Abstract

In this paper, I will apply a recognition function on an ROS Corobot system to detect whether Corobot will be able to pass through door by using several 2D image processing techniques. There are few key-points which are really important for me to solve this problem — edge extraction, light and effective elements of environment. The edge detection and light area will help me to locate the location of door and its opening area. The effective elements are used in experiments which will test my function and help me to proof that my function is reliable. My function is successful at locating a stable area which provides a reliable result for recognition in a closed environment which is not effected by any unexpected elements. Based on the performance of experiments of my function on different extreme environments, my function works well on the area which is nearby door even though the wall is glasses or the door sitting deeper in wall. Furthermore, according to my experimentation it is suitable for real-time applications in moving robots.

1. Introduction

In the modern world, more and more people come to be interested in AI and robot system, because of technology increasing and the great future of robot. An entire Robot consists of different parts of functionality. The aim of this paper is trying to build an app for a default robot system to recognize the opening door for the robot and make sure whether the robot will be able to pass through the door or not.

For the door detection, this relates to the computer vision and intelligent system. Normally, what we do for this job is finding the edge of the door and apply them to build an entire shape. Based on the shape recognition function, the system will tell if the shape is a door or not. However, there are a lot of elements will effect the result of the whole process, for example the light, the angle of the camera, the items on the wall and the color comparison between the door and the wall. So, it’s a challenging job which is worthy for me to work.

Recently, researchers try to use the different way to solve this problem, like fuzzy logic[12] and monocular door[13]. However, most of them do not seem work well when the door is opening and the more efficient ones (up to 90% of recognized doors) are typically based on a large range finder.

To approach the goal of my project, I am trying to apply the SIFT learning method and some image processing technology to increase the adaptability and accuracy of my function. So this is the way I used to recognize the door. However, my goal is detecting whether the robot will be able to pass the door or not. So another important element is door frame. If I can recognize the door and door frame, then I can calculate the width of the opening size. However, the door frame won’t be a problem if I can detect the door. So how to track the door will lead to the biggest problem of my project.

2. Related Work

Before I start my real work of my project, I would like to talk about the recent work in the same area which also formed the foundations of my work to find the right direction to solve my problem.

Since our goal is to recognize doors in an office environment, we first have to determine related geometrical features. Indeed, doors can be represented by a rectangular shape where width is given by the distance between its two doorposts.[13] However, this way is not so perfect. Sometimes, the door is too much deep in the wall. In this case, the frame of door will also be detected by 2D image technique and these noise edge will affect computer. In this case, it’s hard to apply simply edge detection to recognize the door.

To solve this problem, F. Mahmood and F. Kuwar apply an edge feature detection function and a line label function[5]. They chose line detection algorithm developed by Kovesi to group the intensity edges into straight line segments not hough transform which they believe that HT is quite sensitive to the window size. After this, they label edges by their features which will ignore the edges of frame of door.

However, Rafael and Miguel[12] provide a new way to solve this problem by edge. Basically, they apply edge detection function first and save the information of line. After this, they use a fuzzy method to recognize the frame of door and then to detect door. The fuzzy logic function is really clever to solve their problem. However, their function is based on a logic function and the accuracy of experiments are not good enough. Christopher Juennemann, Anthony Corbin, Jian Li[9] use almost same way to solve the problem. However, their function seems more functional by the display with result. But I find the amount of their experiments is too small to tell a strong result. So this method is not sufficient to solve this problem.

Then I come back to the door detection by matching function part. Xiaodong and Yingli[16] provide a wonderful report about how to use the feature of door and corner of edge to detect the door. In their method, they apply a per-process image function to smooth image and turn the colorful image
into gray. After they apply the corner extraction function and edge extraction function to detect the edge and corner. Finally, they apply the information from database to compare with current information of edge and corner. The performance of their function is good. However, I think their function only works for closed door. It’s because this paper didn’t cover the result of opening door on their experiments. But it gives me a good idea about using matching function to solve the problem.

However, most of these works are based on closing door which means they can model the door in most situation. It’s because they ignore the effective elements which come from the view of opening part and the frame of door. Normally, we don’t know what will be displayed when door is opening and we believe most of them are different between each other. When the door is opening, it is impossible to simply apply these functions to recognize the door. For example, if door is opening, the way which F.Mahmood and F.Kuwar work would not be able to work. It’s because the most important part of their function is edge label part, and the edge feature which they apply to solve this problem will not keep the same equation. Furthermore, when the door is opening, the view inside door is unpredictable and we know the Canny edge detection function is a sensitive function which means the binary edge image will be a mass of line.

Since the aim of my function is recognize whether the Corobot will be able to pass through door, I also need to locate the edge of frame of current door. Even though these papers won’t really solve my problem, it still gives me a direction to lead me to the target.

3. BACKGROUND

Computer vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos[2][8][14]. From the perspective of engineering, it seeks to automate tasks that the human visual system can do. Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, for example, edge detection, object recognition and so on. Given the goal of computer vision, it’s best if they run in real time.

So in my project, though I discuss the robot, the most important part is camera. It’s a typical computer vision problem. So what we need to do is make sure that my function works in real-time which means the robot will detect the door immediately and because of the frame of the door, it won’t be difficult for me to calculate the width of the opening part of the door. In this case, I can apply this width with the size of the robot, then the system will tell me the result. For this part, I will explain how the methods work, why I use these methods in my function and why they fit with my function.

3.1 Canny Edge Detection

In the computer vision area, sometimes we need to abstract the edge from the image. The edge from a binary image. The principle of this method is trying to find the competitive gradient of the pixel in the binary image. It’s easy to see the edge will come out between two different color space. And in the binary image, the different color space will hold two different gray-scale which means if we can select a wonderful threshold, we will pick up most of the edges from the image. As I mentioned before, the light is a important affective element of my function. So how to select a good threshold will also affect the future work. Canny Edge Detection provides a double threshold method which is a good method to select the threshold. So that’s the most important reason why I ignore other kinds of edge detection methods to choose the canny one.

3.2 Hough Transform

Hough transform was invented by Richard Duda and Peter Hart in 1972 which is a feature extraction technique used in image analysis, computer vision, and digital image processing[7]. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure which you can assume it’s a kind of voting method.

To understand the Hough Transform, you should know that all of points in a 2-D area, we can represent them by Hesse normal form((x,y) to (r, θ)) where r is the distance from the origin to the closest point on the straight line, and θ is the angle between the x axis and the line connecting the origin with that closest point. In this case, the different lines will also have the different angle with the different points on them. At the beginning, Hough Transform is used to solve the problem of collinear points. However, we can also use it to find the straight lines if we can have the set of collinear points. And it’s the reason why I choose it to pick up the straight lines after the Canny Edge Detection. It will ignore the noise of the edge image.

3.3 Shape Segmentation

Identifying and separating objects within images (figure-ground segmentation) represents a significant challenge due to high object and background variability[3]. It will help us to find the shape which we want with a shape detection function. For a closing shape in the binary image, if we scale the entire image with a boolean number and each time when I pass the edge, I will change the value of the boolean number and label that part starting from 1. Then I can label all the shapes from the image with the different counting number. Since we know most of the door is parallelogram, all the detection function only need to calculate whether the shape is a parallelogram or not.

3.4 Feature Detection

In computer vision and image processing the concept of feature detection refers to methods that aim at computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not. The resulting features will be subsets of the image domain, often in the form of isolated points, continuous curves or connected regions. Normally, we will be able to build a feature database for the interesting features of template image. Then we will apply the different image to feature detection function. At the end, we will compare feature of current image with feature of database to recognize the matching part. For the different situation,
we will apply different kinds of matching method. Some of them works better on matching object than the others. Some of them works faster than the others. For my project, I will test SURF (Speeded up robust features) [6] which is a fast method will keep my function running in real time and SIFT (Scale-invariant feature transform) [10] which is good at matching object. Speeded up robust features (SURF) is a method which apply the Hessian matrix to locate feature points. By using these feature points, matching function will be able to match the points which belong to different images have the same feature information. In this way, the interesting point of SURF won’t be change by apply a pyramid method which will speed SURF up. Scale-invariant feature transform (SIFT) is an algorithm which applies the scale feature of image to locate the feature point. And it has a special way to remove the noise of interesting point which will be able to provide a more accuracy result than other matching method. I will explain them with more details in architecture part.

4. ARCHITECTURE

To approach my goal, I divide the entire function into few parts — getting image from ROS system, edge extraction and recognition part. Figure 1 shows the entire process of my function.

4.1 Basic Image Process

At the beginning of my project, I applied Canny edge detection, hough transformation and shape segmentation on the image. As I mentioned before, after these processes, the image will be segmented by the shape of edge. On the plan, if the shape is closed which means the edges of the shape is perfect, I can simply recognize door by the size of the shape and feature of the shape, for example, parallelogram. Here is the simply image of this part. There is a problem on the image. The edge of the image comes from edge detection function which will miss some parts of the edge. In this case, the edge is not likely to form a closed body which means the shape segmentation function won’t works well. So I can apply a fuzzy line extension function to create shape.

But there is another problem. After the line extension function, there are a lot of different kinds of shape in the image. Considering about the way how I get edges, most of these shapes are parallelograms which means I can’t simple apply the shape recognition function to locate the door. To approach the aim of my project, I must find the right one. So, I need a matching function to catch up some features from the door. In this case, the most frequency matching area will be the location of the door on the image.

4.2 Shape Matching

You can see that the door is hard to be recognized from these images. It’s because the edge of door is not perfect. So I come up with a new idea. Maybe I can apply a shape matching function to recognize the handle of the door. In this case, I can locate the location of door on the image. And another advantage is that if I can apply the shape matching function is ignoring noise object on the door. We know that sometimes, there are a lot of pictures or letters on the door. And it will be catch up by the edge detection part. If I can
locate the location of handle, then I can say the larger shape which includes the handle is door. To approach the goal, I try the SURF and SIFT to match handle of door.

4.2.1 SURF

Speeded up robust features (SURF)[6] is a patented local feature detector and descriptor. At the beginning of the SURF, we need apply a Gaussian smoothing which will be able to remove the noise of the image and be ready for the next few steps[6]. Then what we need to do is catching up the interesting point. In this part, SURF uses a blob detector based on the Hessian matrix to find points of interest. The determinant of the Hessian matrix is used as a measure of local change around the point and points are chosen where this determinant is maximal. In contrast to the Hessian-Laplacian detector by Mikolajczyk and Schmid[11], SURF also uses the determinant of the Hessian for selecting the scale, as is also done by Lindeberg. Given a point $p=(x,y)$ in an image $I$, the Hessian matrix $H(p,\sigma)$ at point $p$ and scale $\sigma$, is:

$$H(p,\sigma) = \begin{pmatrix}
L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\
L_{yx}(p,\sigma) & L_{yy}(p,\sigma)
\end{pmatrix}$$

(1)

According to the different size of blob box, the speed of SURF and accuracy of SURF will be different. For now, we know how to catch the interesting point. Then what we need to do is speed up SURF and represent the interesting point.

For this part, we need to resize the image by using pyramid method. The pyramid method is a static way to resize the image and won’t change the scale comparison between the original image and the image after the method. In this case, we can process a $1/4$ or $1/8$ size of original image and get the same result.

After each of these, we need a descriptor to describe the interesting point and apply it to finish the matching part. The dimensionality of the descriptor has a direct impact on both its computational complexity and point-matching robustness/accuracy. A short descriptor may be more robust against appearance variations, but may not offer sufficient discrimination and thus give too many false positives. So in the surf, we apply a circle region of the interesting point to consist a descriptor.

The final step of SURF is matching part. What we need to do is match the different descriptors from different images. If they match each other, then we can say the matching point between the different image represent the same thing.

Here is the experiment of SURF. First, I take an image of handle of door. Since my project is focused on the door in our building. So my matching database only includes one image. Then I set up the robot and put it in the front of door. When the robot is closer to the door, the matching works better. However, when the robot stand in the middle of the road which should be a normal distance, it cannot recognize the handle.

I think it’s because the handle is too small in the image and the SURF is not too effective. For 100 images of half meter from door testing, almost 80 present of result won’t match handle. So this method is inadequate for my goal. However, I still believe that the matching function will help me to solve the problem. So I will try a new method.

Here is the bad result of SURF which is also the reason why I abandoned this method.

4.2.2 SIFT

Scale-invariant feature transform (SIFT)[10] is an algorithm in computer vision to detect and describe local features in images. The reason why I choose it is because the interesting point selection part of SIFT is more effective than SURF which I think will give me a better result.

First, we need the Gaussian filter to convolute image to smooth image and remove noise. Then the difference of successive Gaussian-blurred images is taken. Keypoints are then taken as maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales. Specifically, a DoG image $D(x,y,\sigma)$ is given by :

$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,kj\sigma),$$

For now, we have a lot of interesting points. However, some of them are useless which means we need to remove them. In this part, we apply the Taylor expansion[15] and Hessian matrix to remove the low-contrast keypoints and the keypoints that have poorly determined locations but have high edge response.

Then we need to create the descriptor by orientation assignment. Since each small block of interesting point can be represented by a vector by :
\[ m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \]

\[ \theta(x, y) = \text{atan2} \left( L(x, y+1) - L(x, y-1), L(x+1, y) - L(x-1, y) \right) \]

we can apply the matching function by match the vector box.

At the beginning, I took a handle image and applied SIFT on this image to build my feature database. After that, I set up the robot on the different distance and angle of the door to do the matching part. As the result return, it has some problems. Some of the results are not as good as expected.

You can see that SIFT can mostly recognize the correct feature. However, since the handle is too small in the image, it also be effected by noise which will return a bad matching.

I try to set up a summary region as a voting method which I believe can solve this problem. I think if I can set up a circle which cover the interesting point and vote this part after each matching, the most frequency voting area will be the interesting point of handle. However, since the door is too small in the image, there are not too many matching points on the handle area. Considering the result of SIFT, this method is not suitable for my project.

### 4.3 Light Process

After the testing of SURF and SIFT, I came back to the image process method. During many experiments, I found the light inside the room will be turned on automatically when the door is open. Since the aim of my project focus on our building, considering about the real-world elements, I think the light will be the key to solve my problem. So I started to try to apply the light to solve my problem.

When considering about the height of robot, I think I should extend the line of edge first. After the hough transformation, I already picked up "edges". After the analysis these edges, I find some of them are the edges of objects in the image and some of them are the reflection of light area. And the light area is not clear which give me a new idea. If I can fill the light area correctly, then it will provide a good comparison between the light area and the door.

We know that a lot of image processing function is based on the binary image using special the edge detection function. The light will affect the result of edge detection. If the threshold is too small, we won’t miss any edge but have too much noise and most of them is impossible to remove. If the threshold is too big, we will miss some edges. According to the light, it’s hard for me to set up a wonderful threshold for every image. And if the light is too strong, there are some light reflection area on the door.

To approach the goal, I need to remove the noise and select the light area correctly. Coming back to the Canny edge detection, we know that Canny uses an auto-double-threshold to process the image. Considering about the definition of gray image, the edge image after Canny will cover the light pixel and edges. Hough transformation will determine the most frequency line and won’t change any pixel value of Canny image. So my aim image will be the image after HT. Canny edge detection applies the double threshold filter to remove the noise and remove the dark area. So if the pixel of my target image is not zero, then I can say this pixel belong to the light section. And it should be because Canny function already remove the darkest part. However, these pixels may invisible in the image, it’s because the value of pixel is too small. Since the ground is flat, the image from robot is also flat. So I can extend the light pixel as a line if 1/3 part of the line is consisting by light pixel.

After this, the light line will consist a light area which is the reflection of light. Figure 8 is the result from light process. You can see that we can simply determine which is the opening area of door. To compare the light area and the darkness area which is the door. I can find the width...
of opening size of door. Based on this comparing this width and the width of robot, I will be able to tell whether the robot will pass the current door or not.

4.4 Communication Between ROS And My Function

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license.

The Corobot is built on the ROS system, so I must set up a communication function between my door recognition function and Corobot.

There are two parts are significant for application. What I need to do is set up a subscribe library and apply the CV bridge library to send the image from camera to my function. The subscribe library will allow me to build a node which has a call back thread[1]. When the function finds an input image which comes from camera, it will automatically call the call back function which I can relate with my recognition function. The CV bridge library will allow me to run my function under ROS system and send image from camera to my function. It’s because my function is built in openCV. I will discuss more details on the implementation part.

5. IMPLEMENTATION

There are three parts of my function, door recognition, communication and support function.

For the door recognition part, it includes all the methods which I discussed before. Here is the construction my function.

Recognition function :
1. Canny edge detection
   Step 1: Apply a Gaussian blur
   Step 2: Find edge gradient strength and direction
   Step 3: Trace along the edges
   Step 4: Suppress non-maximum edges
2. Hough transformation :
   Step 1. Corner or edge detection.
   Step 2. Rho range and Theta range creation
   Step 3. Hough accumulator
   Step 4. Voting in the accumulator
3. Blind line extension :
   Step 1. loop the line and find the light pixel.
   Step 2. count the pixel numbers and mark the line
   Step 3. remove the noise line and build the light area
4. Recognition

Communication function :
1. image from camera
2. call back thread with the input image and recognition function
3. result summary and return the final result

Here is the image of whole process of my function.

5.1 Line Extraction

Line extraction method involves using the openCV in-built library that allows us to load and modify the image as a matrix. First, the function will change the color image into gray, then apply the Canny edge detection and Hough line function to catch up the edge of the image. Since the in-built Hough line function of openCV will provide the line as a set of points. The function also needs to draw the line in a new image which has the same size as the input one. In this case, we will be able to get edges with the same image size and same location which is important for me to solve the problem.

Line extraction function does take it upon itself to provide some functionality beyond that of a Line extraction. For each image, the function will recognize if the image is flat first. It’s because my function is based on the flat image and sometimes computer will rotate the camera images with 90 degree automatically. For instance if the user types 1 when Line extraction function find the image is not flat, the image will be rotated by 90 degree.

It was mentioned that the line extraction will catch up all kinds of lines from the image. In my project, it’s not important whether line extraction covers some noises. The most important part is that it won’t miss any edges.

5.2 Light Area And Comparison

Light area part is using the openCV and C++ in-built library that allows a function to modify the image as a matrix and collect data by a stack. The stack is necessary for my function which will save the location of blind line and apply them into a light area. The pop function will also increase the speed of my function which will decrease the complexity of my function. In this case, I can keep my function working in real-time.

I use x axis to represent the location and index of light line. The function will save these information into the stack and skip light line neighborhood of last light line.

If light line is smaller than 5 which means the door is
closed. It’s because function will at least catch up 4 edges which are the edges of door and edges of frame of door. Otherwise, comparison part will compare light area and darkness area which is consist of nearly light lines to return the result.

However, there are a lot of different kinds of elements will affect result, such as shaking, moving and human being. Since function will apply image from camera, I set 10HZ to catch up image for the camera. And after 100 times which means 10s, the summary of the result will be more stable.

5.3 Communication

The communication portion of the problem is done by using the subscriber library and CV bridge library which are able to set images which come from camera up as a thread of input and convert them from ROS into openCV format. Since it’s a subscriber function, the recognition function will run immediately when the input is coming.

Due to the way how the ROS system works, I need to set up my entire function as a node first. A node is a process that performs computation. Nodes are combined into a graph and communicate with one another using streaming topics, RPC services, and the Parameter Server. These nodes are meant to operate at a fine-grained scale; a robot control system will usually comprise many nodes. For example, one node controls a laser range-finder, one Node controls the robot’s wheel motors, one node performs localization, one node performs path planning, one node provides a graphical view of the system, and so on.

After this, I will be able to run my function in ROS system. Another problem is data type. ROS save image as a string message which is unable to run in my function. So I need the CV bridge library to send image to my function and I also need a transform function to convert them into matrix. For now, I am ready to test my function.

6. EXPERIMENTS

Considering about the different kinds of door in the building and a few of different situation of environment. I decide to design my experiment as the follow way.

Figure 10 shows the basic test of setup.

![Figure 10: Test Design](image)

Since Corobot will be able to locate the approximate location of door, my experiment will focus on these areas which means they have only one door in the image mostly. Let’s assume that the width of corridor is d and I set up D1, D2, D3 as 1/4d, 1/2d and d. For these three distances, I draw a circle and pick up view points on these circle by 30 degree, 60 degree, 90 degree, 120 degree and 150 degree. The reason why I ignore the degree between 0 to 30 and 150 to 180 is because these angle is too small even for human eyes to recognize the door. The wall or rather, the frame of door will cover these degrees which is unable to extract edge of door. For each point, I will run my function 100 times and take the most common result. The highest frequency return will be the result of one time and then summary 20 times of result to provide a more stable value of current point to recognize the opening door to pass. The reason why I chose 100 * 20 to provide final result not 2000 is because I can modify the angle of camera during those 20 times. For the real world, Corobot will not scan door at same view point with same angle. It will move its head or stand up with different position of head. That’s the reason why I need to divide 2000 running into 100*20 which will allow me to imitate this shaking or position of head.

Furthermore, there are three basic states of door— door is closed, door is open but the robot is unable to pass and door is open and the robot is able to pass. Normally, we can simply divide situation of door into two— door is unable to pass and door is able to pass. However, the challenges between door is closed and door is open but the robot is unable to pass are totally different. So I need to know what the performance of my function during these two situations.

Finally, I will test all of these experiments based on three environments. The first environment is door is closed, door is open but the robot is unable to pass and door is open and the robot is able to pass. Normally, we can simply divide situation of door into two— door is unable to pass and door is able to pass. However, the challenges between door is closed and door is open but the robot is unable to pass are totally different. So I need to know what the performance of my function during these two situations.

To evaluate the influence of nearby humans, I will choose the stable area of view point which comes from result of experiment above and do the same experiment with human moving and standing with different distance between human and Corobot. In this case, I will be able to know which distance will be a secure distance for human which will allow Corobot works and ignore noise.

6.1 Evaluation

Based on my experiment, I will evaluate my function for different Dashboard. First, I will calculate the accuracy for each view point for each situation for each environment. For these data, I will assume that if the accuracy is more than 90 percent which means my function is reliable for the current view point, if the accuracy is between 70 percent and 80 percent which my function works OK if the accuracy is less than 70 percent which my function is useless. Normally, I
will keep the area which accuracy is larger than 80 percent or 90 percent, it depends on whether the value of accuracy is decreased by special experiments. However, I decide to extend my limit, it's because angle of camera will affect the result. I test my function by moving camera angle 20 times. In this case, if the accuracy is more than 70 percent which means the most angle of current viewpoint works well which is good enough to say my function works well. For the useless area, I will re-mark them and analyze the reason why it doesn’t work. We all know that we can recognize door opening only when we can see the door and its open area. However, some viewpoints in my experiments are unable to see the opening area. If this is the reason why my function doesn’t work for the current viewpoint, I will skip this mistake. It’s because this problem is unsolvable unless Corobot will be able to look through wall.

The final evaluation result will come from the data which is processed by above steps. I will summarize all these dates and pick up the most stable viewpoints or view area. If I will be able to get more or equal to 5 stable view point, I would like to say I solve the problem of my project. The reason why I choose 5 as my evaluation limit, it’s because 4 of my view points are useless sometimes. For example, the V1D1 and V5D1 will be useless if door is opening no matter the width of opening is enough to pass or not. V1D3 and V5D3 is too far away from door, so it’s too likely it will be affected by other doors, poster and corridor. These elements are unexpected, so the result of these two viewpoints may not be beautiful. However, if I can get 5 stable view point to create a view area. I can say that my function is reliable for this area. Corobot know the fuzzy locations of different door on different floors which means it will be able to stand close to door. Compared to my stable area, it's good enough for Corobot to recognize whether it will be able to pass door or not.

6.2 Data And Result

For this section, I will explain my test with more details and describe and analysis data of each test. Also, I will share my ideas of some special tests during the bigger sub-test. After the analysis part, I will evaluate each test follow the rules of evaluation. At the end of this part, I will provide a final result which comes from the summary of all of these tests to make my result more reliable.

6.2.1 Closed Door

Figure 11 is the result of my test on a regular closed door with my function.

![Figure 11: Closed Door Test Result](image)

<table>
<thead>
<tr>
<th>D/V</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>20%</td>
<td>0%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>D2</td>
<td>18.51%</td>
<td>100%</td>
<td>90.90%</td>
<td>100%</td>
<td>19.35%</td>
</tr>
<tr>
<td>D3</td>
<td>12%</td>
<td>100%/?</td>
<td>100%</td>
<td>100%</td>
<td>16%</td>
</tr>
</tbody>
</table>

You can see that the accuracy of V1D1 is quite poor. It's because the angle is too small and there are a lot of noise which is be visible by Corobot and apply these information to consist light area.

Here is the example of V1D1.

You can see that the QR code picture and corridor affect the recognition function. Since QR is too dark and the far part of corridor is too light, function will compare with these two part as light part and darkness part. However, it won't happen on the other side. As the table displayed, the V5D1 works well. It's because there is no QR on the other side. I also test the same angle with same view point without QR picture. It provides almost same result as V5D1. For the V1D2 and V5D2, their accuracy is exceptionally poor. The reason is because Corobot is too far away from door which will have too much angle will be effected by corridor. For these part, QR doesn’t affect anything. It’s because Corobot is far away from the picture and even thought HT is a sensitive function, it will be unable to get information from this picture. However, it doesn’t mean my function is useless on V1D2 and V5D2. During my test, I find when Corobot face or almost face to door which means the corridor won't stay too many part in camera, the result is good. In fact, there is a database in Corobot which will allow it to locate doors on each floor. Though the accuracy is not 100 percent, it’s still very well. So I would like to say these two view point works well even though most of angle of current view point doesn’t work.

6.2.2 Door Which Is Able to Pass

Figure 14 shows the table of my test on regular door which is able to pass.

Most of time, function works well. In the figure 15, you can see there is great comparison between light area and dark area.

However, there are three points which I need to explain, it’s because they are different from above explanation. First, the result of V1D1 is 0 percent. It’s because camera won’t be able to see the opening area of door. Figure 17 and 18 are the examples for this situation.

You can see that opening area is covered by wall no matter how I change angle of camera. So I mark this part as 0 percent even though there are few of result returns correctly. Since we don’t know how door will open, the same situation will happen on V5D1. For the V5D3, I mark it as 90.90 percent with R. The R means there is not really 100 percent but it may higher than 90.90 percent. It’s because this result will be effected by corridor for few angles which means it works well most of time. During the basic test, the result is static which means always return the right answer with right solution. In this case, the result may be higher than now. At the end, I will analysis why there are few points displayed a different result from above test, special V5D2. According to my analysis, I find light area will be created by opening area. Since light of room will be turned on automatically, opening area is much lighter than other part. Considering the distance between Corobot and door, corridor and other elements won’t affect result too much. That’s why the results of these points are different from above one.

### 6.2.3 Door Which Is Unable To Pass

In fact, the result of this part is almost as same as opening one. The only problem is V1D1 or V5D1 and V1D2 or V5D2. For the D1 part, Corobot is unable to find the opening area even though we know that it’s unable to pass for Corobot. So the result is right, but I can’t be sure it will be right all the time. It’s because when the size of Corobot changes or the size of door changes, it will be an error. For D2 part, it shares almost same situation with D1. For the opening door test, we know that D2 part won’t be effected by wall. It’s because Corobot will be able to catch up opening area of door. For this test, sometimes Corobot will not be able to catch up opening area for some opening angles of door or some camera angles. However, I won’t agree with that D2 provides the same result with D1. It’s because most of test of D2 of this experiment work well which means the right result comes from right solution. So I don’t think it will be
effected too much even size of Corobot will change or size of door will change.

6.2.4 Door Set Deeper In Wall

For this test, there is one more thing I need to talk about. Since door is deeper than normal one in wall, D2 part will be effected by wall. Figure 19 and 20 are the examples.

As the image displayed, the opening area is covered by wall even on the V1D2 location. In fact, this is an open door which is good enough for Corobot to be passed. The reason why I display this situation is I want to discuss the limit of influence of wall when door is much deeper in wall than normal one. In fact, this is the deepest door I can find in the building. I would like to say maybe this is the biggest influence of wall. And it will help me to get the final result of accuracy of my function. I will decrease the accuracy of these view points on my final table. Fortunately, there are only few of rooms have this kind of door which means it won’t affect too much.

6.2.5 Presence Of Glasses Wall

There are some rooms in our building which have glass walls. In fact, there are two different kinds of glass room—the frame of door is really thin and the frame of door is thick. I will focus on discussing the first one, it’s because the result of second is as same as other experiments.

Figure 21 is the result of rooms with glass walls.

The capital E means error which means my function can not recognize anything even some of the results are correct. It’s because the glass wall represent like a light area. In this case, my function is unable to catch the opening area. However, you can see that only D1 position still works well. It’s because my function relies on light reflection area on camera and the glasses wall will provide too much unexpected light area. When Corobot is closed to door in D1 part, it will ignore this noise. Otherwise, the result will be really bad even though result is correct sometimes. I mark three capital R on the table, it’s because these viewpoints will work for some angles of camera.

Figure 23 and 24 are the examples from these three view points.
bad. However, result reverse on V2D2 and V4D2. There are few of angles of these two viewpoints work well. It’s because the glass won’t cover too much on camera for these three point. Considering for this situation, I will decrease the value of these three point on the final table with different value.

### 6.2.6 Human Test During All Experiments Above

To test, I ask a volunteer to act as a human noise source for all of above experiments. I ask him to move in front of camera, stand with different distance in front of camera and try him best to cover the door.

By the result, if human stay really closed to camera, there is no useful information we can apply. If human stay away from camera between half meter and zero, then it will decrease the accuracy of result. The further human stay from camera, the better result we get. The closer Corobot stay to door, the bigger influence human make. It’s hard to say what’s the exactly influence value of human being. So I just discuss this point and won’t represent this result on my final table. However, it’s important for note that human interference is an important issue.

### 6.3 Conclusion

Here is the final result of my function which I summary it from all experiments.

<table>
<thead>
<tr>
<th>D/V</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>13%</td>
<td>91.11%</td>
<td>97.14%</td>
<td>98.57%</td>
<td>20%</td>
</tr>
<tr>
<td>D2</td>
<td>18.5%</td>
<td>87.09%</td>
<td>98.7%</td>
<td>91.80%</td>
<td>19.35%</td>
</tr>
<tr>
<td>D3</td>
<td>8.57%</td>
<td>47.14%</td>
<td>71.42%</td>
<td>60%</td>
<td>9.16%</td>
</tr>
</tbody>
</table>

For the opening door, unable to pass door and closed door, I decrease the value with useless angle for different experiment and average the values of them. For example, if 5 of 27 angle works, then I set the current point with 5/27 and summarize all the values of same point together and average them. For other special consideration such as glass walls and deeper wall, I also decrease the value with useless angle. Furthermore, I multiple value with coefficient which comes from the value of special door of all doors. For example, there is 2 doors with glass walls in the entire test. The result will be calculated by 2 of amount of door. Then I average the amount number of all of my experiments to get the final result. As you see, there are 7 stable points on the table and all of them are connected with each other which means it will be able to consist of a stable area. Considering the evaluation part, I would like to say my function works well on these areas and it will be functional for real world.

However, there are some disadvantages of my function which I think I should discuss them too. First, there is the light problem. I already explained why I wanted to apply the light area to approach the goal of my project. However, it’s not perfect, especially for the glass rooms. It also gives me a new idea which I may should apply a new method to solve the problem which will be able to avoid light part. I will talk little bit more about this part on future work. Another problem is location of the door. My function relies on Corobot’s function which means Corobot should know which is the fuzzy location of door. It may not be really accurate, but it should know where is the door more or less.

Considering about the solution of my project, Corobot will at least be able to stand on stable area. However, it’s unknown result. If Corobot is not able to get this area, my function won’t work at all. The last one is the influence of human being. I test it and also discuss it on the experiment section. However, I didn’t represent it on the final result. But I need to say it’s also an important element to consider in real world which will affect my function. There is also a big advantage of my project. Since I can consist a stable area which means Corobot will be able to recognize whether it will be able to pass door during this area. In this case, Corobot will be able to run my function with moving. And it’s really important for Corobot to run in real world.

### 7. CURRENT STATUS & FUTURE WORK

In this paper I have presented an approach for a camera based door detection system. By using the edge extraction and light area, 7 reliable points have been found and the accuracy of most of them are more than 90 percent. In this case, my function will be able to return a good result even Corobot is running on the floor. And since my function is based on 2D image processing technology, it runs in real-time. Furthermore, I create a summary function which will be able to collect 100 times result in 10secs which will be able to increase the accuracy of result and remove noise when Corobot is moving or changing its camera.

However, there are some disadvantages which are necessary to be fixed in the future as I mentioned above. There are three plans which are most likely to try in the future. The first one is feature matching with a fuzzy voting function. As I mentioned before, I applied SURF and SIFT on my project and the results of them are bad. However, I think I may be able to fix them when I did experiments. For SIFT, we know matching function will be effected by noise. And if I can set up a database to save the locations of matching point during multiple times. The high frequency area will be the fuzzy location of handle of door. Since we only need to location a fuzzy location of handle to recognize the shape of door, it will solve the noise problem of matching part. However, it also has another problem which is moving problem. When Corobot is moving, the location matching is changing. To collect enough information to recognize that fuzzy area, I may need few seconds to run the matching part. In this case, it seems Corobot will not be able to locate the fuzzy area correctly. So it works best when Corobot stop and moves its camera to collect informations. Considering about the functionally of Corobot, I will be able to analyze the state of Corobot. In this case, I can run this function when I find Corobot is stopped. It will increase the accuracy when door has glasses wall around it.

The second one is color segmentation. Since the aim of my project is focusing on our building, I can apply the color of door to do a colorful segmentation which will be able to shape the door and the frame of door from colorful image. However, it will be effected by pictures or poster on the wall or door. I think if I can combine this with edge extraction part, it will be better. In this case, function will also be able to avoid glasses problem.

The last one is edge label. Edge labeling is using the static information of an edge to label the edge by simple topic, like the edge of door, the edge or wall and the edge of frame of door. This ideas comes from the paper of F.Mahmood and F.Kuwar[5]. They provided a method which is using learning
function to catch features of edge to recognize door. This may provide a good result based on my work.

8. REFERENCES