartmut. "An introduction to spatial database systems." *The VLDB Journal—The International Journal on Very Large Data Bases* 3.4 (1994): 357-399.

[33] Bradski, Gary, and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[34] *P3P (Perspective-Three-Point): an overview | iplimage*. (2017). *Iplimage.com*. Retrieved 12 June 2017, from http://iplimage.com/blog/p3p-perspective-point-overview/

[35] Chou, Shang-Ching, and Xiao-Shan Gao. "Ritt-Wu's decomposition algorithm and geometry theorem proving." *10th International Conference on Automated Deduction*. Springer Berlin/Heidelberg, 1990.