

Master Semester Project Report (2017 Spring)

**Experimental indoor localization using
distance measurements obtained from visual data**

Yuan Yao

June 19, 2017



Master Student: Yuan Yao

Project Supervisor: Frederike Dümbgen, Robin Scheibler

Project Director: Prof. Martin Vetterli

Date: June 19, 2017

1 Abstract

In recent years, the emergence of artificial intelligence increases the demand of automatic and robust localization outdoors and indoors. While GPS provides enough accuracy in most outdoor cases, there is still a lack of robust and efficient indoor localization systems available on the market.

In this report, an experimental framework for indoor localization is developed and tested. To operate an automatic robot owned by LCAV laboratory, two different operation modes have been successfully implemented, including controlling a robot in real-time or with extra input containing a list of commands. Also, a new visual fiducial system have been developed, and is able to capture the locations of Apriltags inside a room accurately. Multidimensional scaling (MDS) and Squared range least square (SRLS) algorithms containing distance information will also be introduced and the final localization result is within 2% error of tolerance compared with the ground truth results measured by a laser meter.

All the codes and results are documented for later use for indoor localization and related experiments within the laboratory.

2 Acknowledgement

First of all, I would like to express my sincere gratitude and thankfulness to my project supervisor, Frederike Dümbgen and Robin Scheibler for their continuous, patient and selfless support and help of my project and related research. Their guidance helped me during all time of research and writing of this report. What's more, their advice and enthusiasm not only encourage me finishing this project as perfect as possible but also influence my studying and daily life.

Besides my supervisors, I am also grateful to Professor Martin Vetterli and all the members of LCAV who provide me an opportunity to begin this semester project, and also give me access to the laboratory and research facilities.

Last but not the least, I would like to thank my family and Wenyuan supporting me spiritually throughout the experiment.

Contents

1 Abstract	1
2 Acknowledgement	2
3 Experimental Setup	4
3.1 Robot	4
3.2 Camera Setup	6
4 Robot Movement	6
4.1 Real-time Movement	6
4.2 Command In-advance Movement	7
5 Camera Calibration	7
5.1 Intrinsic Matrix Calibration	9
5.2 Extrinsic Matrix Calibration	10
6 Visual Localization	10
6.1 Accessory Tools for Localization	10
6.2 Experimental Setup	11
6.3 Data Reconstruction	12
7 Experimental Results	14
7.1 Intrinsic Matrix Calibration	14
7.2 Localization Experiment Sample	16
7.3 Data Reconstruction	19
7.4 Room Experiment	20
8 Conclusion	26
9 Learning Experience	27

3 Experimental Setup

The experimental robot used in this experiment is designed and produced by Gigatec S.A. As shown in Figure 1, the robot is driven by two DC motors mounting on two 31cm diameter wheels. The battery of the robot is located inside the white box behind the chest. The chest includes a actuator for the moving movements. Also, as shown in Section 2, there are three web cameras attached with each others and will be part of robot in order to localize the robot while moving in the room.



Figure 1: Outlook of robot



Figure 2: Outlook of web camera

The two motors are equipped with an encoder of 256 pulses per revolution, and the processors on the motors can detect twice more pulses, which means, for each revolution of the motor axis, the board will detect 512 pulses as the motor is coupled with a 43:1 gearbox, the total of pulses seen for a wheel revolution will be 22016.

The three web cameras are run under *Raspberry Pi* module can be controlled programmatically and remotely with specific IP addresses. When connecting with a same wifi address, operator can control theses web cameras directly through local networks.

3.1 Robot

The main usage for this robot is to simulate a person moving inside a room point to point. The distance between two wheels are $L_{wheel} = 40\text{cm}$. Given the left and right record of pulses (p_{left}, p_{right}), it is possible to calculate the relative moving distances from point to point.

Under some reasonable hypotheses that the wheels won't slip while moving and the pulse encoder is able to record a same amount of pulses per resolution, $N_{pulse} = 22016$, pulse measurement can be converted into trajectory distance:

$$l_{left} = \frac{2\pi R_{wheel} p_{left}}{N_{pulse}} \quad l_{right} = -\frac{2\pi R_{wheel} p_{right}}{N_{pulse}}$$

There is a minus in the expression of l_{right} because these two pulse encoders and motors are facing each other and count with two directions. Therefore, only when $l_{left} = -l_{right}$ does the robot move straight forward. Due to the limitation of its mechanical design, it can only move with these four commands: forward, backward, right rotation and left rotation. If the robot want to turn right, it has to do right rotation at first with certain angle and then forward to the certain place. The relationships between command and trajectory distance are shown below: (assuming $l_{left} > 0$ represents forward)

Table 1: command and trajectory distance

Command	Forward	Backward	Right Rotation	Left Rotation
$l_{left,right}$	$l_{left} = -l_{right}$	$-l_{left} = l_{right}$	$l_{left} = l_{right}$	$-l_{left} = -l_{right}$

As mentioned before, there is no way to let the robot turn right while moving forward, due to the fact that both wheels must have the same values of trajectory distances during each period. Thus, if the robot wants to move from a position to a certain position, there are many routes, and the shortest way is to rotate a certain angle at first and move forward to the target. Assume the starting position is (x_i, y_i, θ_i) , and the target position is $(x_{i+1}, y_{i+1}, \theta_{i+1})$. θ represents the angle between robot axis and global (room) y axis. Mathematically, the robot need to rotate θ' at first, and move forward distance s , given by:

$$\tan(\theta') = \frac{x_{i+1} - x_i}{y_{i+1} - y_i} \quad s = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad \theta_{i+1} = \theta_i + \theta' \\ \theta' = \frac{l_{left}}{L_{wheel}/2} = \frac{l_{left} + l_{right}}{L_{wheel}} \quad s = l_{left} = -l_{right}$$

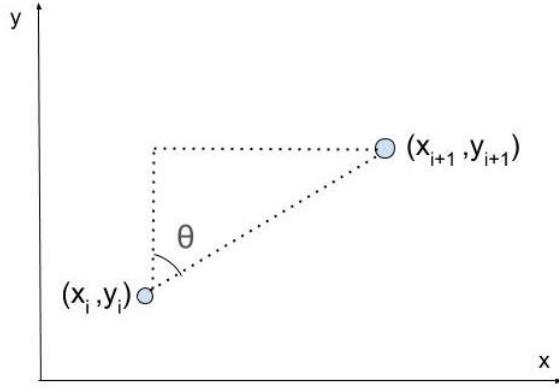


Figure 3: Geometry Example of Robot Position

With the help of these formulas, it is possible to move a robot from one point to another coordinate with simple commands. More information will be shown in Section 3.

3.2 Camera Setup

The cameras used in this experiment has been fully implemented by Frederike Dümbgen before, more information on camera hardware and setup can be reference in [1] [2]. There are still some important points which should be mentioned in this report.

First of all, these three cameras use *Raspberry Pi 2* camera module to take still photographs as well as videos. The Operating System loaded on the Raspberry Pi 2 is a Raspbian WHEEZY system. [2] All cameras have been set to connect to wireless network *korebot* automatically.

It is possible to preview the live images through web browsers directly before experiment. However, the live preview moves unstably so that we are supposed to move the cameras slowly and carefully.

4 Robot Movement

In different situations, it is better to use different modes to control the robot. For example, if we want to move the robot randomly without specific routes, it is better to control the robot in real time, which means inputing commands while the robot is moving. If an operator is willing to set the robot's route in advance, it is better to give the robot commands before experiment and the robot will move and record measurement data automatically.

4.1 Real-time Movement

In real time movement mode, the operator is able to control the robot while it is moving. There are 6 available commands for real-time movement: '*f*' for forward, '*b*' for backward, '*r*' for right rotation, '*l*' for left rotation, '*s*' for stop and '*end*' for ending the movement experiment. Between each command, the movement system will record the number of pulses at both motors as shown in Listing 1. The left column contains the commands we send during experiment, and the numerical lines are the target encoder readings of the two motors. The differentials of values of pulses are the absolute number of pulses during two commands, from which we are able to convert pulse measurement to trajectory distance and position coordinate simply by equations in Section 3.1.

Listing 1: output.txt

<i>s</i>		
-10048		-10837
<i>b</i>		
-15031		-4780
<i>s</i>		
-15031		-4780
<i>l</i>		
-2003		9402
<i>s</i>		
-2003		9402
<i>f</i>		
4030		3528
<i>s</i>		
4030		3528

However, when we run the experiment, we find there exist some problems affecting the performance of the robot. The first one is the time delay caused by wireless transmission. When command is typed on computer, it needs time to transmit the signal to robot by wifi. Thus, the robot will still move a short distance even if we type *end* command. The other problem is that the robot cannot reach its setting speed immediately, which means the robot needs time to accelerate so that the robot won't reach its destination precisely. These discrepancies are not a major problem for the experiment because we can control the robot in real time and correct the errors while robot is moving.

4.2 Command In-advance Movement

If the operator is willing to control the robot automatically, he is supposed to write the commands in an extra file. A typical command file is shown in Listing 2 shown below:

Listing 2: command.txt

0	s
1	r
3	s
4	f
4 . 8	s
5	b
7 . 5	s

In *command.txt*, the first column represents the absolute time when sending the command and the command is shown in the second column. For example, the robot will start at time 0, start turning right at 1 second, stop turning right at 3 second, start forward at second 4, stop forward at second 4.8, start backward at second 5 and end movement at second 7.5.

Basically, there is no error in this mode because we only define the movements of robot instead of its expected location. However, if we analyze the output of the pulse measurement, we are able to find out that the moving distance is not simply the same as theoretical value shown in Section 3.1, and usually bigger than theoretical value because of the acceleration process and wireless connection problems described in Section 3.1. What's more, if we want to use the equations in Section 3.1 to calculate the locations of robot, there will be an unavoidable problem with the rotation of robot. We make an assumption before that the wheels won't slip while moving, however, due to the limitation of robot's mechanical design, the robot will definitely slip while rotating. What's more, the slipping distances are different each time, depending on the friction coefficient of the floor and the wheels, as well as the power energy supplied by the batteries. Thus, even if in-advance movement sounds convenient and easy to operation, it is nearly impossible to determine the robot's location in advance. Both styles have their drawback and the best way to overcome the acceleration problem is by setting the maximum velocity lower so that the robot is able to move around smoothly.

5 Camera Calibration

Camera calibration is a necessary and important step in three-dimensional reconstruction and object localization.[3] The basic calibration method will be developed below.

In a image, a 2-D point can be denoted by $\mathbf{m} = [u, v]^T$, and its corresponding point in 3-D world

is denoted by $\mathbf{M} = [X, Y, Z]^T$. We use $\tilde{\mathbf{m}} = [u, v, 1]^T$ and $\tilde{\mathbf{M}} = [X, Y, Z, 1]^T$ to denote the augmented vector in order to calculate not only rotation but also translation by a matrix multiplication[4]. The relationship between $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{M}}$ is given by

$$s\tilde{\mathbf{m}} = \mathbf{A}[\mathbf{R} \ \mathbf{t}]\tilde{\mathbf{M}}, \quad \text{with } \mathbf{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix},$$

where: s is the scaling factor, \mathbf{A} is a camera matrix, also known as intrinsic matrix. In \mathbf{A} , (c_x, c_y) is a principal point that is usually at the image center and (f_x, f_y) are the focal lengths expressed in pixel units. \mathbf{R} is the camera rotation matrix and \mathbf{t} is the camera translation vector. [4]

The intrinsic matrix does not depend on the scene viewed, which means once the intrinsic matrix is measured, it can be re-used as long as the focal length is fixed. Matrix $[\mathbf{R} \ \mathbf{t}]$ is called extrinsic matrix, which is used to describe the camera motion around a static scene[4]. We define two kinds of coordinate systems, the first one is camera coordinate system, where the location of camera is the original point of the coordinate system. The other one is global coordinate system, of which the original point is a fixed point in a room, and more information is described in Section 6.2. Then the extrinsic matrix is able to translate coordinates of a point (X, Y, Z) in camera coordinate system into (x, y, z) in global coordinate system. Combing intrinsic matrix with extrinsic matrix, the transformation above can also be written as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t},$$

where: $x/z = (u - c_x)/f_x$ and $y/z = (v - c_y)/f_y$. More generally, real camera lenses will contain some distortion, mostly radial distortion and some tangential distortion. In this case, the above parameters will rewrite as:

$$\begin{aligned} x'' &= \frac{x}{z} \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_1 \frac{xy}{z^2} + p_2(r^2 + 2\frac{x^2}{z^2}) \\ y'' &= \frac{y}{z} \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_2 \frac{xy}{z^2} + p_1(r^2 + 2\frac{x^2}{z^2}) \\ r^2 &= \frac{x^2}{z^2} + \frac{y^2}{z^2}, \quad u = f_x * x'' + c_x, \quad v = f_y * y'' + c_y, \end{aligned}$$

where $k_1, k_2, k_3, k_4, k_5, k_6$ are radial distortion coefficients, and p_1, p_2 are tangential distortion coefficients[4]. The following Figure 4 principally describe the coordinate model.

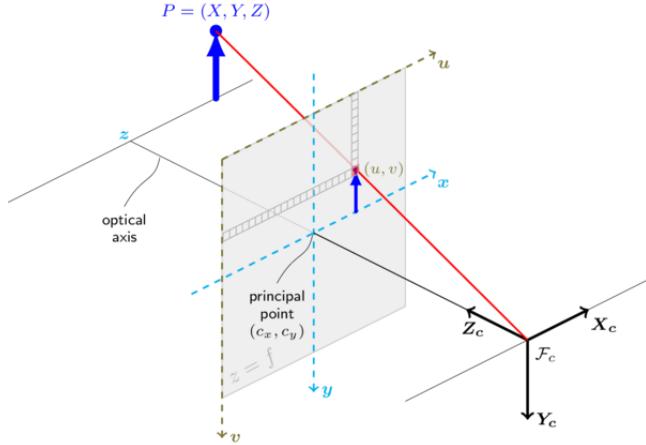


Figure 4: Camera coordinate system and global coordinate system[4]

5.1 Intrinsic Matrix Calibration

Due to the reason that intrinsic matrix of a camera won't change during the experiment, the first preparation step is to find out an accurate intrinsic camera matrix. The most famous and easiest method used for calibration algorithms is implemented by *OpenCV*. The principle of this algorithm is to provide some samples with well defined patterns, like the corners of a chess board in this case. Knowing the coordinates both in real world and in image, the algorithms is able to calculate both the distortion coefficients and focal lengths. In order to cover most of the distortions, at least 10 different pattern images are needed. Figure 5 is an identical example found online[5]. With this algorithm, the intrinsic matrix with distortion coefficients can be calculated. In this experiment, the distortion coefficients are so small that it is possible to ignore the distortion effects on image which will ease the following experiment.

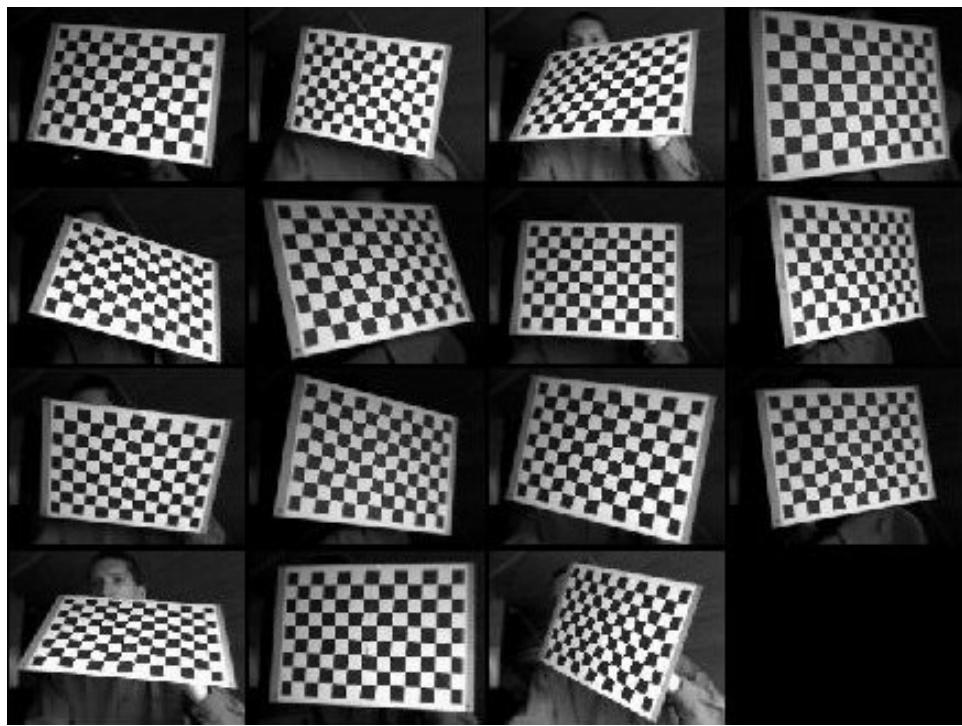


Figure 5: Example Images used for calibration[5]

5.2 Extrinsic Matrix Calibration

As explained in Section 5, the matrix $[\mathbf{R} \ \mathbf{t}]$ is called an extrinsic matrix, and the relationship in Section 5 can be rewritten as:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [\mathbf{R} \ \mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

where: r_{ij} ($i, j = 1, 2, 3$) are rotation matrix factors and t_i ($i = 1, 2, 3$) are translation vector factors. (X, Y, Z) is the position of a point in camera coordinate, and $(0, 0, 0)$ can represent as the center of camera. (x, y, z) is the position of this point in global coordinate frame. There are 12 unknown variables here but only three independent equations, thus three more points should be provided to solve this matrix. Solving extrinsic camera matrix means localizing the camera in global frame. More information on calibrating rotation and translation matrix will be given in Section 6.

6 Visual Localization

Vision-based navigation and localization has been widely used in recent research and industrial works, which is because visual information encodes contextual information on top of position. In this experiment, we are going to use a visual approach to localize our robot.

The main goal of localization is to find the position of the robot, along with camera, inside a room accurately and quickly. Several approaches have been proposed in recent literature to address visual localization indoors:

- 1) Triangulation [6] [7] [8] [9];
- 2) Image-based analysis [10];
- 3) Landmark features [11] [12] [13].

Triangulation relies on computation of angles and distances from the sensor to known points in environment. In indoor localization, the usual approaches are using signal sources like WiFi and Bluetooth. It requires some specific designed electrical objects and somehow is inconvenient in a room without enough electrical power [9]. Image-based analysis is a kind of scene analysis technique. By comparing image patterns with given reference datasets, it can localize an unknown environment, but cannot work under the zones without images covering. Landmark features uses global visual features to localize, like edges of room and vanish points in pictures. But it has difficulties in handling unknown locations [11].

In this experiment, we will come up with a new approach on localization indoors. We combine triangulation and image-based analysis methods by placing given image datasets inside a room and compute the distances of the cameras with respect to images.

6.1 Accessory Tools for Localization

We are considering using such image-marker that is simple to implement and easy to operate while receiving as much information as possible. Similar to QR code localization implemented by *Zhang* [14], Apriltags, designed by *Edwin Olson*, is also a well-known 2D barcode system for localization [15].

More than QR code, Apriltags can not only give us 6 degrees of freedom pose estimation, but also its size is not so complex that camera is able to detect them from a long distance, which makes localization in big room possible [16].

Figure 6 are some samples of Apriltags. An detection algorithm developed by *Edwin Olson* [15] is able to find four corners of an Apriltags in an image. With the help of a calibrated camera, we are able to receive all the position information of each Apriltag in the picture. Every Apriltag has its own unique id, which can be used as an index number for further estimation. And also, the visual information includes the distance from the camera to Apriltag, the 3D location of Apriltag in camera coordinate system, and the rotation relation (roll, yaw, pitch) between Apriltag and camera coordinate system.

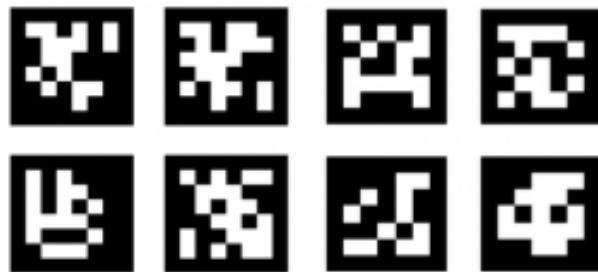


Figure 6: Samples of Apriltags[15]

6.2 Experimental Setup

As explained in section 5.2, more than four Apriltags are needed in one picture to compute the extrinsic matrix given in Section 5. Because more location information is able to provide more stable and accurate results, we would like to place as many Apriltags as possible but not that many that their calibrations become unwieldy. The camera coordinate system is shown in Figure 7 and we manually set the global coordinate system as room coordinate system. Figure 8 is a brief example containing enough Apriltags and global coordinate system. The Apriltags are pasted on the walls and floor, for each measurement, the camera is able to detect more than four Apriltags in its screen, and at the same time, our algorithm is able to calculate the extrinsic matrix of the camera with the input as locations information, and give the location of camera with respect of room coordinate system as output.

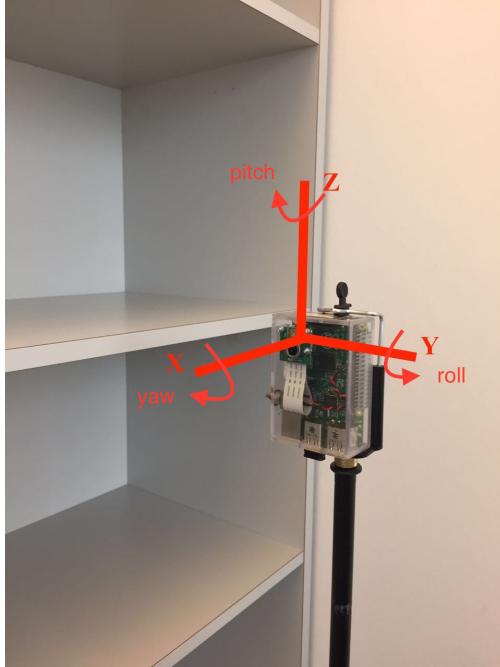


Figure 7: Camera Coordinate System

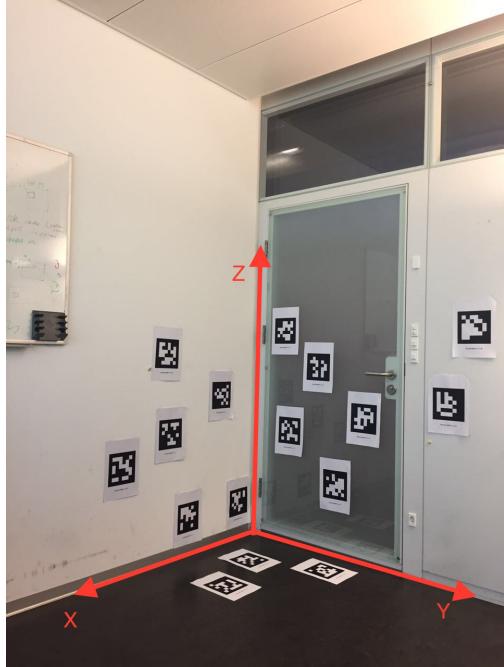


Figure 8: Room with Apriltags

6.3 Data Reconstruction

Demonstrated in Section 5.2, more than four locations of Apriltags are needed for camera localization. Define (x, y, z) and (X, Y, Z) as the positions of the i -th apriltags in the global and camera frames, respectively. Assume \mathbf{A} is the extrinsic matrix, then the extrinsic equation can be written as:

$$\begin{bmatrix} x_1 & \dots & x_i & \dots & x_n \\ y_1 & \dots & y_i & \dots & y_n \\ z_1 & \dots & z_i & \dots & z_n \\ 1 & \dots & 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 & \dots & X_i & \dots & X_n \\ Y_1 & \dots & Y_i & \dots & Y_n \\ Z_1 & \dots & Z_i & \dots & Z_n \\ 1 & \dots & 1 & \dots & 1 \end{bmatrix}$$

which can be rewritten as $\mathbf{y} = \mathbf{Ax} + \text{noise}$, where \mathbf{y} and \mathbf{x} are all $4 \times N$ matrices containing the locations of N Apriltags. In order to find out the extrinsic matrix which optimally meet the criteria above, we can minimize the value shown as:

$$\min_A \|\mathbf{y} - \mathbf{Ax}\|^2 = \min_A \|\mathbf{y}^T - \mathbf{x}^T \mathbf{A}\|^2$$

Knowing the numerical expressions of \mathbf{y} and \mathbf{x} , we can apply *Least Squares Method* algorithm to calculate extrinsic matrix \mathbf{A} regarding minimization. Afterwards, if we set $(X, Y, Z) = (0, 0, 0)$, then the equation comes out with $(x, y, z) = (t_x, t_y, t_z)$, which can be represented as the location of camera in global frame.

The value we calculate above represents as a measurement value which might contain noise and need to be corrected. We introduce two algorithms for data reconstruction in the following contents. Eculidean distance matrix (EDM) is a matrix representing a set of points in Eculidean Space. Proved by Dokmanic et al.[17], it is possible to apply Eculidean distance matrices (EDMs) and its relative algorithms for room reconstruction and sensor localization.

The representation of EDM is simple: consider a set of points $\{\mathbf{x}_i, i = 1, 2, \dots, n\}$ in a d-dimensional Euclidean Space, the Euclidean distance between each point is given as[17]

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2, \quad \mathbf{D} = [d_{ij}]$$

With the help of EDMs, we are able to apply a widely-used method, Multidimensional Scaling Method, for reconstruction. Multidimensional Scaling (MDS) is a group of algebraic technique used to optimally minimize and visualize the dissimilarities between each point in Euclidean Space. The main algorithm we use in this experiment is classical multidimensional scaling (c-MDS)[18] and we use a loss functions, stress function to minimize result errors. In this case, the only point we need to reconstruct is the location of sensor (cameras in our cases).

c-MDS is one of approaches reconstructing noisy point [17]. First, define a geometric centering matrix \mathbf{L} as:

$$\mathbf{L} = \mathbf{I} - \frac{1}{n} \mathbf{1} \mathbf{1}^T$$

where \mathbf{I} is an $n \times n$ identical matrix. Compute the singular value decomposition (SVD) of matrix $(1/2)\mathbf{LDL}^T$, then the solution of classical multidimensional scaling can be given as[17]:

$$(MDS): \quad MDS(\mathbf{D}) = \mathbf{U} \Sigma^{1/2}$$

where \mathbf{U} is a left singular matrix and Σ is a diagonal matrix corresponding to its SVD decomposition. And $MDS(\mathbf{D})$ is the optimal solution.

The same as c-MDS, squared range least square (SRLS) [19] is another approach optimizing the location of camera: Define \mathbf{s} as the sensor's coordinate vector, r_i as the noise observation between sensor and point i , and ε_i as the noise on each point i :[19]

$$r_i = \|\mathbf{s} - \mathbf{x}_i\| + \varepsilon_i, \quad i = 1, 2, \dots, n$$

$$(SRLS): \quad \min_s \sum_{i=1}^n (\|\mathbf{s} - \mathbf{x}_i\|^2 - r_i^2)^2 = \min_s \sum_{i=1}^n (\|\mathbf{s}\|^2 - 2\mathbf{x}_i^T \mathbf{s} + \|\mathbf{x}_i\|^2 - r_i^2)^2$$

Though it is a nonconvex function, however, it is possible to find its global minimum efficiently as shown by Amir et al.[19]:

$$\min_y \sum_{i=1}^n (\|\mathbf{Ay} - \mathbf{b}\|^2 : \mathbf{y}^T \mathbf{Dy} + 2\mathbf{f}^T \mathbf{y} = 0), \text{ where } \mathbf{y} = (\mathbf{s}^T, \|\mathbf{s}\|^2)^T$$

The optimal solution of above is given by[19]:

$$\mathbf{y}(\lambda) = (\mathbf{A}^T \mathbf{A} + \lambda \mathbf{D})^{-1} (\mathbf{A}^T \mathbf{b} - \lambda \mathbf{f}), \quad \text{where } \mathbf{y}(\lambda)^T \mathbf{Dy}(\lambda) + 2\mathbf{f}^T \mathbf{y}(\lambda) = 0$$

The final result \mathbf{y} will contain the optimal solution of camera location \mathbf{s} . We introduce two useful algorithms reconstructing distance data above, both are able to solve the problem of sensor localization. Compared with c-MDS, SRLS provides a more accurate result, but the computation is more complex than the former algorithm.

In Section 6, we introduce a convenient method for visual localization indoors: we use Apriltag as an image-based tool for visual localization. During the experiment, we are able to record all aspects of data including the locations of each reference objects in camera coordinate system, the Euclidean distance matrix corresponding to all interpoint and camera distance, and the measured location of the camera. With the help of two advanced algorithms, we can reconstruct and visualize our data for further analysis. In Section 7, we will apply these algorithms implemented by Frederike Dümbgen with the distance data received by our cameras.

7 Experimental Results

In this section, we will show you the experimental results from three parts. The first one is the results of intrinsic matrix from camera calibration. The second one is the reconstruction results computed by LS, MDS and SRLS algorithms from a corner of a small room. The last part is the reconstruction results in Room BC129. More information is shown below:

7.1 Intrinsic Matrix Calibration

We took 64 images of chess board with different perspectives, which are shown in Figure 9. As discussed in Section 5.1, at least 10 images are necessary for calibration, but furthermore, increasing number of images is able to increase the quality and accuracy of calibration. Thus, we run the calibration algorithm from using a subset of 10 out of 64 images, to all of them. The results of calibration are also shown below. Figure 10 shows the relationships between focal lengths and the number of images. From the figures we can see, the values of focal length (f_x and f_y) are around 1290 pixels with 10 to 20 numbers of images, and suddenly drop down to approximate 1257 pixels when using more than 20 images. It means the first 20 images are not different enough for a good calibration. And the curves become stationary after number of 25 which is around 1255 pixels. For a more accurate focal length, we calculate the average of last 35 values, and we get 1256 pixels for focal length f_x in x-axis and 1252 pixels for focal length f_y in y-axis.

Similarly, we plot the coordinates of the center point of the image shown in Figure 11 and 12. The values change varied before 40 images and turn to a constant after using 40 images. The average values in pixel of last 25 images are 694 for c_x and 373 for c_y . After running calibration iteratively, we compute a precise intrinsic camera matrix \mathbf{A} and will use it during the following experiment.

$$\mathbf{A} = \begin{bmatrix} 1256 & 0 & 694 \\ 0 & 1252 & 373 \\ 0 & 0 & 1 \end{bmatrix}$$

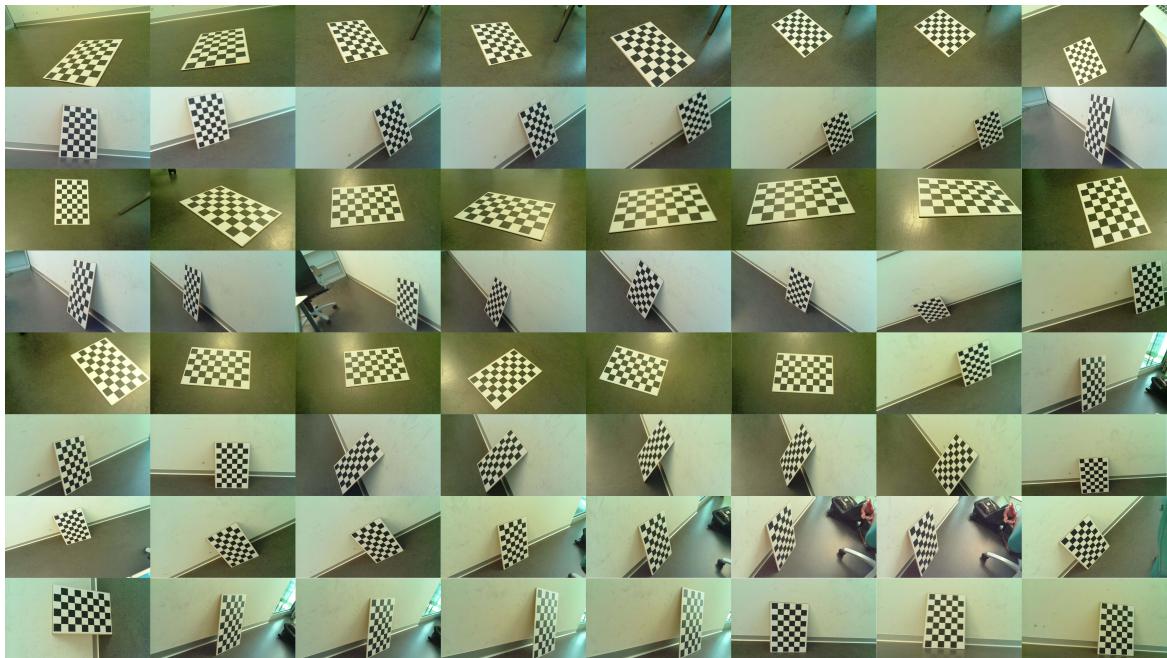


Figure 9: Images of chess board for calibration

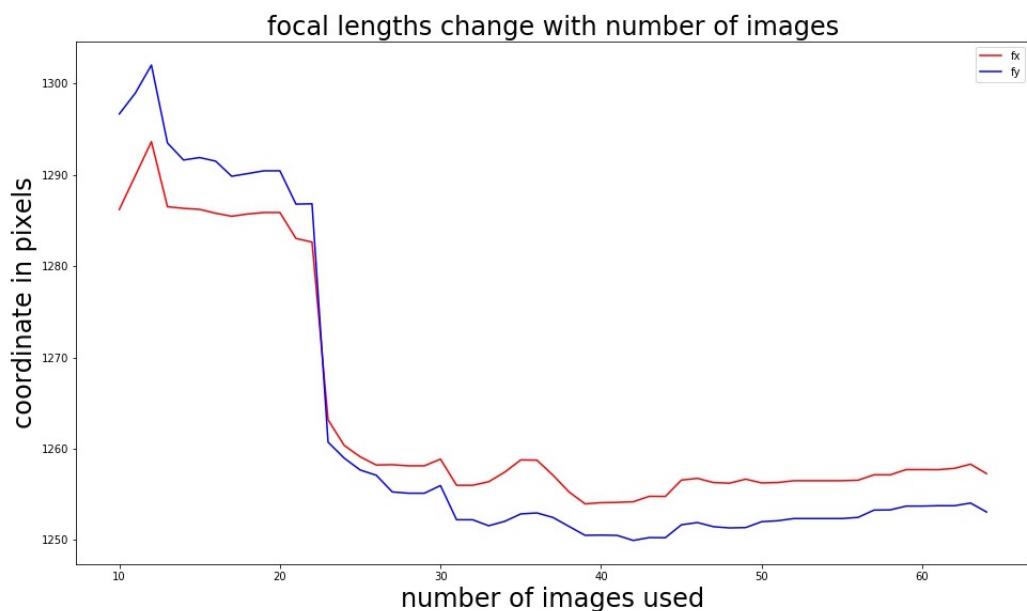


Figure 10: Changes of Focal length f_x and f_y

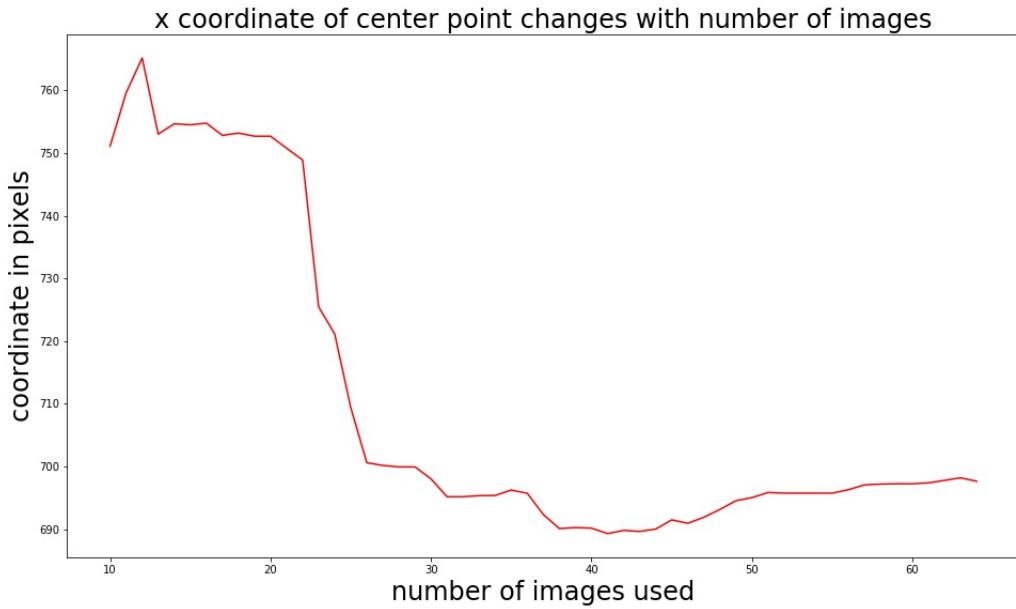


Figure 11: Changes of coordinate c_x

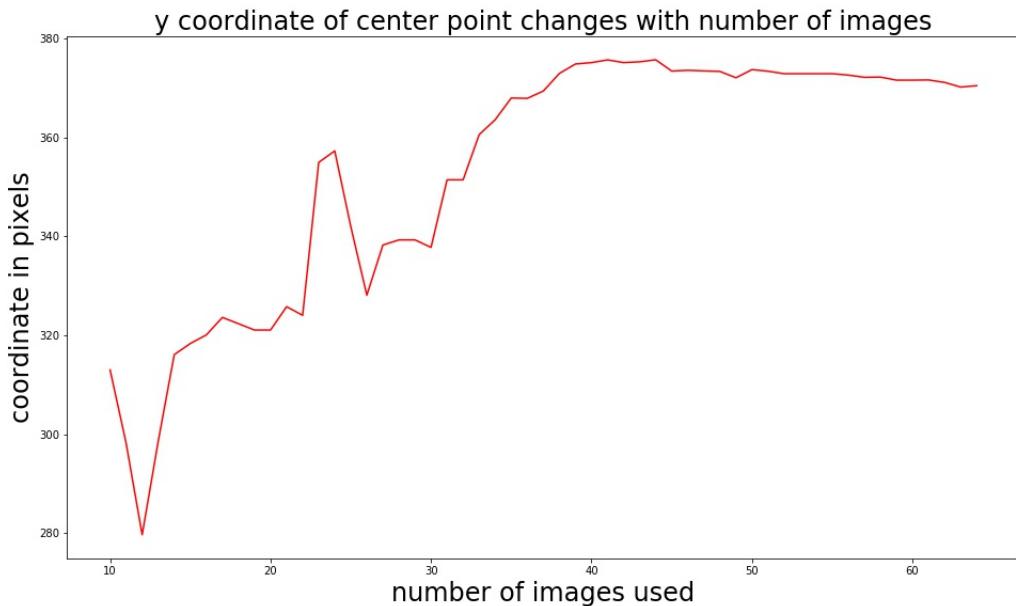


Figure 12: Changes of coordinate c_y

7.2 Localization Experiment Sample

As given in Section 6, Apriltags can be used as an image marker for indoors localization. In order to test the accuracy and feasibility of our localization system, we place 16 Apriltags in a room as shown in Figure 8. Table 2 is the locations of all Apriltags in room coordinate system. We use one camera for a sample experiment at different locations inside the room and use *Least Squares Method* computing

the extrinsic matrix for each measurement. As given in Section 6.3, (t_x, t_y, t_z) can be represented as the location of camera in room coordinate system. Compared with measurement value, we use laser meter to measure the ground truth value of the camera, and also, we compute their square root of MSE (RMSE) and the results are shown in table 3. Figure 13 represents 10 different localization perspectives with increasing distances from camera to the corner of room.

Table 2: Apriltags locations

Apriltags ID	1 st measurement	2 nd measurement	3 rd measurement	Average location
0	(0.151, 0, 0.212)	(0.150, 0, 0.211)	(0.151, 0, 0.212))	(0.151, 0, 0.212)
2	(0.730, 0, 0.691)	(0.729, 0, 0.689)	(0.732, 0, 0.690)	(0.730, 0, 0.690)
4	(0.576, 0, 0.205)	(0.576, 0, 0.205)	(0.575, 0, 0.207)	(0.576, 0, 0.206)
6	(0.330, 0, 0.888)	(0.329, 0, 0.889)	(0.330, 0, 0.888)	(0.330, 0, 0.888)
8	(0, 0.622, 0.403)	(0, 0.621, 0.405)	(0, 0.623, 0.404)	(0, 0.622, 0.404)
10	(0, 0.238, 1.294)	(0, 0.240, 1.294)	(0, 0.237, 1.294)	(0, 0.238, 1.294)
12	(0, 0.491, 1.079)	(0, 0.492, 1.079)	(0, 0.490, 1.078)	(0, 0.491, 1.079)
14	(0, 0.290, 0.677)	(0, 0.289, 0.676)	(0, 0.293, 0.676)	(0, 0.291, 0.676)
16	(0, 0.803, 0.800)	(0, 0.803, 0.802)	(0, 0.803, 0.801)	(0, 0.803, 0.801)
18	(0.397, 0.311, 0)	(0.396, 0.312, 0)	(0.398, 0.309, 0)	(0.397, 0.311, 0)
20	(0.444, 0.746, 0)	(0.444, 0.741, 0)	(0.444, 0.740, 0)	(0.444, 0.742, 0)
22	(0.233, 0.696, 0)	(0.232, 0.695, 0)	(0.231, 0.696, 0)	(0.232, 0.696, 0)
24	(0, 1.284, 0.920)	(0, 1.284, 0.921)	(0, 1.283, 0.922)	(0, 1.284, 0.921)
26	(0, 1.407, 1.310)	(0, 1.409, 1.311)	(0, 1.407, 1.311)	(0, 1.408, 1.311)
28	(0.711, 0, 1.134)	(0.712, 0, 1.137)	(0.710, 0, 1.135)	(0.711, 0, 1.135)
30	(1.046, 0, 0.545)	(1.045, 0, 0.546)	(1.046, 0, 0.544)	(1.046, 0, 0.545)

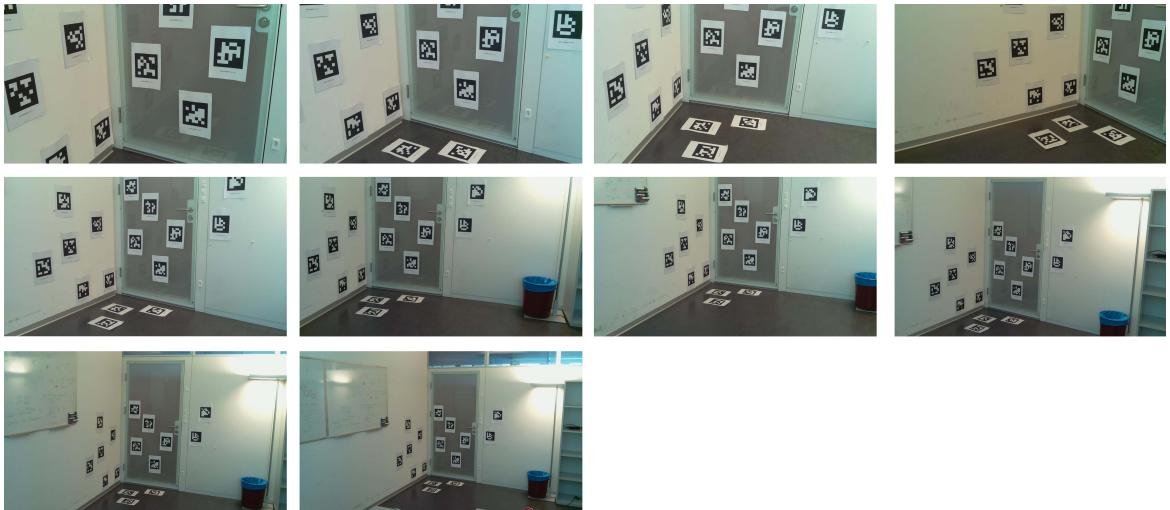


Figure 13: Different perspectives taking images

Table 3: Camera locations and their error measurement

ground truth location[m]	measured location[m]	RMSE[m]	distance [m]	error ratio
(1.577, 1.128, 1.145)	(1.589, 1.121, 1.161)	0.0217	2.252	0.96%
(1.836, 1.507, 1.125)	(1.880, 1.525, 1.164)	0.0611	2.628	2.33%
(2.422, 0.924, 1.223)	(2.472, 0.944, 1.251)	0.0612	2.866	2.14%
(1.907, 2.198, 1.129)	(1.939, 2.221, 1.152)	0.0458	3.121	1.47%
(2.650, 2.154, 1.130)	(2.601, 2.161, 1.132)	0.0494	3.597	1.37%
(3.343, 1.801, 1.201)	(3.310, 1.812, 1.215)	0.0373	3.983	0.94%
(3.709, 1.840, 1.204)	(3.643, 1.842, 1.204)	0.0658	4.312	1.53%
(3.638, 2.833, 1.591)	(3.586, 2.839, 1.605)	0.0543	4.878	1.11%
(4.389, 1.933, 1.613)	(4.306, 1.941, 1.607)	0.0832	5.060	1.64%
(5.010, 2.532, 1.748)	(4.920, 2.532, 1.755)	0.0903	5.879	1.54%

In order to analysis how bad the distance affects its corresponding results, we compute the error ratio of each measurement as:

$$\text{Error ratio} = \frac{\text{RMSE}}{\text{distance}}$$

We draw the relationship between distance to the origin point and error ratio in Figure 14. At the beginning, the value of error ratio changes a little bit because the camera cannot detect all Apriltags. As the distance increases, the camera is able to capture information from all 16 Apriltags, and the error ratio remains consistent. The final error ratio is around 1.5% and we are confident to say that our localization system is precise enough that the error is below 2% regardless of its position indoors.

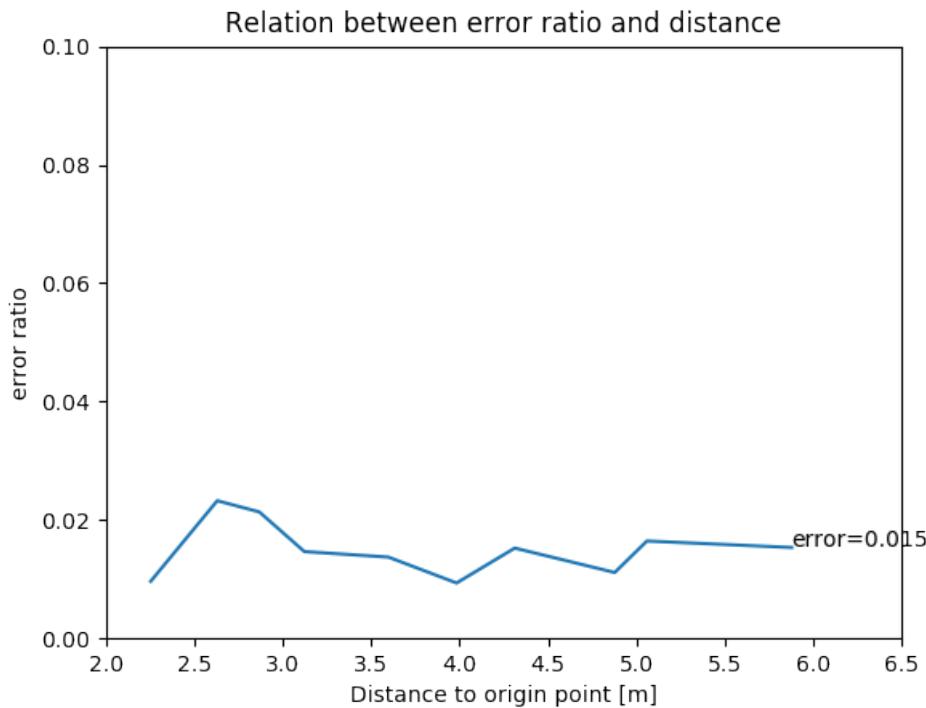


Figure 14: changes of error ratio

7.3 Data Reconstruction

As discussed in Section 6.3, we will use two different algorithms c-MDS and SRLS to denoise and reconstruct the camera location from noisy measurements. In this experiment, the locations of Apriltags are fixed, thus the only point needed to be reconstructed is the location of camera. As shown in Table 3 and Figure 14, the errors between measured location and ground truth value remain stable with distance increasing between camera and Apriltags. In order to minimize the complexity of figure presentation and computation, we choose the first image in Figure 13 for data reconstruction where 6 Apriltags are detected and their locations in global frame are in Table 2. Then, we apply MDS and SRLS algorithms to this datasets and the final results with Least Squares Method (LS) are shown in Table 4 and Figure 15, 16 and 17:

Table 4: Reconstruction results

	Ground truth location[m]	Computed location[m]	RMSE	error ratio
MDS	(1.577, 1.128, 1.145)	(1.607, 1.100, 1.013)	0.139	6.17%
SRLS	(1.577, 1.128, 1.145)	(1.603, 1.093, 1.019)	0.132	5.89%
LS	(1.577, 1.128, 1.145)	(1.589, 1.121, 1.161)	0.022	0.96%

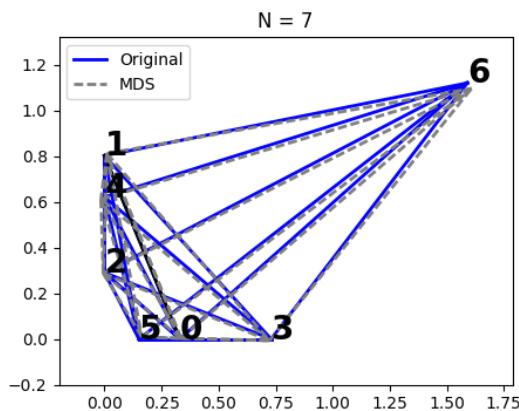


Figure 15: 2D reconstruction of MDS

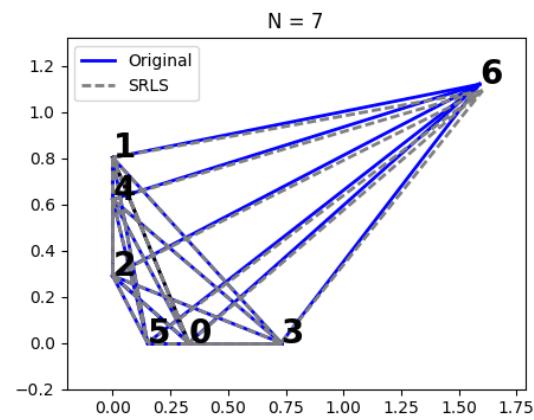


Figure 16: 2D reconstruction of SRLS

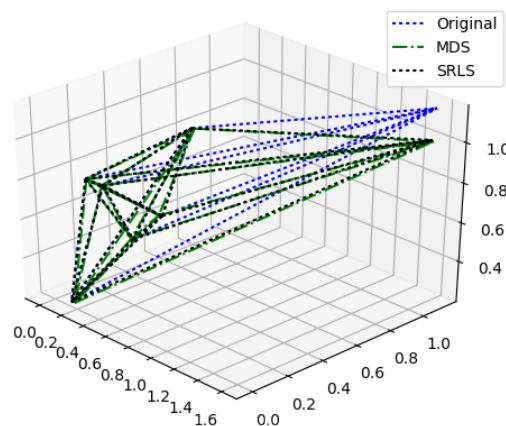


Figure 17: 3D reconstruction of MDS and SRLS

According to the figures above, SRLS works better than MDS in this case, but both are worse than LS algorithm. If we measure the time for running these two algorithms, we can see the running time of MDS is shorter than SRLS's. From the Figure 15 and 16 we can see, the MDS reconstructed locations of tags are slightly different from the original locations while SRLS reconstructed locations of tags are totally the same as origins. Thus, if we take the errors of Apriltags' locations into account, SRLS works better than MDS on camera location reconstruction.

7.4 Room Experiment

This visual localization system works well on the test sample, in order to prove that our system is universal in any space, we setup our experiment in Room 129 in BC Building. We place as many Apriltags as possible around the walls and measure the locations of each Apriltag corresponding to global coordinate. Figure 18, 19, 20 and 21 show the brief appearance of the Apriltags inside the room.

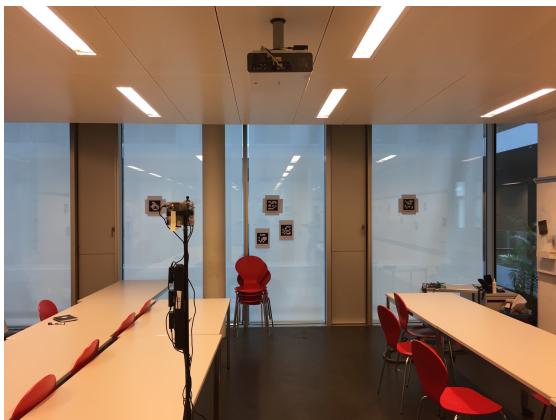


Figure 18: Apriltags inside the room



Figure 19: Apriltags inside the room



Figure 20: Apriltags inside the room



Figure 21: Apriltags inside the room

And also, we combine three cameras together, in order to capture images from all around the room at the same time. However, three cameras can't cover 360 degrees around the camera set, and there are some blind areas and should be improved in the future. The outlooks of the camera set are shown in Figure 22 and 23, and the locations of Apriltags are shown in Table 5:



Figure 22: Outlook of camera set



Figure 23: Outlook of camera set

Table 5: Apriltags location in room BC129

tagid	location[m]	tagid	location [m]	tagid	location [m]
122	(0, 0.933, 1.75)	142	(0.975, 0, 1.414)	141	(0, 0.714, 1.235)
118	(0, 1.693, 1.75)	116	(0, 2.513, 1.75)	120	(0, 3.123, 1.75)
114	(0, 3.685, 1.75)	110	(0, 4.361, 1.75)	108	(0, 4.899, 1.75)
36	(0, 5.539, 1.75)	35	(0, 6.243, 1.75)	46	(0.893, 7.102, 1.75)
38	(1.914, 7.102, 1.75)	40	(2.615, 7.102, 1.75)	42	(3.183, 7.102, 1.75)
44	(3.819, 7.102, 1.75)	138	(4.917, 7.102, 1.75)	124	(0.687, 0, 1.75)
126	(1.355, 0, 1.75)	128	(2.193, 0, 1.75)	112	(3.797, 0, 1.75)
130	(4.861, 0, 1.75)	132	(7.355, 1.119, 1.75)	134	(7.355, 3.15, 1.75)
136	(7.355, 4.837, 1.75)	76	(3.079, 0, 1.625)	52	(3.297, 0, 1.447)
80	(4.337, 0, 1.336)	64	(0, 2.79, 1.419)	60	(0, 3.385, 2.014)
62	(0, 3.921, 1.314)	68	(0, 4.583, 1.376)	66	(0, 5.231, 1.981)
33	(0.448, 7.102, 0.78)	48	(0.892, 7.102, 1.218)	58	(1.495, 7.102, 0.987)
50	(2.337, 7.102, 1.327)	54	(3.514, 7.102, 1.142)	70	(4.45, 7.102, 1.495)
72	(4.807, 7.102, 1.291)	145	(1.414, 7.102, 2.2)	56	(2.909, 7.102, 2.249)
74	(7.355, 2.944, 1.381)	78	(7.355, 2.991, 1.265)		

And then, we localize the camera set and applied our LS algorithm on 7 different positions. The ground truth positions are those cameras which are able to apply LS algorithm, and the computed positions and the ground truth positions measured by the laser meter are shown in Table:

Table 6: LS reconstruction results

	Ground truth location[m]	Computed location[m]	RMSE	error ratio
position 1	(5.336, 3.372, 1.670)	(5.253, 3.333, 1.665)	0.092	1.41%
position 2	(2.280, 3.396, 1.670)	(2.160, 3.566, 1.688)	0.208	4.72%
position 3	(2.346, 4.922, 1.670)	(2.282, 5.059, 1.671)	0.151	2.65%
position 4	(3.644, 4.201, 1.670)	(3.602, 4.139, 1.672)	0.075	1.29%
position 5	(2.601, 3.350, 1.670)	(2.563, 3.351, 1.670)	0.038	0.83%
position 6	(2.159, 2.842, 1.670)	(2.137, 2.876, 1.679)	0.041	1.05%
position 7	(1.592, 5.276, 1.670)	(1.617, 5.284, 1.681)	0.028	0.49%

Compared the ground truth locations with the computed locations, we can draw the conclusion that, in general, the LS method is able to reconstruct the location of camera inside a room within 3% error ratio and all the z locations are approximately the same as the original values. Then, we draw the movement of cameras in 2d coordinate frame from up to down and the movement route in 3d frame:

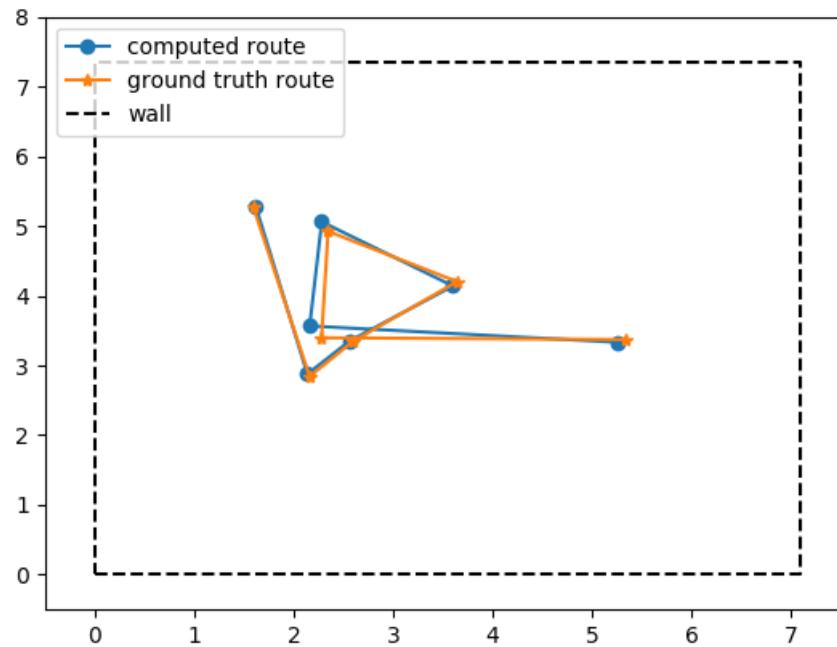


Figure 24: LS reconstruction result of camera movement in 2d

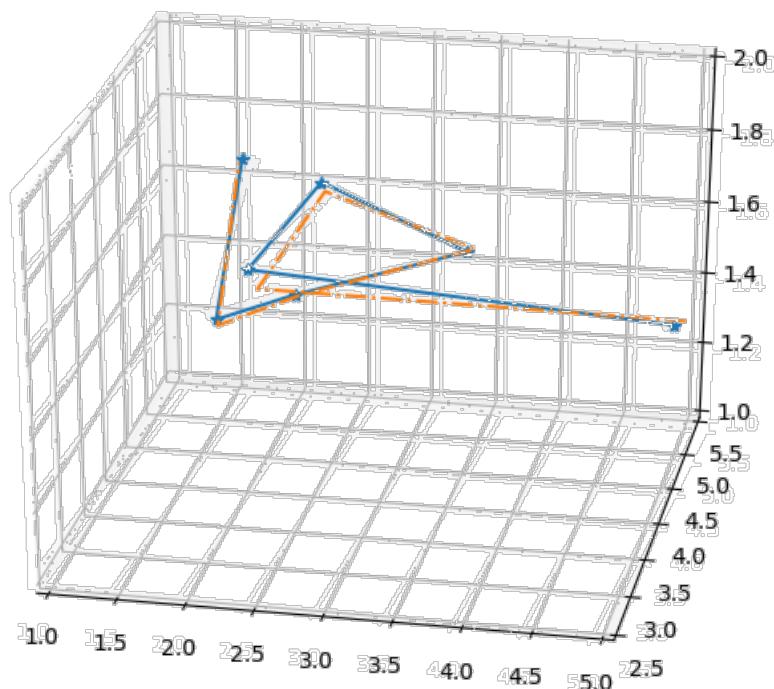


Figure 25: LS reconstruction result of camera movement in 3d

The same as what we do in Section 7.3, we apply the MDS and SRLS algorithms on all these 7 positions. In this case, we have the ground truth locations of three cameras, and we represent their average as the locations of the camera set inside the room. And also, because we change the locations of cameras, we need to modify the distances from Apriltags to cameras. We make an assumption that adding 0.04m to each distance data, which is the distance between camera lens to center of camera set, will improve the results accurately. The new ground truth locations of cameras are shown below:

Table 7: Locations of three camera and new ground truth location

	camera 139 location	camera 141 location	camera 145 location	new ground truth
position 1	(5.407,3.396,1.670)	(5.336,3.372,1.670)	(5.360,3.445,1.670)	(5.368,3.404,1.670)
position 2	(2.280,3.635,1.670)	(2.252,3.564,1.670)	(2.212,3.631,1.670)	(2.248,3.610,1.670)
position 3	(2.401,4.940,1.670)	(2.459,4.913,1.857)	(2.346,4.949,1.670)	(2.374,4.929,1.670)
position 4	(3.655,4.247,1.670)	(3.719,4.221,1.670)	(3.644,4.201,1.670)	(3.673,4.223,1.670)
position 5	(2.601,3.350,1.670)	(2.671,3.308,1.670)	(2.596,3.270,1.670)	(2.623,3.309,1.670)
position 6	(2.159,2.814,1.670)	(2.223,2.794,1.670)	(2.146,2.767,1.670)	(2.176,2.792,1.670)
position 7	(1.567,5.352,1.670)	(1.592,5.276,1.670)	(1.516,5.292,1.670)	(1.558,5.307,1.670)

Then, the reconstruction results using MDS and SRLS algorithms are shown below, and also, Figure 26 and 27 represent the RMSE and error ratio values of 7 positions with three reconstruction algorithms.

Table 8: MDS and SRLS reconstruction results

	#tags	MDS location[m]	RMSE	error ratio	SRLS location[m]	RMSE	error ratio
position 1	22	(5.380, 3.435, 1.771)	0.106	1.62%	(5.381, 3.439, 1.686)	0.041	0.62%
position 2	17	(2.249, 3.622, 1.536)	0.134	2.93%	(2.234, 3.623, 1.592)	0.080	1.75%
position 3	16	(2.459, 4.913, 1.857)	0.206	3.61%	(2.452, 4.922, 1.732)	0.100	1.75%
position 4	22	(3.664, 4.216, 1.688)	0.021	0.36%	(3.661, 4.226, 1.670)	0.012	0.21%
position 5	20	(2.638, 3.306, 1.761)	0.093	2.04%	(2.638, 3.315, 1.648)	0.075	1.65%
position 6	20	(2.229, 2.795, 1.778)	0.121	3.08%	(2.219, 2.796, 1.726)	0.071	1.81%
position 7	14	(1.655, 5.358, 0.992)	0.686	11.89%	(1.636, 5.375, 1.524)	0.179	3.09%

Table 9: LS, MDS and SRLS reconstruction errors

	LS RMSE	error ratio	MDS RMSE	error ratio	SRLS RMSE	error ratio
position 1	0.092	1.41%	0.106	1.62%	0.041	0.62%
position 2	0.208	4.72%	0.134	2.93%	0.080	1.75%
position 3	0.151	2.65%	0.206	3.61%	0.100	1.75%
position 4	0.075	1.29%	0.021	0.36%	0.012	0.21%
position 5	0.038	0.83%	0.093	2.04%	0.075	1.65%
position 6	0.041	1.05%	0.121	3.08%	0.071	1.81%
position 7	0.028	0.49%	0.686	11.89%	0.179	3.09%

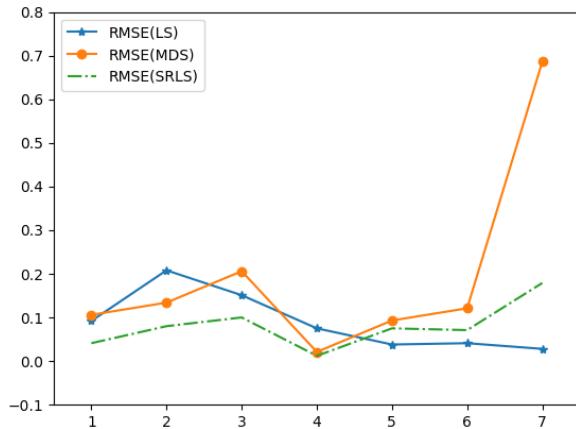


Figure 26: RSME of 7 positions

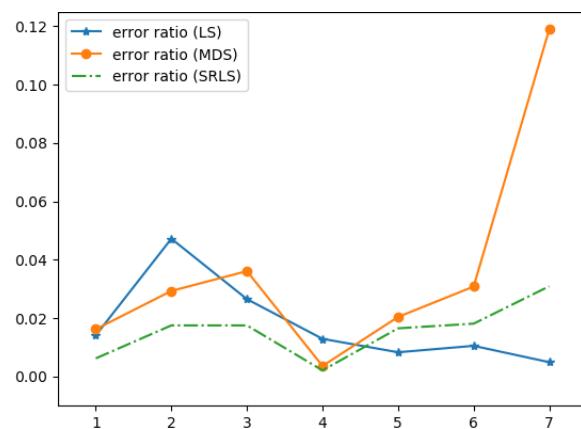


Figure 27: error ratio of 7 positions

Figures 28 and 29 shown below are the camera movement according to the MDS and SRLS reconstructed locations. The routes look similar and almost overlap, which means these two algorithms can reconstruct the camera's x and y locations basically. However, we cannot see too many differences from the images, and we need to analyze more based on the numerical results and 3d reconstruction.

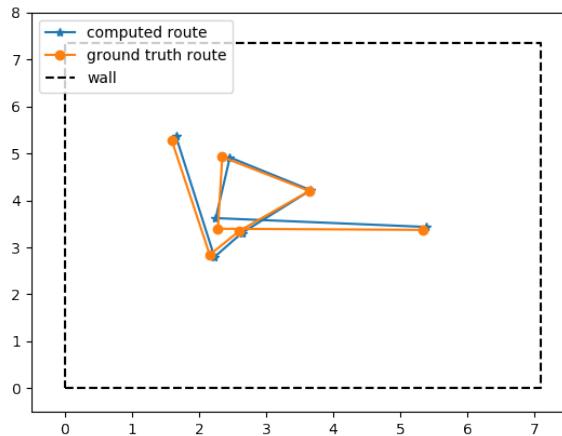


Figure 28: MDS reconstruction result

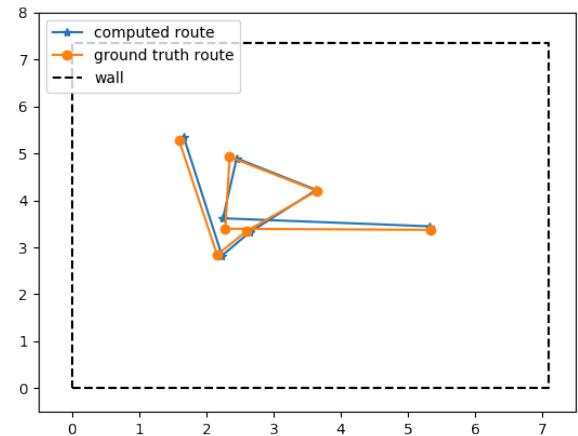


Figure 29: SRLS reconstruction result

Figure 30 and 31 are two sideviews of the 3d reconstruction. Generally, MDS reconstruction performs worse than SRLS reconstruction because the former algorithm has a big variance on z axis, which means MDS algorithms cannot reconstruct the height of the camera set correctly. As for SRLS algorithm, the heights of the camera set are reconstructed properly except for the last point with lots of errors.

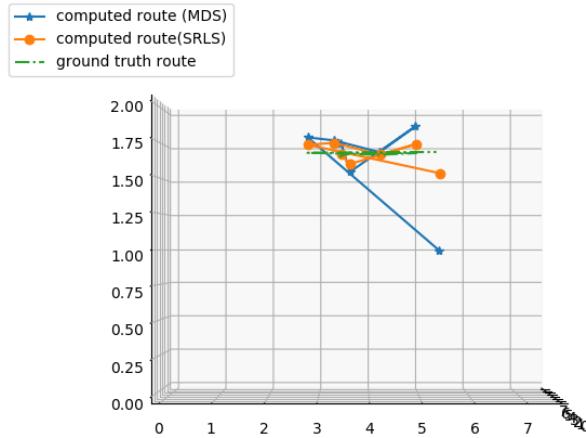


Figure 30: sideview of 3d reconstruction

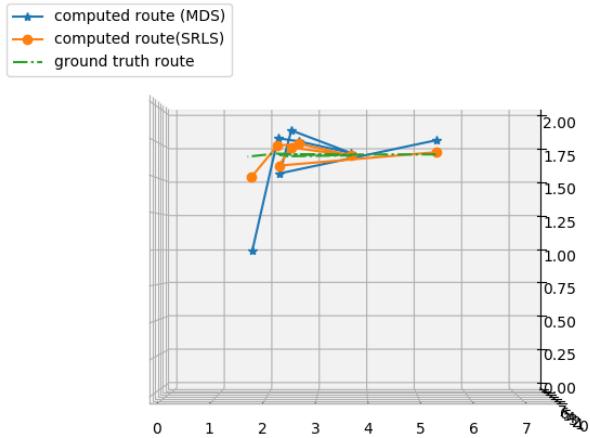


Figure 31: sideview of 3d reconstruction

According to the Table 8 and Table 9, Figure 30 and 31, the RMSE and error ratio values of MDS are all bigger than SRLS's, which means in this case, SRLS algorithms works more accurate and meaningful than MDS algorithm. And this is the same as what we expect in Section 6.3. And different from Section 7.3, the accuracy of SRLS algorithm is not worse than LS algorithm and sometimes even better than LS algorithm. And we notice when we increase the number of Apriltags on the walls, the performance of these algorithms will improve. Thus, if someone is going to reconstruct the distance data, it is better to use SRLS algorithm instead of MDS, and the increasing amount of distance information is able to improve the results.

8 Conclusion

In this project, the experimental framework for indoors visual localization has been successfully implemented and developed. An automated robot sporting three web cameras is placed inside a room. With different demands, the operator is able to control the robot under two operating modes, the first one is to control in real time while the other is to design the robot route in advance.

As for camera calibration, three web cameras have been calibrated convincingly, whose results will be used in further experiments. What's more, a reliable visual localization system has been developed and tested. Firstly, the visual localization system is tested in a small room, where some image markers, Apriltags, are pasted on part of the room's walls and ground, so that cameras are able to detect their relative locations inside a room. Afterward, the locations of camera are computed by the *Least Squares method*, and being compared with ground truth location measured by a *laser meter*, the computed location has an error tolerance within 2%, if we ignore the errors caused by laser meter.

Furthermore, this visual localization system is tested in a bigger classroom, BC129. The data reconstruction results show that when using *Least Squares* for reconstruction, the computed locations are similar to the ground truth locations within 3% error on 6 positions. As for two new reconstruction algorithms *MDS* and *SRLS*, the results using former algorithm are sometime meaningless and wrong, which should not been used anymore. On the contrary, *SRLS* algorithm works better and the error ratios are below 2% in most cases.

There are some possible improvements can be considered in the future. First of all, the LS algorithm looks convenient and powerful but actually not suitable in all cases. When the locations of Apriltags are in a same plane, it is nearly impossible to compute the 3d transfer matrix correctly, and also the location of camera. Even if the system is accurate, I have to make sure that not all the Apriltags are in a same plane.

Another possible improvement is that we can try to setup a new Apriltags system. Because everytime we do the experiment we need to put our Apriltags around the walls, which is inconvenient if the room is too large or not suitable to post paper. A possible solution is to create a big board with many Apriltags, just like the chess board we use for camera calibration. When we measure a new room, we can put the board inside the room and only need to measure one point's location in the global frame. And also, some small problems can also be considered, such as how to measure the locations of the cameras and Apriltags inside a room faster and more accurately, how to calibrate the camera in a more efficient way and what's more, how to localize inside a room without the help of Apriltags. They are all left as open questions that need to be solved in the future.

In general, the main goal of this visual localization system is achieved and the error ratio is small enough that the system is able to be considered as an accurate system. However, more works are required including modifying the algorithm for data reconstruction and measuring the locations of camera and Apriltags more accurately.

9 Learning Experience

This is the most significant project I have ever finished so far. Because this is the first research project that all materials are in English, which is not my mother tongue and brings some obstacles at the beginning. And also, this is the first research project that requires python, which I never used before. At the beginning of experiment, we were planning to improve and develop the experimental framework for EchoSLAM, which was last year semester project finished by Frederike. However, one day during testing the robot, we connected two wires by mistake and the robot got damaged. Thus, we changed our research topic more on visual localization setup. There are some kinds of inevitable problem happening during experiment and I learn changes run faster than plans so that we need to change our schedule according to some unexpected events.

Building a user-friendly human-machine interface is another thing I have learned. Just like Robin always tells me, a code-friendly interface will make the experiment testing and debugging much easier and faster. And also, some simple but sufficient comments can help others reading the codes and restart this experiment quickly and correctly. I started developing this habit at the middle of my experiment and I found it was very uncomfortable and improficient reading the codes I wrote before. And now I improve all the codes I write and all of those are uploaded online. I hope they can be further developed and extended by other users.

Well document organizing is also an important procedure for experiment. Due to the time consuming nature of experiments, there are tons of files and folders created by testing, so the awkward and messy file saving will increase the difficulties when searching for a specific document. Well organized documents can help us extract useful and expected features and information efficiently.

The most challenging and time consuming problem during my experiment is the version of software. The basic codes for controlling robot are from Frederike, which are written by Python 2. However, I am supposed to run the whole experiment under Python 3 environment so that lots of functions cannot work anymore and I have to rewrite them all. For example, Python 2 and Python 3 save byte-like data differently. In Python 2, bytes data is saved as a string while byte-like data is saved as bytes type in Python 3. The same as Python, the version of *OpenCV* is another trouble I meet. During the period I work on my project, the latest version of *OpenCV* is only supported in Python 3.5. However, the latest version of Python is Python 3.6, which doesn't support *OpenCV* and I spend plenty of time figuring out this problem.

Searching for solution online is an efficient method for working. However, there are many limitations and definitions should be noticed. I was solving a least squares problem before and I needed a function to compute the process efficiently. I found a solution but the results were terrible because the function I found was only used for non-negative inputs but I ignored this limitation, which also cost me plenty of time debugging. Finding solution online is a good method but we need to be carefully about its constraints and limitation.

In general, I acquire many useful technical skills from this project including learning how to use Python for achieving different requirements and setup a webcam and its related local network. Moreover, I learn some knowledge on indoor localization and computer vision, which are the areas I never touched before. Last but not the least, I develop many good studying and coding habits through this semester project that I will never forget.

References

- [1] Frederike Dümbgen. Routines to control a humanoid echolocator robot. <http://lcav.github.io/AcousticRobot/pages/develop.html>, 2016.
- [2] Frederike Dümbgen. Experimental setup of sound emitting and processing robot for acoustic-based slam applications. <https://infoscience.epfl.ch/record/215272?ln=en>.
- [3] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
- [4] OpenCV. Camera calibration and 3d reconstruction. http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html, 2016.
- [5] chief2787. Calibration example. <http://www.dreamincode.net/forums/topic/318080-opencv-stereo-camera-calibration-run-on-linux/>, 2013.
- [6] Davide Macagnano and Giuseppe Thadeu Freitas De Abreu. Algebraic approach for robust localization with heterogeneous information. *IEEE Transactions on Wireless Communications*, 12(10):5334–5345, 2013.
- [7] M. Hazas and A. Hopper. Broadband ultrasonic location systems for improved indoor positioning. *IEEE Transactions on Mobile Computing*, 5(5):536–547, 2006.
- [8] I. Dokmanic, R. Parhizkar, A. Walther, Y. M. Lu, and M. Vetterli. Acoustic echoes reveal room shape. *Proceedings of the National Academy of Sciences*, 110(30):12186–12191, 2013.
- [9] Yu Kawahama and Ryo Katsuma. High-accuracy localization via measurements of rssis and led light angles for low-cost wmsns. *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, 2017.
- [10] Jason Zhi Liang, Nicholas Corso, Eric Turner, and Avideh Zakhor. Image based localization in indoor environments. *2013 Fourth International Conference on Computing for Geospatial Research and Application*, 2013.
- [11] Marc Pollefeys, Olivier Saurer, Friedrich Fraundorfer. Visual localization using global visual features and vanishing points. *CLEF (Notebook Papers/LABS/Workshops), volume 1176 of CEUR Workshop Proceedings, CEUR-WS.org*, 2010.
- [12] Claudio Piciarelli. Visual indoor localization in known environments. *IEEE Signal Processing Letters*, 23(10):1330–1334, 2016.
- [13] Branislav Micusik and Horst Wildenauer. Descriptor free visual indoor localization with line segments. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [14] Huijuan Zhang, Chengning Zhang, Wei Yang, and Chin-Yin Chen. Localization and navigation using qr code for mobile robot in indoor environment. *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015.
- [15] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. *2011 IEEE International Conference on Robotics and Automation*, 2011.
- [16] John Wang and Edwin Olson. Apriltag 2: Efficient and robust fiducial detection. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.
- [17] Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli. Euclidean distance matrices: Essential theory, algorithms, and applications. *IEEE Signal Processing Magazine*, 32(6):12–30, 2015.

- [18] J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964.
- [19] A. Beck, P. Stoica, and Jian Li. Exact and approximate solutions of source localization problems. *IEEE Transactions on Signal Processing*, 56(5):1770–1778, 2008.