

Abstract

In this project, there are two main components. The first one is robot detects the landmark with Kinect. I propose an algorithm of which the time complexity is $O(n)$, faster than another landmark detection algorithm, which is $O(n \log(n))$ [22]. And for landmark detection algorithm, I add pixel RGB information which helps to improve the accurate of the landmark detection. The second one is when mobile robot detects the world with the Kinect, I use image segmentation method to let robot know which part robot can move toward. In detail, I use Gaussian model to train certain images which can be described the feature of the indoor area. Each train image represents one property of the indoor area, such as obstacles, walls or accessible areas. With the training rule, each pixel on camera frame can be classified into different regions. Each region represents the property of the indoor area, such as accessible areas, obstacles. Therefore, robot can know which area it is able to move toward with these regions.

Contents

Introduction.....	6
Aims and objective	6
Project Goals	7
Software Goals	7
The concept of mobile robot navigation	8
Simultaneous localization and mapping (SLAM)	8
The concept of SLAM.....	8
The Background of Mapping.....	9
How to represent map.....	9
Grid map.....	9
Feature Mapping.....	10
Topological map.....	10
The difficulties and challenges of SLAM.....	11
The data association problem in SLAM.....	12
The definition of data association problem	12
Map building and location problem	12
Scan matching	12
Local characteristics of data associated.....	13
Loop closure problem.....	14
3D point cloud	15
Kinect Control	16
The principle of depth detection of Kinect.....	17
Simple processing of depth image.....	18
Depth of image data histogram.....	18
Landmark Detection	19
How to find the gap between objects.....	19
The problem of Kinect detect the depth of image	21
Improved landmark detection algorithm	22
Remove the noise data.....	23

EKF-SLAM	24
How to determine the accessible area.....	27
Average Distance Calculation Method.....	27
Image Segmentation Method.....	27
Edge-based Segmentation	28
Mixture of Gaussians.....	29
Result & Data analysis	36
The 3D point cloud.....	36
Landmark detection with depth information of then image	36
Bad landmark statistics.....	37
Program problem.....	38
Kinect problem	39
How to solve.....	39
Compare with MRPT 3D visual SLAM.....	39
Convert the detect landmark to the 2D map view	39
Interpreting Kinect Data	40
Accessible area	41
Average distance calculation method.....	41
Image Segmentation Method.....	43
The EKF-SLAM.....	51
Discussion	54
The landmark detection	54
The Gaussian model	54
EFK-SLAM.....	55
Future work	56
Conclusion	56
Acknowledgements	57
Reference	58

Introduction

A mobile robot is an automatic machine, which is able to move around in a given environment. Mobile robots are capable of moving around in their environment and are not fixed to one physical location. By contrary, industrial robots usually are made up a jointed arm (multi-linked manipulator) and gripper assembly (or end effector) which is attached to a fixed surface [1].

SLAM technique is used to make a map within an unknown environment or to update a map with a known environment by robots and autonomous vehicles. SLAM is one of the most widely researched fields on robotics. It seems simple, and looks like a classic "eggs and chicken" problem. Therefore, how to map in the unknown environment is like the argument of "egg come first", whilst how to navigation on the unknown environment is like that of "chicken come first". A robot with camera provides move around in the environment in order to get some environment information, like objects and obstacles which are around the robot. If the human being is involved, it is not a difficulty problem; However, if the robot practices autonomously without any human assistance, it is an extremely tough problem. In the SLAM problem, mapping is the basically to determine the location of objects in the environment, and localization is to establish the robot's position with respect to these objects. Therefore, how to map is the basic issue of the SLAM, an excellent map algorithm and a wonderful map represent way will help robot to move better in the real world. On the other hand, how to detect the landmark is the basic problem of the mapping, so this project will put more emphasis on this with Microsoft Kinect. I will discuss some algorithms and improve them, and make use of some image process method to make mapping better. In SLAM part, I will discuss an important filter, which is named as Extended Kalman Filter, and make some changes on it and analysis the result.

Aims and objective

This project has two main aims. First, I will learn and implement the landmark algorithm to detect the landmark in order to let robot detect the real world. Then I propose an improved algorithm for Kinect to make landmark detection in order to adapt a variety of lighting conditions and to improve the accuracy of landmark detection. After that I will use Gaussian mixture model to make image segmentation in order to make robot more intelligence detect the objects and to select the accessible areas. And the second aim is to learn how to use iRobot Create to implement a simple action for robot with some reaction from the real world; For example, running around the object or avoiding touching the obstacles. Because SLAM is a very hard problem for me and I cannot implement it in just two month, so in this project I just learn the EFK algorithm and implement a simple EFK-SLAM application which can simulate how robot makes SLAM with known

landmark. At last, I try to combine the landmark detection and the simple slam, and make a real time slam (I have not finished this part)

Project Goals

1. The main goal of this project is to use 3D point cloud to make a better landmark detection. Thus, robot has a better platform to navigate.
2. To use some image process and machine learning way, in order to make robot detect the landmark more intelligence and to compensate for some shortage of Kinect.
3. To study and understand the SLAM algorithm.

Software Goals

1. To implement the landmark detect algorithm and to improve it with my algorithm.
2. To implement the simple EKF-SLAM Simulation on windows.
3. To combine the EKF-SLAM Simulation and landmark detection algorithm and to make SLAM simple.

The concept of mobile robot navigation

Navigation is the technique which guides the movement of ships, aircrafts or spacecraft's from one place to another place [2]. Autonomous robot only knows where I am and how to move from one position to another position. Therefore, in field of the autonomous mobile robot technical study, navigation technology is one of the core problems. The goal of mobile robot navigation is to make the robot move without any interventions. It needs to answer 3 questions: "where I am?", "where am I going" and "how can I get there?" When mobile robot enters an unknown environment, the robot must know where I am. This requires robot through continuously to explore the unknown environment and to build environment space model. The space model is called to be the robot navigation map. Navigation map is the abstract model of the physical environment. Only by the environmental navigation map, the mobile robot can execute some actions, like movement, and complete given task. In order to obtain the unknown environment space model, the following several sub problems have to be solved [3][4].

- Mapping (Environmental Modeling) [5][6]. This problem is on how to obtain environmental through sensor, and to achieve the physical environment of space model.
- Localization. The robot localization problem is a key problem on making truly autonomous robots. If a robot has no idea about where it is, it is difficult to determine how to do in the next step. In order to localize itself, a robot has access to relative and to absolute measurements giving the robot feedback about its driving actions and the situation of the environment around the robot. With this information, the robot is able to determine its location as accurately as possible. What make this difficult is the existence of uncertainty in both the driving and the sensing of the robot. The uncertain information needs to be combined in an optimal way[5][6].
- Motion Planning and Control. When robot makes navigation, it tries to find the optimize way to get to the destination.

Simultaneous localization and mapping (SLAM)

The concept of SLAM

Simultaneous localization and mapping (SLAM) is a technique used by robots and autonomous vehicles to build up a map within an unknown environment (without a priori knowledge), or to update a map within a known environment (with a priori knowledge from a given map), while at the same time keeping track of their current location[5].

SLAM problem can be divided into four basic aspects, as follows:

- How to describe the environment, how to make use of map to describe the environment.
- How to obtain information from the environment
- How to represent the environment information which is acquired. And update the map based on the environment information.
- Develop a stable and reliable SLAM system.

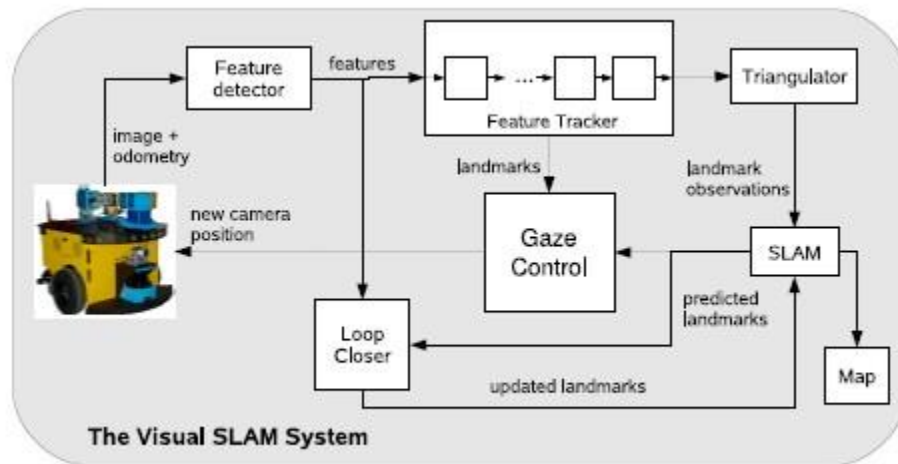


Figure 2.1 the visual SLAM system(Google search)

The Background of Mapping

How to represent map

Map is an important element in the robot world model. It has an important influence on sensor information process and SLAM. The map which robot built on processing SLAM can be divided into three categories: grid map, geometric feature map and topological map.

Grid map

In the year of 1968, W. E. Howden proposed the method of grid map. He translated the map into separate cells. And review whether or not each grid was occupied by obstacles and set each grid different identity value. Grid map identifies values to find out an optimal path, from the initial state to target state, without touching any collision. The advantage of grid map is accuracy. And the description of the path planning is easy. But the disadvantage is also obvious that the size of the cells must be smaller, in order to get more accurate optimal path. But the search algorithm will increase exponentially and it also needs an abundance of memory storage space.

Feature Mapping

Feature mapping refers to the abstract geometrical features which are collected from the environment. These geometric features, also known as landmarks. The environment is described to be the landmark. Figure 2.2(b) is the example of feature mapping. The circle in the map indicates the estimated position. Triangle indicates the trajectory of the robot. The elliptic curve represents the uncertainty region which is estimated by the state under a certain degree of confidence. Feature mapping is more compact, because it only needs a small amount of storage and it is easy to locate and to track the target. However, the extraction of geometrical characteristics needs further processing, of which the calculation is really big. Furthermore, in order to ensure the reliability of feature extraction, the high demand of accuracy of the sensor relative and the accurate modeling of the system are necessary.

Topological map

Topological map is composed of 3 parts: to division the state space. To build the characteristics map, and to search path in the characteristics map. The topology method use node to represent a specific location and use links to represent the relationship between these locations. The advantage of topology method is that it can reduce the search space, and it is easy to use symbol to make representation. There are no requirements of store space for too much data. The disadvantage is that when the environment is very complex, it is difficult to build a topological map. When the obstacles increased, how to insert a new node in a complete topological map is also a hard point.

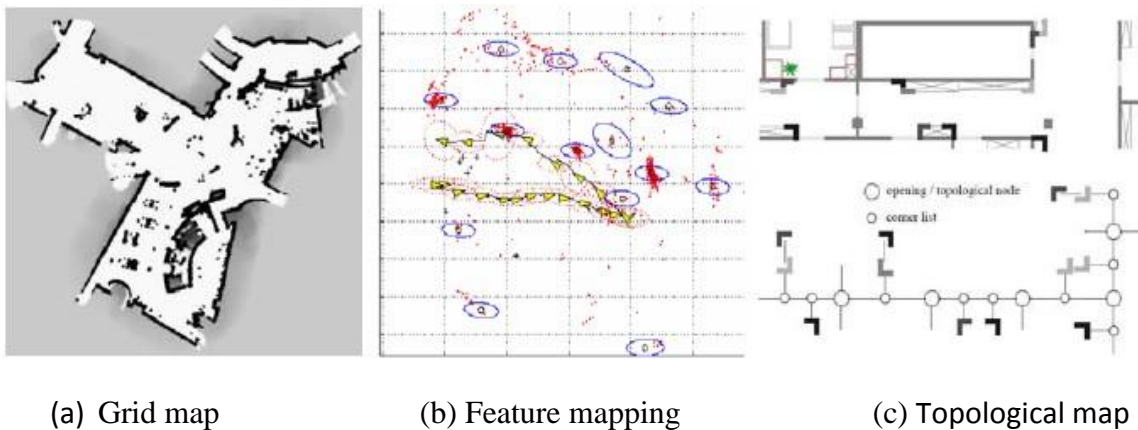


Figure 2.2 the examples of map representation (Xiucan Ji 2008)

When mobile robot builds map, an appropriate map representation method will be chosen by task and sensor according to the real environment. The method indicates above are flexible and a great number of applications has been used them.

The difficulties and challenges of SLAM

SLAM has a history of 25 years, and has made a large amount of achievements. But there are still many issues need to be resolved. There are 5 major difficulties and challenges [8].

First, there is a challenge from the inherent error of the system. In order to build the map, the mobile robot must get environment information through sensor. However, the measurement of sensor is often inaccurate, which is known as noise. On the other hand, the range of observation of robot is restricted. Therefore, when robot builds maps, it must make navigation. And there are also some errors on the movement of robot. The simple measurement cannot determine the position of robot. If the measurement error is independent on each other, the process of SLAM will be easier. Mobile robot can do more and more measurement to eliminate errors. Unfortunately, the measurement error on robot when building map is related to each other. As SLAM goes on, the measurement error will be amplified. Thus, how to fix measurement error and how to reduce measurement error at the beginning of the SLAM is the key factor of the robot mapping. Also it is the one of the main reasons that the SLAM problem is a complex. So a number of existing algorithm, both mathematics and implementation, are surprisingly complex.

Secondly, another challenge of SLAM is the high dimension of map. A detailed two-dimensional map may need thousands of feature description if topology map is not used. Therefore, SLAM problem has an amazing high dimension. And how to deal with high dimension data is a statistic estimation problem. Meanwhile, it is also one of the difficulty points of SLAM problem.

Thirdly, data association, which also known as the correspondence problem. Data association means that the corresponding relationship between map features and data from sensor based on different time. The correction of data association is the key factor of state estimation of SLAM. Therefore, the data association with its related problem is the main content of this thesis.

The forth problem is on building map on the dynamic environment. At the present stage in artificial intelligence and pattern recognition technology field, the robot is difficult to distinguish between static environment and dynamic object in static environment, even though, robot is able to distinguish between moving object and background environment. The model of dynamic environment is also a big challenge. The majority of robots building map algorithm is unable to establish meaningful dynamic environment map. On the contrary, mapping is dependent on hypothesis of static environment. The robot is the only one dynamic object in the environment. Other moving objects can be seen as noise. Therefore, we can set time space into small piece. Each time piece we assumed that environment is static.

Fifth, choose the right path to make exploration. The robot exploration is facing part, incomplete environment model, so any feasible exploration strategy must be able to deal with emergency situation. We have to say SLAM problem is the most difficulty problem is robot field.

The data association problem in SLAM

Figure 2.3 indicates the basic process of SLAM. It is divided into three layers. Bottom layer is for perception. Robot gets their motion perception information from the internal sensor. Middle layer completes the process of sensor information. The top layer is for robot localization and map estimation, which usually is the sense of SLAM problem. The researchers generally believe that SLAM problem can be divided into two parts: state estimation and data association problem [9]. State estimation problem refers to the problem of environment characteristics (for grid map it is the obstacle, while for feature map it is namely landmarks), and the problem of position and robot positioning estimation. The solution space is continuous, and the solution space of data association is discrete. Data association for state estimation provides input. The correct data association is to achieve correct state estimation of the premise. So the data association for SLAM problem is very important.

The definition of data association problem

The data association refers to SLAM is based on different time between sensor and sensor measurement, and on different location between map feature and real world. It also includes the process, which converts the old characteristics, not matching the map, to the new characteristics, matching the map. In brief, Data association problem is on the measurement corresponds to landmark. Data association for SLAM problem is very important. Incorrect data association will make maps divergent, and even lead to the failure of the whole SLAM process[10]. Data association is mainly composed of three sub-problems, as follows: map building and location problem, circulation closed problem and many machines in SLAM map merger problem.

Map building and location problem

According to the latest sensor information, map building and location problem refers to the process of creating maps and self-localization. Based upon the map representation methods, data association can be classified into two categories: scan matching and local characteristic data association.

Scan matching

In the process of the establishment of grid map, the sensor mainly use range sensor (such as ultrasonic array, laser radar, etc.). A sensor measurement is called as a scan. In general, scanning point does not exist between direct corresponding relationships. Therefore, mobile robot needs to ensure that the same physical entity area from different scanning in

order to acquire the robot relative position. This kind of data association problem is named as scanning matching problem. Scan matching process is shown in figure 2.4. Figure (a) indicates the map image the robot gets from sensor in the position of A and B. And they are respected by the dot line and the solid line. The aim of scan matching is to find two public parts in two images and to determine the relative position of A and B like figure (b). There are many algorithms about the Scan matching problem, like ICP [11][12] algorithm, histogram matching algorithm [13] and normal distribution transform method [14].

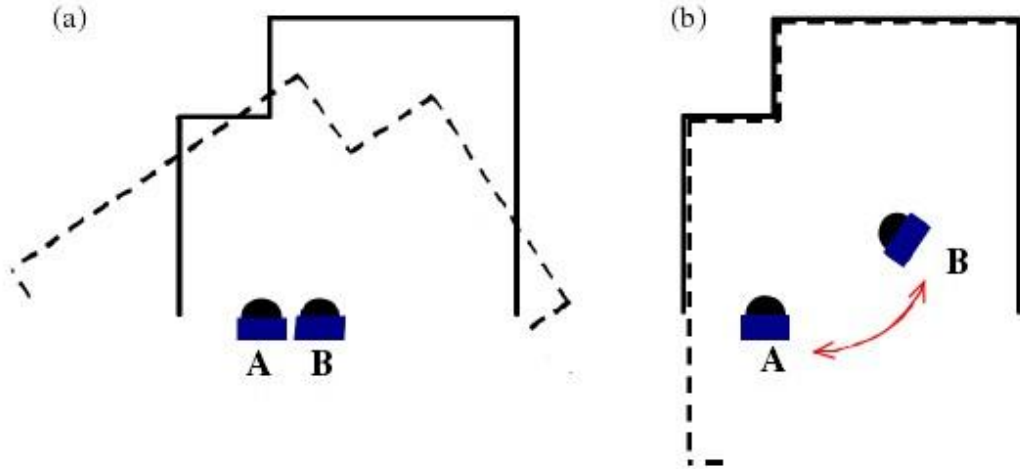


Figure 2.3 Scan matching diagram [15]

Local characteristics of data associated

On the process of building feature-based map, the related data needs to be determined several times by the corresponding landmarks or the current observed landmark with the recently created maps. As information showed in the figure 2.4, figure (a) indicates that there are 21 landmarks in the map. Figure (b) declares that the distribution of the observed landmarks in robot coordinate system and which contains 8 landmarks. These 8 observed landmarks $\{1,2,3,4,5,6,7,8\}$ is corresponding to the landmark in global map $\{20,21,18,19,17,16,15\}$.

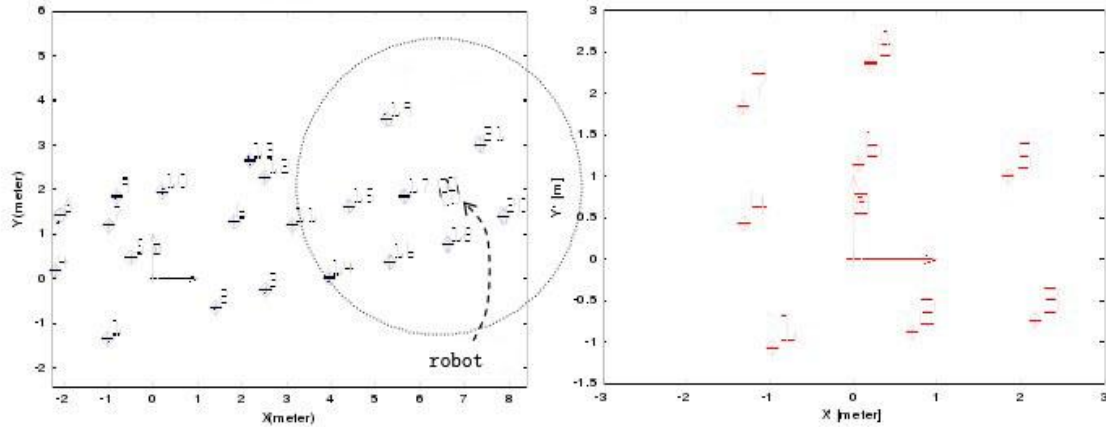


Figure 2.4 Local feature data association diagram [15]

Loop closure problem

SLAM is a self-positioning process of dead reckoning. The error on robot SLAM process is generally incremental and unbounded. The ideal state is that there is a large cycle in the environment, when robot moves around this cycle once; However, due to the gradual accumulation of the robot position error, it is difficult for robot to estimate whether robot has been back to the start point of the cycle. Therefore, it is necessary to get all aspects of information to confirm that robot has been back to the starting point of cycle, and get the relationship between current observations and circulation at the beginning of the map features. In this way robot can make error correction when robot make mapping. This is the SLAM loop closure, regarded as closed-loop problem. The closed-loop process consists of multiple sub-problems: closed-loop detection, closed loop associated closed-loop confirmation, status optimizes and etc. [16]. One of the key issues is to judge which creates earlier between the most recent observations map and the local map. It is shown in the figure 2.6 that after the robot complete the cycle SLAM and go back to the start point there may be a large position error. And it is suggested adopting other measures to deal with this problem.

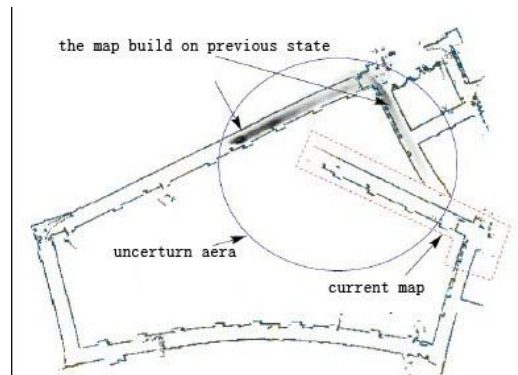
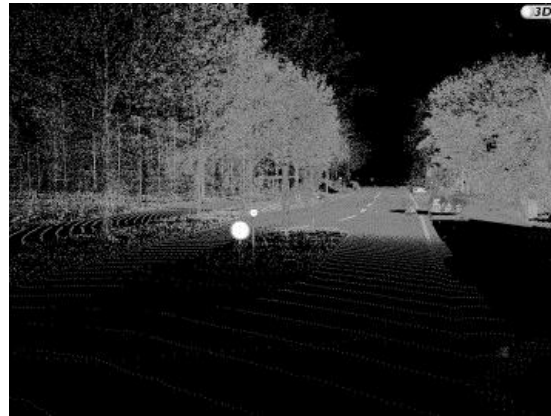
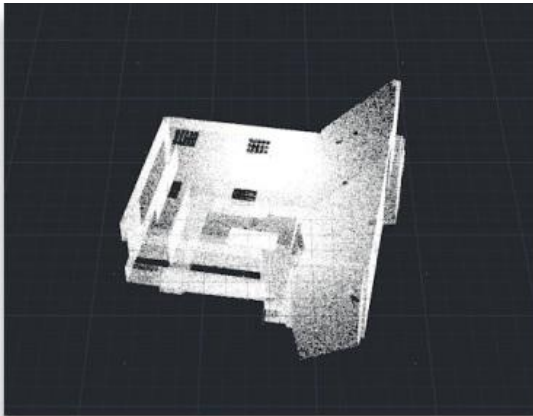


Figure 2.5 Cycle data associated problems diagram [16]

3D point cloud

In the 3D world the point consists of the coordinates of x , y and z . And each point in the real world has the 3D world distance. A point cloud is a set of points in a 3D space. You can view the cloud from different angles and lighting conditions. You can even apply colors and textures to its surfaces. A point cloud is one step away from a full 3D model and sometimes it is more useful. For robot, the original technique for robot view is that we treat each image for robot is a 2D image. We only use the color information to describe the feature of the object. There are many disadvantage of using 2D image as robot view. First, it is hard for robot to know the structure of the indoor space, 2D image only can describe the world with two dimensional. It cannot tell robot how depth of the world, and also it cannot tell robot how far away from the obstacle. Secondly, Robot must use another sensor to help let it know where I am, and what kind of the environment around the robot. Many other techniques must be used to assist the robot to detect the area, such as GPS, infrared ray, Ultrasonic wave and so on. Using 3D point cloud can help robot to know the world more exactly. 3D point cloud not only can describe the color, but also the shape, depth and the textures of the world. Therefore 3D cloud point can help the robot to know the world better. This project I will use Kinect as the robot's "eye" and try to use Kinect to detect better landmark with 3D point cloud and use some image process technique (e.g. Image segment, image sharp detect technique) to make the robot mapping in a better detection way.



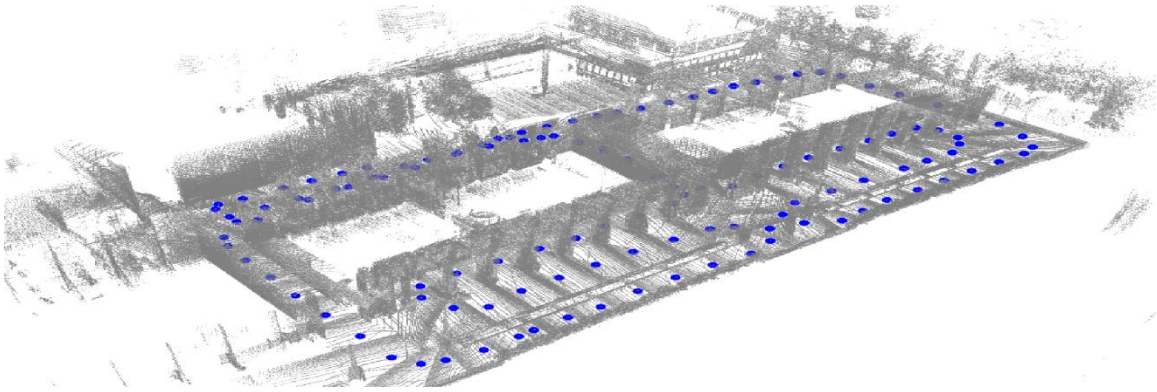


Figure 3.1 some point cloud image used as navigation (from Google image search)

Kinect



Kinect builds on software technology, developed internally by Rare, a subsidiary of Microsoft Game Studios owned by Microsoft [18]. Kinect is the low-level camera which can be used to detect the 3D object and the depth of the image. Based on the basis of the Kinect support and the windows Kinect SDK 1.5, I can implement the 3D detect landmark algorithm. And the target I use Kinect is to get the RGB and depth information in the real world.

Kinect Control

With the Kinect for windows SDK, I can program on Kinect easlier. It provides many methods to get information which I want to get. There are two methods which is most useful in my program. One method is to receive the location of people in the frame, using the information stored in the depth array. And another is to get the RGB information of each pixel from the color array. When using the Kinect for windows SDK. If Reflective surface was detected by the Kinect, it would return a special value of depth which is -1. This value we can set it as bad data, we can ignore it sometimes and removed in from my landmark detection algorithm. Although it will effect the obstacle detection, I will solve this problem in next section. Finally, for the issue of cutting down some data, some of the data is useless like the data near the bottom of frame image. From the experience of

human being that when robot make navigation, the information of image center is more useful than that of any other range. There are 2 good reasons for the reduction the frame size when robot makes detection. One is reducing the frame size can reduce the size of the data processing. And the other is to avoid some noise affection. Thus, a horizontal slice of the depth information is taken. For example, from the array of 640x480, only the data points where the height is 200 may be used in the algorithm. Such a slice can be seen in Figure 2. By reducing the data, the SLAM problem simplifies from a 3D slam algorithm to a 2D algorithm, and the computational intensity is greatly reduced [20].

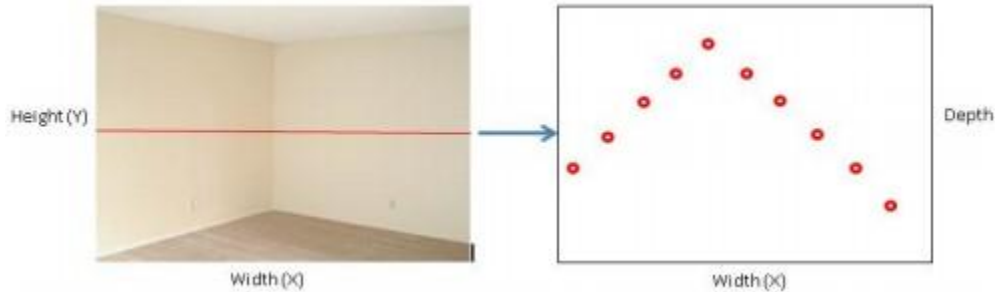


Figure 3.3 taking a slice of the depth data at the red line (Richard Marron Advisor: Dr. Jason Janet 2012)

The principle of depth detection of Kinect

It is same as the other cameras; infrared camera in Kinect also has the field of view. The vision of the Kinect cameras is limited. The detail is showed below:

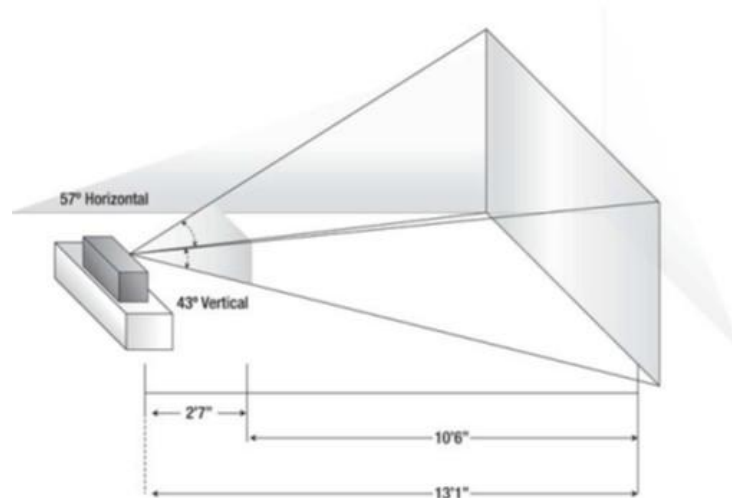


Figure 3.4 (Microsoft Kinect msdn 2010)

As the figure 3.4 indicates the view of infrared camera's field is a pyramid shape. If the object is far away from the Kinect cameras, it will have larger field of cross-sectional

area than the object which is close to the Kinect cameras. This means that the height and the width of the image, such as 640 x 480. It does not correspond to the physical location of the field of the view of the Kinect camera. But the depth value of each pixel is corresponding to the visual field of the object distance from the Kinect camera. Depth frame data in each pixel represented 16 bits and two bytes for each pixel. A depth value for each pixel occupies only 13 bits in the 16 bits.



Figure 3.5 (Microsoft Kinect msdn 2010)

Simple processing of depth image

We discussed how to obtain the pixel depth values, and how to generate an image based on the depth value. We filtered out of the points outside of the threshold value. This is a simple image processing, called the threshold processing. Using threshold to filter the point is a little rough, but useful. A better approach is the use of machine learning way to calculated threshold from the image data of each frame. Kinect depth value of a maximum of 4095mm. value 0 and -1 usually indicates the depth value which is unacceptable value, usually the value of 0 will be filtered out. Microsoft recommends that when user is making development, the value range from 1220mm (4 ') to 3810mm (12.5') can be used. Before we process depth image, we should use the threshold filter depth data to the range from 1220mm to 3810mm.

Using statistical methods to deal with the depth of the image data is a very commonly way. The threshold can be determined by the average value of the depth of the data. Statistical methods can help us to determine which point is the noise: shadows, or other meaningful comparison objects. For example, the part of the obstacles. Sometimes the pixels of the visual sense are not considered, we can attempt to use the original depth of data mining. The purpose of the depth data processing is to make the recognition of objects' shapes or objects space relationships. With this information, the program can determine the position which respects to the Kinect.

Depth of image data histogram

The histogram is a very effective tool for statistical data distribution. Here we are concerned on the distribution of the depth value of image. Histogram can intuitively

reflect the distribution of a given data set. From the histogram, we can see the frequency of the depth value and aggregation packet. With this information, we are able to determine the threshold values, and use threshold value to filter the image; therefore, this way can make it possible to maximize the depth data information in the depth image. We will show the histogram of the image data with depth field and use some simple techniques to filter out the pixels in section 5.5.

Landmark Detection

When we want to detection object as landmark, we usually detect the edge of the object first. The parts surround the edge usually are the main body of the object. So in this way we transform problem on the detection of the edge of the object.

If there are two object are connected in the real world, we can treat these two objects as a whole one (if two objects' color is similar). Therefore if two objects are not close enough, there will be a gap between each object, and the gap is the key factor which we want to find to detect the edge of the object.

How to find the gap between objects

Objects on the image are composed by the pixel and we treat each pixel as a point. Therefore, we can use the consecutive points and the neighbor points to describe an object. If the points are on the surface of the object, the slope of the point between point and its neighbor tends to 0. Thus, there is a slope of the point between point and its neighbor with significant value. That means this point is the edge point and we treat this point as edge of the object.

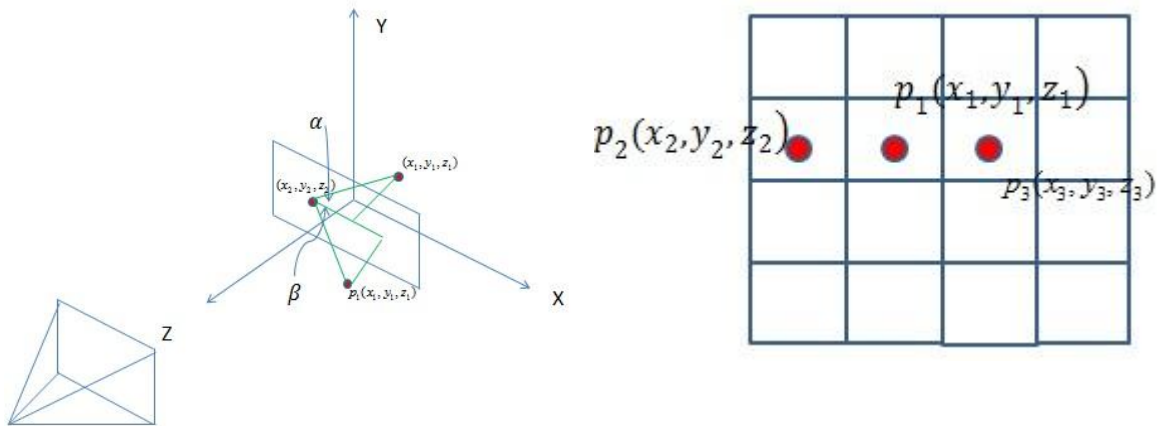


Figure 3.6

For the slope $p_1, p_2 = \tan \alpha = \frac{z_1 - z_2}{x_1 - x_2}$

For the slope $p_1, p_3 = \tan \alpha = \frac{z_1 - z_3}{x_1 - x_3}$

The Pseudo-code shows as follows

```
For each pixel in image
If pixel is a good data
If the point is closer to the robot than its neighbor
Rightslope = slope between the pixel and its closest right pixel
Leftslope = slope between the pixel and its left pixel
Slope = abs(Rightslope - leftslope)
If slope > threshold
Set pixel as landmark
```

There are several features of the algorithm above. All the pixels are iterated only once. Therefore, the time complexity of algorithm is $O(n)$. This algorithm is improved, compared with other common landmark extraction methods which often run $O(n \log(n))$. The bad data are ignored, but if there are two points laying on each side of a section of bad data, they will be still treated as neighbors. The reason why I use this precaution is that the Kinect often regard the sharp and reflective surfaces as bad data. These bad data points are likely to be the results of the IR lasers hitting the surface at a glancing angle significantly distorting the Kinect's laser array making it un-interpretable [20]. These objects with sharp and reflective surfaces are exactly the type of surface which the algorithm attempts to detect. Unfortunately, many landmarks are undetected if ignoring of these points. In addition, this algorithm will ignore the landmarks which are far away from the Kinect, even though the landmarks are neighboring pixels. This review is to ensure that the landmark detected is on the confidence which is 1220mm to 3810mm (*Microsoft Kinect SDK 1.5 Document*).

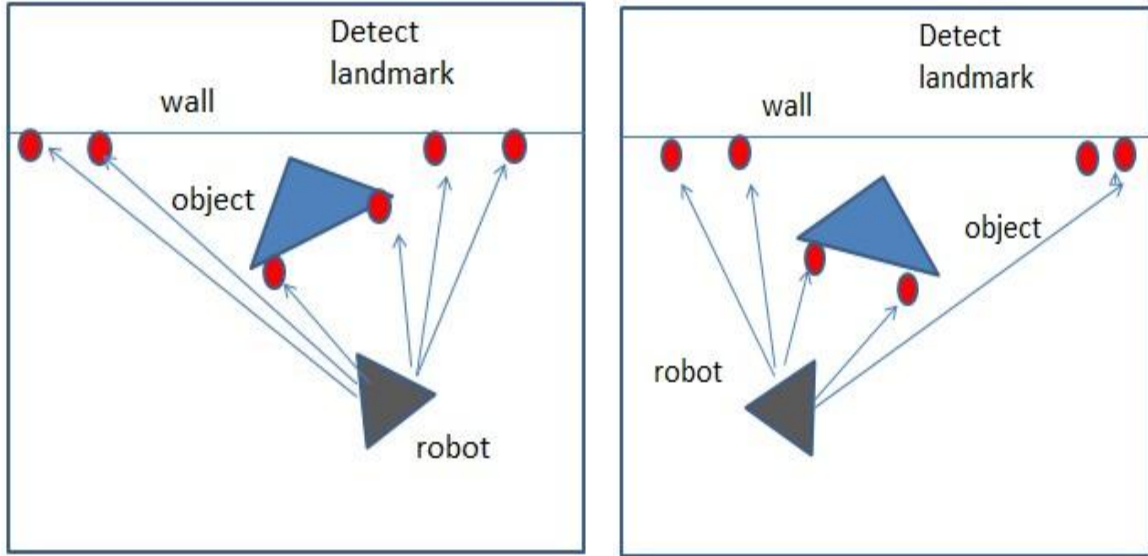


Figure 3.7

In Figure 3.7 the red points represent the pixels which are detected as landmarks. We can see that the landmark on the objects still remains consistent with the object's position, but the landmark behind the objects shift along the back surface.

The interesting part of my work is on the landmark detection algorithm. It calculates the difference in slope between the central pixel and its neighbor pixels. If the algorithm only takes care on the slope value and do not calculate the difference in slope. There may be a problem like that if a point on the wall with 90 degrees to the robot direction, the robot will still think that will be a landmark. This will make the deviation of landmark detection being larger.

When landmark has been detected, it will be translated to estimate the original SLAM reference frame.

The problem of Kinect detect the depth of image

Kinect is the low – level detect device. It cannot get accurate distance of each point. It only can detect the distance of each point less than 4095 millimeter [23]. There is a problem with Kinect device. The other problem is the Kinect cannot detect the distance for the close objects. Sometimes the Kinect will return the value -1 when robot closes to the object. It not means that the robot is very close to the object, conversely, if reflective surface was detected by the Kinect, it would return a special value of depth which is -1. Therefore, I improve the landmark detection algorithm and add some color information to the algorithm.

Improved landmark detection algorithm

My experience of viewing the color on the surface of an object is that the color on the surface of an object will be similar or change in a gradual way. In other words, there is little difference in pixel between pixel itself and its neighbor on the surface of the object. If there is too much difference between pixel and its neighbor, the pixel is likely on the edge of the object.



Figure 3.8 the edge point (red dot) on object

We can see clearly that the red dot on the object is the edge point. As to the surface of object, the color of the edge is blue, and the surface of the desk is the yellow. By this way, Kinect can easily detect the edge.

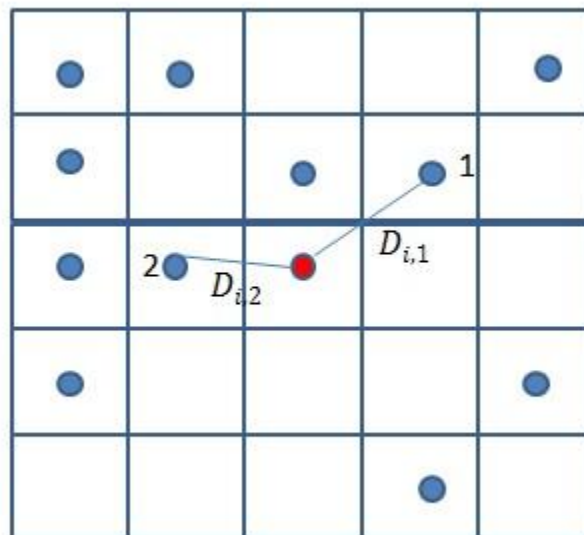


Figure 3.9

The figure 3.9 shows the result of searching for the nearest neighbor for the red point. The voxel center of a plane of the 3D map is shown as black dots. First we compute the slope of $D_{i,k}$ between the position of a point i of the current point and a center k of the current map performed as follows:

$$D_{i,k} = \tan \alpha = \frac{z_k - z_i}{x_k - x_i}$$

It is similar with the algorithm which I method in previews sections. We calculate the difference in slope between pixel and its neighbor first. Then in this algorithm we need to calculate more information on the pixel which is the RGB information of pixel and its neighbor.

$$W_{color} = \sqrt{((R_i - R_k)^2 + (G_i - G_k)^2 + (B_i - B_k)^2)}$$

$$W_{pixel} = \alpha W_{color} + \beta D_{i,k}$$

Where $\alpha + \beta = 1$. And W_{pixel} is the weight of the pixel. The pixel will be detected as the landmark with bigger weight than that of the thresholds. There are some differences between this algorithm and the algorithm which has been method in previous section. There is some color information in this algorithm. I set different weight for the color and the depth of image. Therefore, how to determine the α and β will be the key factor which affects the result. According to *the Microsoft Kinect Document 1.5*, the left of the first circle on Kinect is IR Illuminator. IR Illuminator is used to get the depth of the image it will not be affected by the brightness of the environment. In other words, no matter the brightness of the environment, the data of depth information of image will be same. There will be a big effect on the color information of image when the brightness of environment is changed. Therefore, the color detection is more sensitive to the light. So we can set the weight of α and β referring to the brightness of environment. If the environment becomes darker, the α is set to a smaller value and the β is a bigger value. If the environment becomes brighter, the α is set to a bigger value and the β is a smaller value.

Remove the noise data

In the indoor area there is much noise like the dynamic objects (e.g. the moved people, the change of light and so on). Kinect has many good algorithms to detect the human beings. For instance, if there is a human moving into the Kinect's view, Kinect will detect him and get his information immediately.

EKF-SLAM

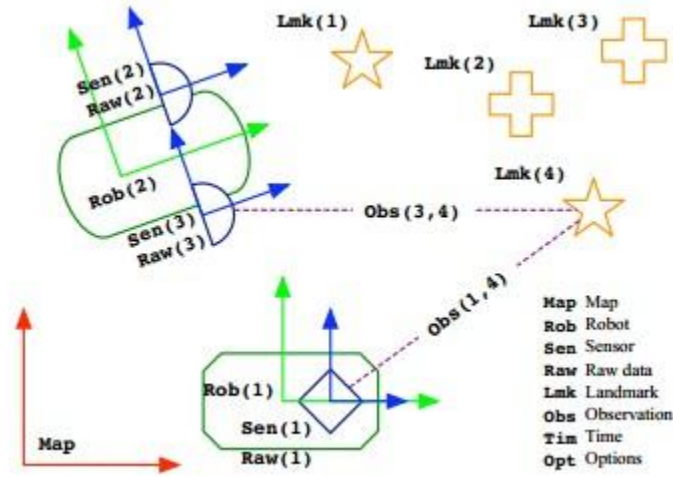


Figure 3.10 typical SLAM entities [24]

After the new landmarks being associated with the existing ones, the SLAM algorithm uses an extended Kalman filter (EKF) to correct the robot's position based on the error between the associated landmarks.

The following table indicates the similarities and differences between EKF and EKF-SLAM.

Event	SLAM	EKF-SLAM
Robot moves	Robot motion	EKF prediction
Sensor detects new landmark	Landmark initialization	State augmentation
Sensor observes known landmark	Map correction	EKF correction
Mapped landmark is corrupted	Landmark deletion	State reduction

Table 3.1 EKF operations for achieving SLAM [24]

(Kalman Filter, KF) is an efficient filter with recursive way. It can estimate the dynamic state of the system by a series of noise measurements. The simple Kalman filter must be applied in the Gaussian distribution system. And one of the improvements is extended Kalman filter which can be used in nonlinear system. The implementation of EKF-SLAM is based on the establishment of the robot's movement and observation model. Robot movement model is below:

$$x_k = f(x_{k-1}, u_k) + v_k$$

$f(\bullet)$ is nonlinear function and it describes the movement of robot. u_k is the control vector which apply in the robot on the time $k-1$ and let robot reach state x_k on the time k . v_k is the Gaussian noise. And the covariance Q_k is used to describe the error on the movement of robot.

On the movement of the robot, robot uses sensor to detect the landmark on the environment, we use observe model to represent it. And on the time k the observe model is below:

$$z_k = h(x_k) + w_k$$

Where nonlinear function $h(\bullet)$ is used to describe observes model of robot. w_k is the Gaussian noise. And the covariance R_k is used to describe the error on the movement of robot.

Taking advantage of these two models, EKF method can be applied into SLAM. EKF-SLAM can be realized via the iterative process of forecast, observe and update. \bar{x}_k stands for prior probably estimation ($\bar{x}_k = x_{k|k-1}$), and x_k^+ for posterior probably estimation. For other variables, "+" and "-" share the same meaning.

1. Predict

In this step, the robot state is predicted to be \bar{x}_k at the moment of k , the covariance is P_k^- .

Observing \hat{z}_k^- , the calculation formula is as follows:

$$\bar{x}_k = f(x_{k-1}^+, u_k)$$

$$\hat{z}_k^- = h(\bar{x}_k)$$

$$P_k^- = \nabla f_{x_{k-1}} P_{k-1}^+ \nabla f_{x_{k-1}}^T + Q_k$$

Where, Jacobian matrix $\nabla f_{x_{k-1}}$ is the $f(\bullet)$ taylor expansion at x_{k-1}^+

$$\nabla f_{x_{k-1}} = \frac{\partial f}{\partial x} | (x_{k-1}^+, u_k)$$

2. Observe

After observing x_k when the robot state is z_k , the innovation V_k can be calculated:

$$V_k = z_k - \hat{z}_k^-$$

In the assumption of the good data correlation, the new covariance of innovation is:

$$S_k = \nabla h_{x_k} P_k^- \nabla h_{x_k}^T + R_k$$

Where ∇h_{x_k} is the linear approximation of the observe function $h(\bullet)$

3. Update

The last step is to update the robot state estimation x_k^+ and the covariance P_k^+ . The formula is:

$$x_k^+ = x_k^- + W_k V_k$$

$$P_k^+ = P_k^- - W_k S_k W_k^T$$

Where, the gain of Kalman is: $W_k = P_k^- \nabla h_{x_k}^T S_k^{-1}$

SLAM can be solved by practicing these three steps iteratively. EKF-SLAM provides a system to solve the problem. And much work has been done on its convergence, the growth of map, the changes of uncertainty and etc. However, EKF-SLAM still has many problems in practice. First, the linearization of nonlinear function may result in errors in the system. The second problem is the Gaussian Model for noises and the irrelevant hypothesis are often unacceptable. Both of these two factors may cause the uncertain operation of EKF-SLAM. In the meanwhile, the algorithm depends on the correct motion model and observe model, so it is critical to model the robot and the environment. EKF-SLAM is usually suggested avoiding using for those applications where it is hard to model the robot and the environment.

While this paper I try to implement the extended Kalman filter follow the paper: Simulation localization and mapping with the extend Kalman filter (Joan sola August 26, 2012). SLAM is a very hard problem to implement, therefore follow this paper can help me to understand the ekf-slam algorithm easier. My target is follow this paper to implement a simple ekf-slam simulation software and try to combine the landmark detect algorithm with it.

How to determine the accessible area

In the last section we have already discussed the algorithm to detect the landmarks in the real world, but it keeps silent on which part or which direction is accessible for the robot. Kinect can detect the depth of the image. Each pixel in the depth image frame has a distance value. But the distance of each pixel is independent, which means we cannot get an exact value when we want to know how far away from the object. Because the distance of each pixel on the surface of object is different, so if we want to get the value of distance we must let Kinect know what part in the image is the object, which part in the image is the wall, which part is accessible.

Average Distance Calculation Method

When calculating the distance between the camera and its face objects, we only need to calculate the average distance of the depth of each pixel in the image [25].



Figure 4.1(camera move close to the object (Obstacle))

Figure 4.1 indicates that camera is approaching the object. The resolution of the camera is 320 x 240, which means that there are 76,800 pixels in the image. As the camera moving to the object, there will be more pixels on the surface of the object. Theoretically, the closer the camera is to the object, the more accurate the parameter of distance is. And the highest accuracy can be reached when the pixels are occupied the entire image. Therefore, this method can roughly estimate the distance between object and camera.

Image Segmentation Method

Image segmentation is the division that an image divided into regions or categories, which is corresponding to different objects or parts of objects [25]. Every pixel in an

image is allocated to one of certain number of related categories. A good segmentation is typically one in which:

- The value of pixels' grayscale in a same area or category is similar, which is also leading to a connected region being formed.
- The value of pixels varies with categories.

Besides the fact that we can treat each frame as one image, and applying the knowledge that Kinect detects the world with 30fps, we could use image segmentation method to tell the object and the wall apart. Depending on this measure we could divide image into different regions with likelihood. Each region would be represented by certain property, such as, "wall", "obstacle" or "closed object".

Edge-based Segmentation

General ideas:

- Detecting strong edges
- Extending or deleting them in order to create closed boundaries

There are some common problems, which are caused by image noise or adding information in the image like edge presence in locations where there is no border or no edge presence where a real border exists.

Conceptual examples:

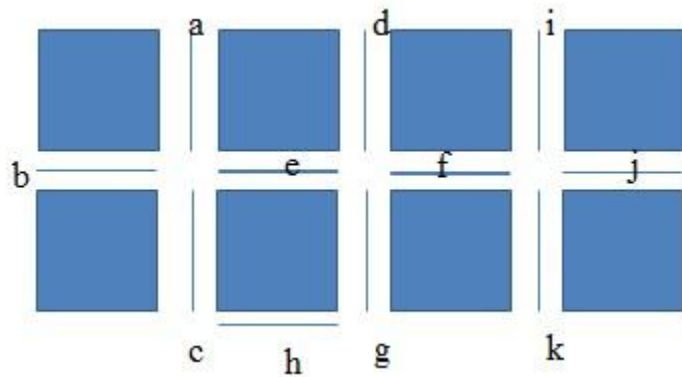


Figure 4.2 Minimal edge neighborhoods

A weak edge positioned between two strong edges provides an example of context; it is highly probable that this inter-positioned weak edge should be a part of a resulting boundary. On the other hand, an edge (even a strong one), which is positioned by itself with no supporting context, is probably not a part of any borders. An assumption could be proposed that strong continuous edges in the image are related to region boundaries of interest.

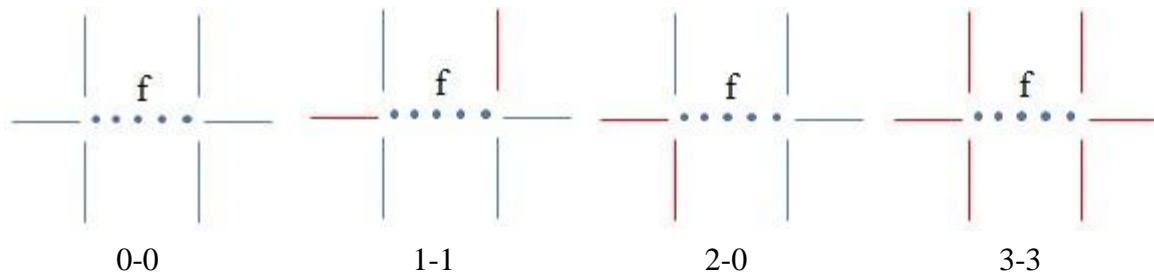


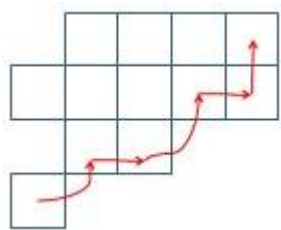
Figure 4.3

- 0-1 uncertain – weak positive, or no influence on edge confidence
- 1-1 continuation- strong positive influence on edge confidence
- 1-2,1-3 continuation to border intersection – medium positive influence on edge confidence
- 2-2, 2-3, 3-3 bridge between borders – not necessary for segmentation, which is influencing on edge confidence [25].

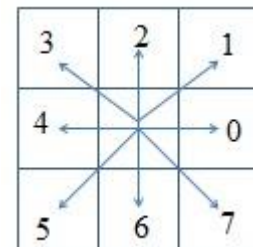
The method how we trace the region boulder:

Greedy border tracing:

Building region boundaries by following strong edges until closed regions emerge



greedy border tracing



8-connectivity

Figure 4.4

Graph search builds a graph of detected cracks, their strength (weights) and their adjacency; then they are tried to be connected in the shortest path from a given start node to an end node, which creates a region boundary. Dynamic programming finds boundaries in local sections of the image and combines them as a global solution. It is a little bit difficult to apply this means in programming, because there would be numerous of time being spent on processing the image for each frame when the resolution is big.

Mixture of Gaussians

For one image, Robot is interested in some parts of it. Those parts are occupied in some areas of the image. And some features (Such as wall in room, accessible area, object and obstacles, etc.) are different between the adjacent images. Some of these features could

be obtained easily. But some others might be very subtle, and even tough to be detected by human's eyes. With the development of computer image processing technology, image is possible to be obtained and processed with computing image information. The image segmentation is the foundation of image detection.

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal-tract related spectral features in a speaker recognition system. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori (MAP) estimation from a well-trained prior model. (Douglas Reynold 2008).

Gaussian mixture model has a very good application in the processing of each frame in video images. In this project Kinect is used to make detection from the real with 30 fps. We could treat each frame with single Gaussian model; therefore we have 30 single Gaussian models for each second.

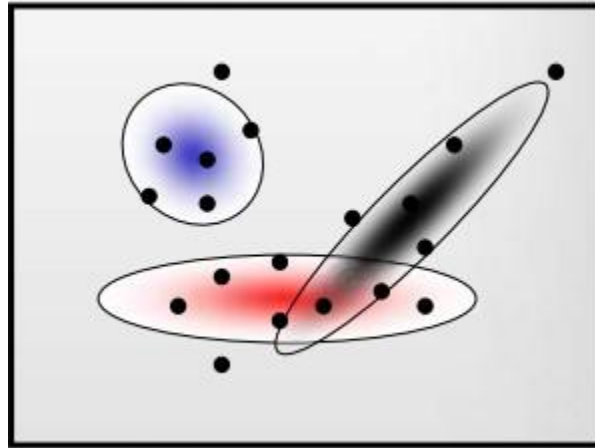


Figure 4.5 (Douglas Reynold 2008).

Before we use Gaussian mixture model, our training set for Gaussian mixture model should be selected. Because of our environment area is in door area, I determined to divide our room space into three categories:

- Wall
- Obstacles
- Accessible path or accessible space

Therefore, I choose 3 images as the training set represent these three categories as shown in Figure 4.6.

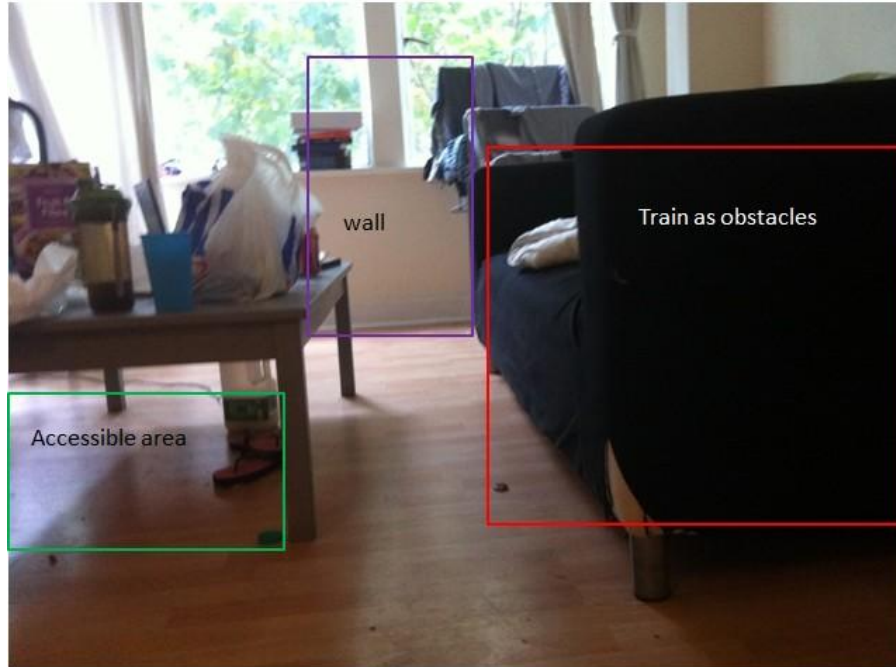


Figure 4.6

I took an image shot cut of my indoor area. From this image we could see that 3 rectangles are used to set 3 parts as the training set. (The more training set the better Segmentation area for the image) The most pixels in red rectangle are on the sofa and present the obstacles that cannot be accessed. The pixels in green rectangle mean that area is available for robot. And the purple rectangle represents the wall, which means it is the border of the indoor area (the border of the map). The three training set is like below:



Sofa (obstacle)

Floor (accessible)

Wall (border of the map)

Figure 4.7(3 train set for my indoor area)

The training set need to be processed when we determine the training set. Firstly, we make a statistics for the pixel of the training set. And the histogram of the training class image would be calculated. Then I can get two important parameters. One is the mean of

the histogram μ and the other is standard deviation σ^2 . Therefore, I can use Gaussians found to get probability of each pixel.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Then we will use the prior and the learned Gaussians base on Bayesian inference measure to predict the most probable content class for each image pixel.

$$p(class | pixel_i) = \frac{p(pixel_i | class)p(class)}{p(pixel_i)}$$

Therefore, our target is to get each $p(class | pixel_i)$ and to compare probability of $p(class_0 | pixel_i)$, $p(class_1 | pixel_i)$ and $p(class_2 | pixel_i)$. And I classify the pixel to the corresponding class which probability is the biggest in three classes. Therefore each pixel can get its own corresponding classes, which belongs to 0, 100, and 200 (RBG value).

The detail process is showed below:

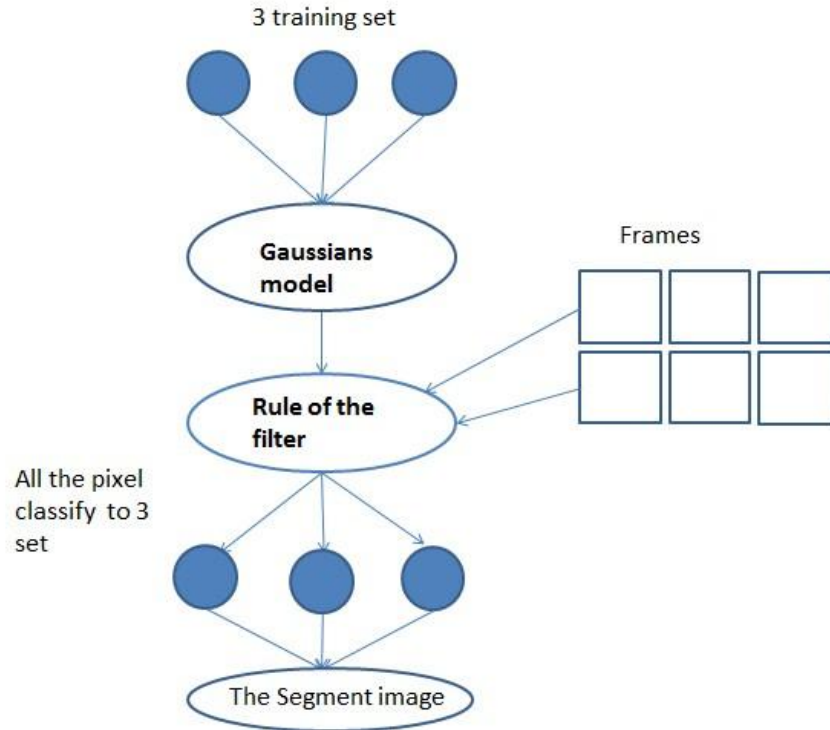


Figure 4.8

The figure 4.8 indicates the process how to train with training set and how to use classify set to generate the new segment image.

Key Issues:

In Bayesian classification, each $p(pixel_i) = 1$. Therefore, I get Bayesian classification formula as follows:

$$p(class | pixel_i) = p(pixel_i | class)p(class)$$

Three images will be used as the training set. Therefore, the prior probability of each class $p(class)$ is set to 1/3. Thus, the key fact that affects the performance is $p(pixel_i | class)$. In other words, the key fact is the Gaussian distribution for each probability of pixel. According to the Gaussian distribution formula, μ and σ^2 determine the probability of distribution. Therefore, how to calculate μ and σ^2 will be the main method to increase the accuracy of segmentation.

I use two methods to calculate μ and σ^2 . One is using threshold to calculate the μ and σ^2 , the other is to calculate the average value of each pixel to get the μ and σ^2 . The reason why I use a different method to calculate μ and σ^2 is that different images have different contrasts. Therefore, only use the general way to process all images will get the weak segmentation on some exceptional images. Sometimes using some special way to process some special image will be the better choice.

Average value

This is an easy way to calculate the μ . The time complexity of this algorithm is O(n). The formula is as follows:

$$\mu = \frac{\sum_{i=1}^n pixel_i}{n}$$
$$\sigma^2 = \frac{\sum_{i=1}^n (pixel_i - \mu)^2}{n}$$

The threshold method

To find a threshold we can use an image histogram. First we count how many pixels in the image have each value. Then show peaks around regions of the image. Here is the algorithm to select the global threshold [27].

1. An initial estimate will be selected for the threshold T.
2. Segment the image using T. This will produce two groups of pixels: G_1 consist of all pixels with grey levels $>T$ and G_2 consisting of pixels with grey values $< T$.
3. Compute values of the average grey level m_1 and m_2 for the pixels in regions G_1 and G_2 .
4. Compute a new threshold value: $T=(m_1 + m_2) /2$
5. Repeat steps (2.) though (4.) until convergence.

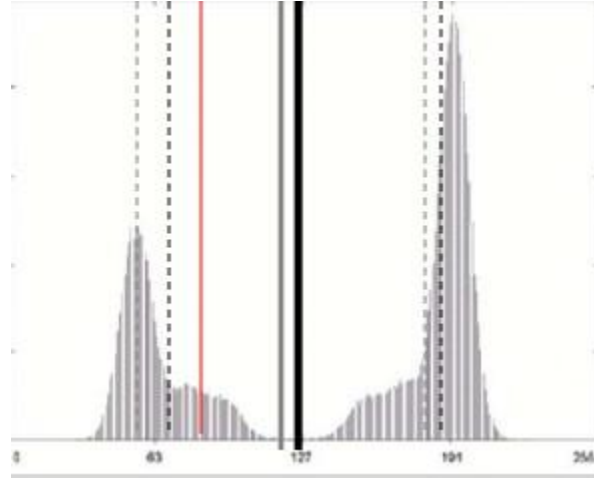


Figure 4.9 [27]

The figure 4.9 indicates the how the algorithm works.

The red line is the initial estimate. The grey dot line is the average values (round 1). The black dot line is the average values (round 2). The grey line is the threshold after round 1. The black line is the threshold after round 2.

A Gaussian mixture model is consist of M component single Gaussian model with a given weighted.

$$p(x|\theta) = \sum_{j=1}^k \alpha_j p(x|\theta_j)$$

Where $\theta = [\mu_1, \dots, \mu_n, \sigma_1, \dots, \sigma_n]$. The overall likelihood of observing x is a weighted mixture of Gaussians.

In statistics field, there is an expectation–maximization (EM) algorithm. It is using iterative way to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables [26]. The EM iteration alternates between performing an expectation (E) step, which

creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step[26] .

Here, our goal is to find blob parameters θ that maximize of the likelihood function:

$$p(data | \theta) = \prod_x p(x | \theta)$$

Expectation Maximization (EM) is divided into two steps. The first step is E-step which given current guess of the blobs, compute ownership of each point. The second step is M-step which given ownership probabilities, update blobs to maximize likelihood function. Then repeat this two-step until convergence.

Expectation Maximization (EM) Details

E-step

Compute probability that point x is in blob j , given current guess of θ

$$p(j | x, \mu_j, \sigma_j) = \frac{\alpha_j p(x | \mu_j, \sigma_j)}{\sum_{i=1}^k \alpha_i p(x | \mu_i, \sigma_i)}$$

M-step

Compute probability that blob j is selected

$$\alpha_j^{new} = \frac{1}{N} \sum_{i=1}^N p(j | x_i, \mu_j, \sigma_j) \quad (\text{N data points})$$

Mean of blob j

$$\mu_j^{new} = \frac{\sum_{i=1}^N x_i p(j | x_i, \mu_j, \sigma_j)}{\sum_{i=1}^N p(j | x_i, \mu_j, \sigma_j)}$$

Covariance of blob j

$$\sigma_j^{2new} = \frac{\sum_{i=1}^N (x_i - \mu_j^{new})(x_i - \mu_j^{new})^2 p(j | x_i, \mu_j, \sigma_j)}{\sum_{i=1}^N p(j | x_i, \mu_j, \sigma_j)}$$

Result & Data analysis

This section will show the detail of results and analyze the results with previous methods. Following the results of some analyses further questions are posed and additional analyses are carried out. All the analyses are based on the implement of the methods which mentioned in previous section. This section I will put more emphasis on the reason why the result is look like that and I will also analyze the important factors which have significant effect on the result of the experiment.

The 3D point cloud

Figure 5.1 shows the result by using the Kinect to detect my room with 3D point cloud; the white part in the figure is the object with reflective surface.

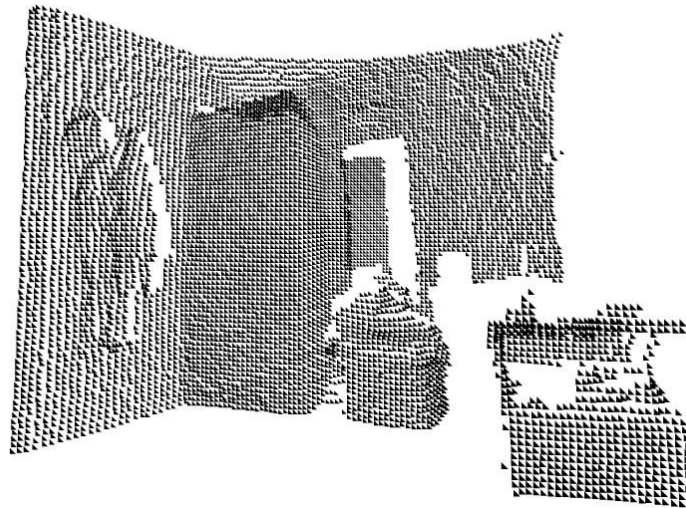


Figure 5.1 3D point cloud of my room

The figure 5.1 describes the feature of my room. The 3D point cloud can describe the depth of the image. Each point in my result with 3 dimensional coordinates. Here we set each coordinates of point as x , y , offset- z . The reason why I use offset- z to replace z is here we just want to describe the space feature of my room. And we do not need to get the more accurate depth value. Therefore, using offset- z just to zoom in or zoom out points can be better described the world, although it cannot use as navigation but it can be used to describe the world.

Landmark detection with depth information of then image

This analysis is based on the method which has been mentioned on section 4.1 I use 3 screen-shots via the Kinect to show the result of landmark detection based on the different threshold. The result is showed in figure 5.1

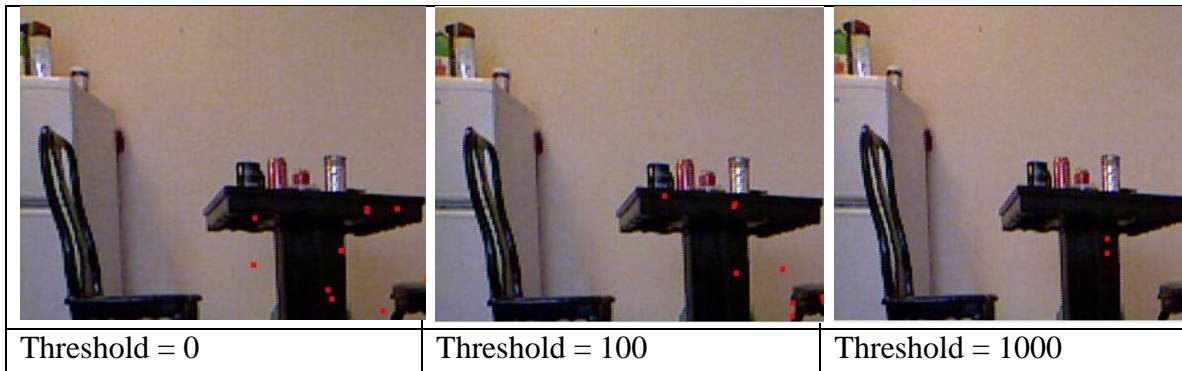


Figure 5.2 the landmark detection with 3 different threshold and Kinect resolution is 320 x 240 and the process pixel from X: 160 to 320 Y: 120 to 240 (the right bottom part of the image)

We can see the result from figure 5.1. First, I set the resolution of Kinect to 320 x 240, because the reason is that Kinect detect the world with 30 frames per second. Therefore, detection algorithm will execute once in 1/30 second. The draw point method will also execute in 1/30 second. It costs a long time on each frame if the resolution is too large (640 x 480 or more). What is more, frame may lost when Kinect make detection.

From the figure 5.1 we can also see that with different threshold, the landmark (red dot) will be decreased. If increasing the threshold, the number of bad landmarks will be decreased. We can see from the figure 5.1 that there are 2 bad landmarks on the image which threshold is 0, 1 bad landmark on the image which landmark is 100 and 0 bad landmarks in the image which landmark is 1000.

Bad landmark statistics

I randomly pick 10 frames among 30 frames in one second with different threshold and I statistic the number of good landmarks and the number of bad landmarks.

Threshold	0	100	1000
Good landmark	90	54	31
Bad landmark	55	23	6
Total landmark	145	77	37
Good percentage	62%	70%	84%

Table 5.1 the statistics of the landmark from 10 frames, the number of landmark is the sum of 10 frames (each threshold for 10 frames)

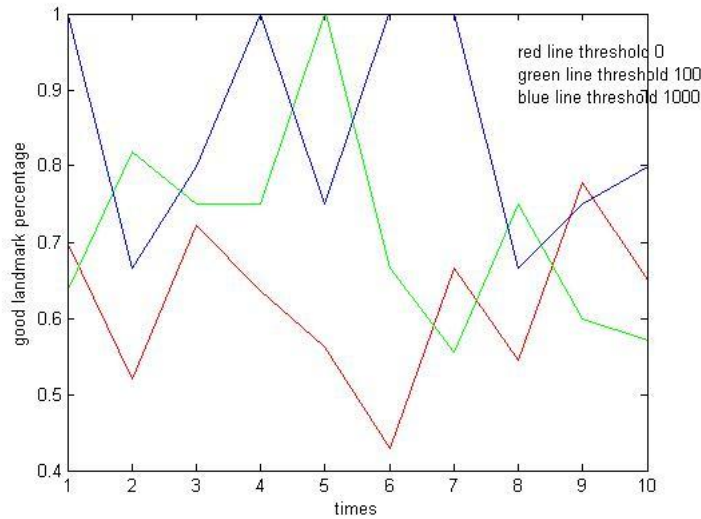


Figure 5.2

As is shown in the table 5.1, we can see that with the increase of threshold the number of good landmarks will increase. The more numbers of good landmark you get hold of, the less probability the robot is to lose way. But here the problem is why Kinect with my algorithm will produce bad landmarks?

These are two methods that may cause this problem. One is the program problem and the other is the problem with Kinect itself.

Program problem

When we code Kinect program, if we want to get the depth of the image, we must use the function, which is named `KinectSensor_DepthFrameReady`. In this function, we can get the depth Pixel Data which is a byte array. Which structure is indicated as follows:



Figure 5.3

Figure 5.3 shows the structure of depth pixel array (Microsoft Kinect Sdk 1.5 Document). It is easy to get the distance of each pixel with some bit operating. The problem here is that each second program will run this function 30 times (need to generate 30 frames per second). With too many operations running on this function (like bit operating to get depth of each pixel), the frame may lose data. Each time the function only executes 1/30s. If all the operations have not done in 1/30s, the `kinectSensor_DepthFrameReady` function will not wait but start again to detect new frame. Therefore, some landmark information will stay on memory and affect next frames data. And the bad landmarks will come out.

Kinect problem

It has been mentioned in previous section that if Kinect detect some objects with sharp or reflective surface the depth of the pixel on the object surface will be -1. Therefore, there is possibility that the pixel around the reflective surface becomes the bad point.

How to solve

We cannot change the hardware of the Kinect, so we try to solve it in software way. We try to use another thread to handle the function `kinectSensor_DepthFrameReady`. The landmark algorithm is processed in this thread. Our target is to confirm the completion of each frame. Even that may reduce the FPS.

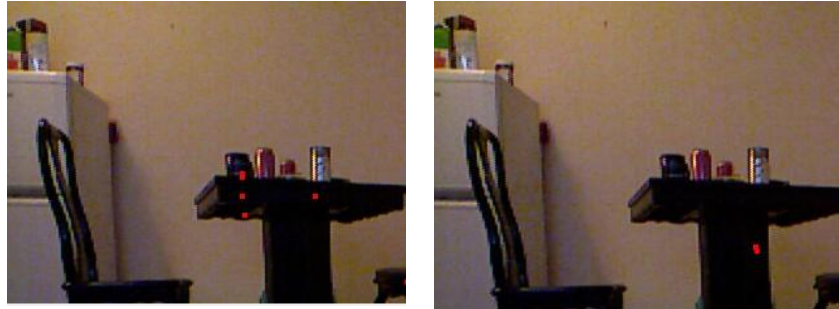


Figure 5.4

Compare with MRPT 3D visual SLAM

Comparing with the MRPT 3D visual SLAM, my result will have fewer landmarks than MRPT's result. Therefore, with fewer landmarks I could easy to make localization.



Figure 5.5 the comparing with MRPT 3D visual SLAM

The fewer the landmarks robot gets hold of, the harder the robot makes localization. Unfortunately, visual landmark detection will be easily affected by the environment. Thus, there is the possibility that different landmarks the robot detects are actually in the same position. This will affect the robot to make localization.

Convert the detect landmark to the 2D map view

The real view from Kinect	The 2D feature to describe the image left
---------------------------	---

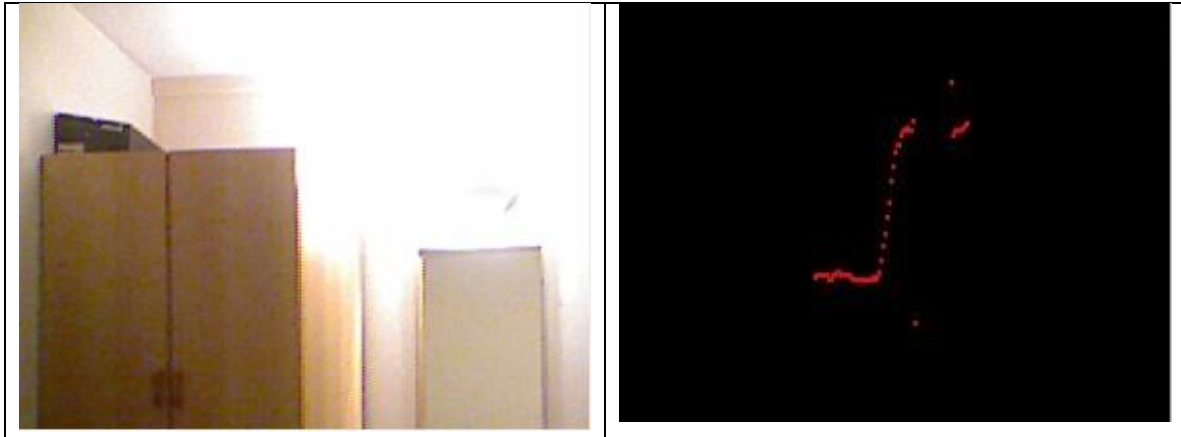


Figure 5.6

The figure 5.6 shows the 2D feature to describe my room. But it can be seen that there are several bad point in the right image. This is showed as follows:

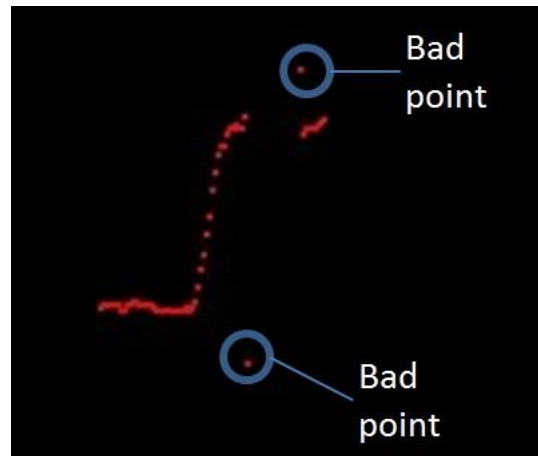


Figure 5.7

There are two reasons. The one is the reflective surface is detected by the Kinect. We can see that, in the left image in the figure 5.7, there is a mirror in the center of the view. It is hard for Kinect to get depth value in front of the mirror. Therefore, when I convert it to the 2d map view in figure 5.6, the red dots does not represent as the mirror. The bad point of which the depth is extremely small (bigger than -1) may be detected as the landmarks. It will not affect the final result, because the percentage of the bad point is small.

Interpreting Kinect Data

For the interpretation of the Kinect data, both the detection of the human body and object with a reflective surface are successful. Figure 5.8 shows an example of manipulated depth data via the Kinect. In figure 5.8, color represents the depth and white represents removed data. As it can be seen, both the human in the scene and the two reflective windows in the background have been detected and removed.

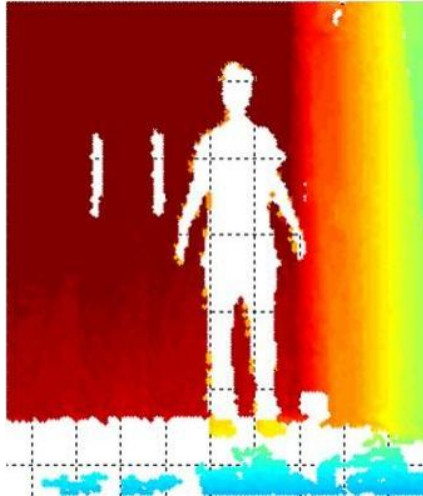


Figure 5.8 the graph illustrating the ability to remove humans and reflective surfaces from a scene. The color represents the depth.

Therefore, the dynamic object can be detected by this way. And the noise can be removed from the environment.

Accessible area

Average distance calculation method

When calculating the distance between the camera and its face objects we only need to calculate the average distance of the depth of each pixel in the image.

By this way, the time complex is $n(o)$. It is extremely fast for the Kinect to make detection. The result is showed as follows:

<p>Kinect detect distance is 303mm the real distance is 500mm</p>	<p>Kinect detect the distance is -0.42mm the real distance is 300mm</p>

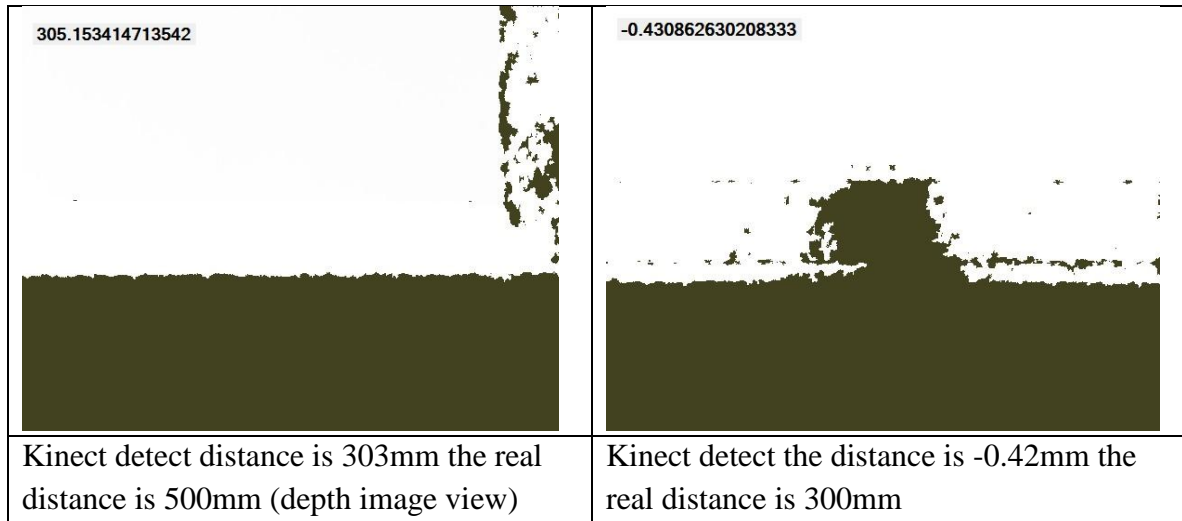


Figure 5.9

With regard to the depth image, figure 5.8 shows clearly that the color of the bottom part of the image is dark, which means that it is too closed to the object, or the object is with a sharp or reflective surface. Obviously, the dark part is the floor which is the accessible area. But Kinect thinks it is an object, and it is too closed to the Kinect. Therefore, if we set the safe distance for the robot is 200mm, when robot is equipped with Kinect and it is standing 300mm in front of the wall. The robot will turn left or right, because it is less than the safe distance. The solution is that: Do not let Kinect detect the bottom part of the world, in order to increase the accuracy of result. But it is not good for the robot to detect the world, because it will lose half of the information, taking into consideration that it only detects the top part of the image.

The advantage of average distance calculation method

The advantage of this algorithm is that it only makes the algorithm $O(n)$. When Kinect detects the real world, there are 30 frames per second. Therefore, each frame needs to process more than 76,800 pixels if the resolution is 320 x 240. The time complexity of this algorithm is only $O(n)$. Therefore, when using this algorithm to process over 70,000 pixels, it will not cause the loss of frame.

The disadvantage of average distance calculation method

Taking into account that Kinect is the detection device for the robot navigation, so we must consider that when Kinect detect some objects with a reflective surface, the special depth value will return to -1. Providing that if there is a big mirror in front of the Kinect, even far away from the Kinect, the depth value on the surface of monitor returns to -1. Therefore, if any objects with reflective object in front of the Kinect, this method will be inaccurate.

Image Segmentation Method

Mixture of Gaussians

This analysis is based on the method which has been mentioned on section 4.1, and Gaussian model is used to train the training-se. Thus, the image is divided into different parts in order to let robot know which part can be accessible. The navigation is more intelligence. First, the training I select is as follows:

		
Train as obstacles	Train as accessible area	Train as wall

Table 5.2

Then we transform this 3 image into greyscale model.

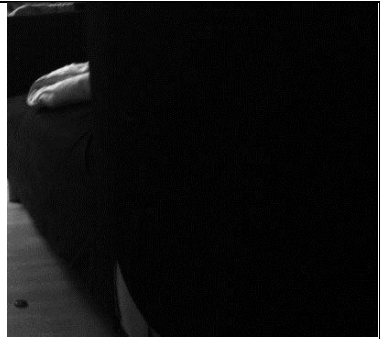


		
Train as obstacles	Train as accessible area	Train as wall

Table 5.3

The result shows as follows:



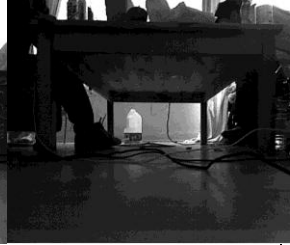
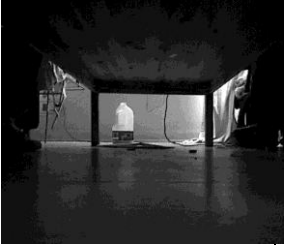


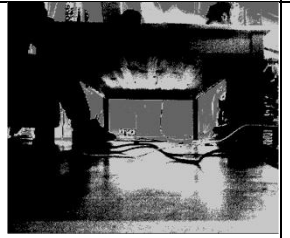

			
Kinect 1000mm from the table	Kinect 750mm from the table	Kinect 500mm from the table	Kinect 250mm from the table
			
Segment image which Kinect 1000mm from table	Segment image which Kinect 750mm from table	Segment image which Kinect 500mm from table	Segment image which Kinect 250mm from table

Table 5.4

As I have method above, I use three training-sets to train the image. Each training-set represents three regions (obstacles, accessible area and wall). In the Table 5.4, it can be seen that the black region represents the obstacles, the grey region represents the wall (the border of the map) and the light grey represents the area where robot can move into.

In detail,

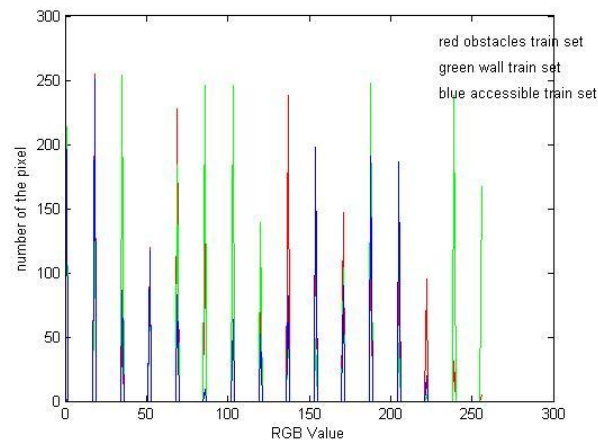


Figure 5.10 histograms for three training-sets

Clearly, three training-classes cannot represent three classification regions. We can see that pixel in 3 training-sets are similar. There is little coverage of pixel 3 training-sets. (The value of most pixels is about 23,40,51,65,100 ... pixel distribution is discrete) Therefore, our target is to find a suitable training-set which contains more RGB value information. Thus, we will redefine the training-set.

The new training set



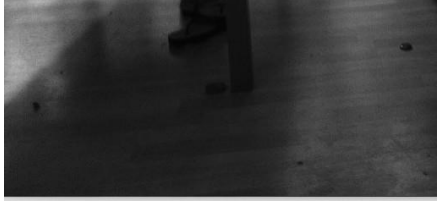
		
Train as obstacles	Train as accessible area	Train as wall

Table 5.3

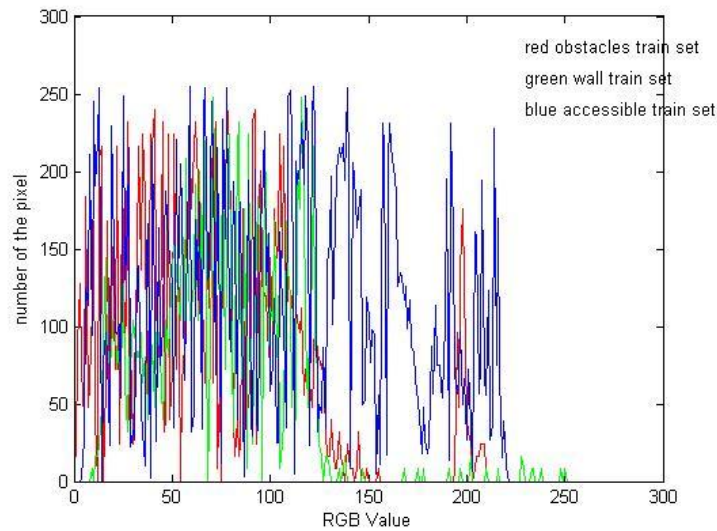


Figure 5.11

The figure 5.11 indicates the histogram of new training-set. We can see that the histogram of three training-sets has many intersections in this time. And all of the three training-sets have more color region than before, which means, in each training images, there are more pixels with different colors. (See the red blue and green line: The range of the three training image is bigger, and the range of RGB value of three training-set is about 0 to 220). The good method for the histogram of the training-set with many intersections is that the more detail of the image can be segmented with more intersections. The disadvantage of the training-set is that the segmented image cannot have a large region for each class. Therefore, small separate region will distribute in the

every part of the image. Each frame with high resolution will be segmented with exceptions.

The result of new segment method with new training-set



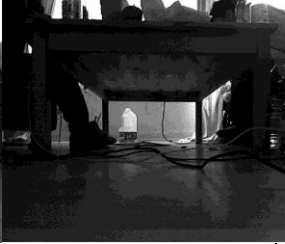
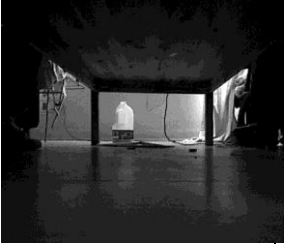



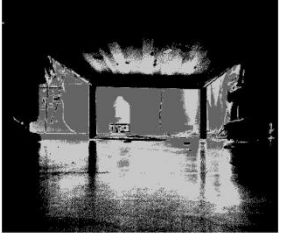
			
Kinect 1000mm from the table	Kinect 750mm from the table	Kinect 500mm from the table	Kinect 250mm from the table
			
Segment image which Kinect 1000mm from table	Segment image which Kinect 750mm from table	Segment image which Kinect 500mm from table	Segment image which Kinect 250mm from table

Table 5.4

Now I use Photoshop to define three different areas in the original image as the ground truth. It can simply reflect whether or not the results of the training-set are acceptable or not.

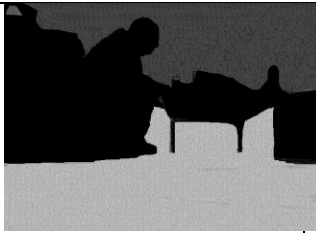
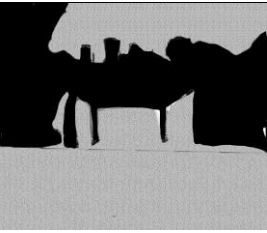
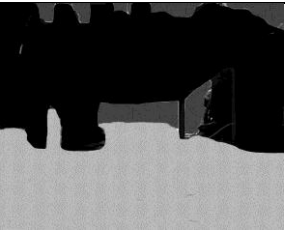
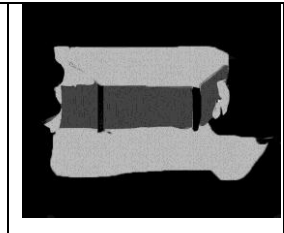
			
Kinect 1000mm from the table	Kinect 750mm from the table	Kinect 500mm from the table	Kinect 250mm from the table

Table 5.5

The different rate of segment image from ground truth

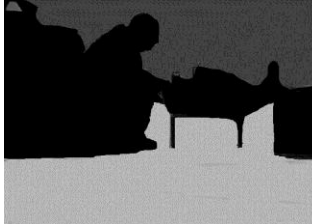
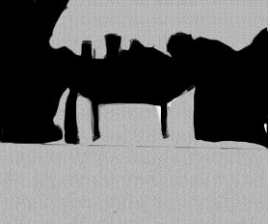
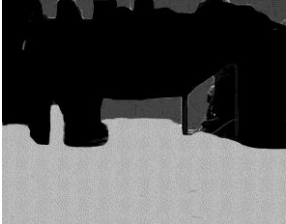
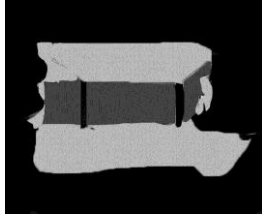


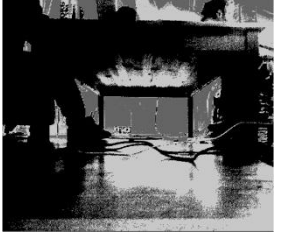
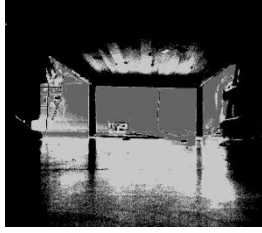
			
			
Difference 65.7%	Difference 70.0%	Difference 61.1%	Difference 49.5%

Table 5.6

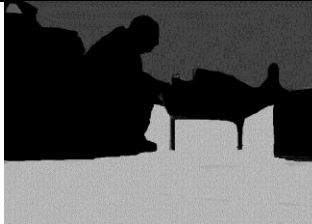
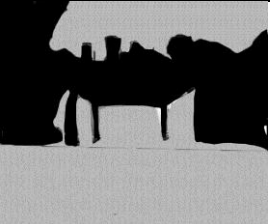
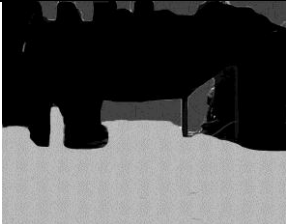
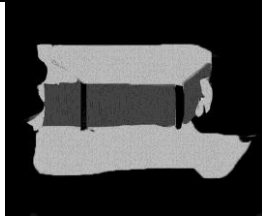


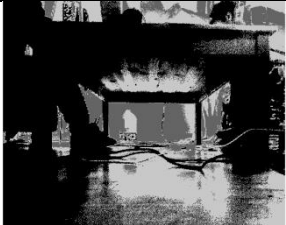
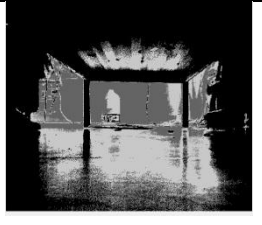
			
			
Difference 61.2%	Difference 65.7%	Difference 58.6%	Difference 48.1.5%

Table 5.7

The results show that the new training-set with a better segment. The ground truth is made by me via Photoshop, and I divide the image into three parts: accessible area (RGB value (200,200,200)), wall (RGB value (100,100,100)) and obstacles (RGB value (0, 0, 0)). The ground truth may be not accurate, but all the results are compared with this ground truth. Therefore, it can be concluded that more complex the training set is, the better result the segment gets.

Compare with two ways to generate the Mu

Method	Average way		Threshold way	
parameter	Mu	sigma	Mu	sigma
Training set 1	11.11107	22.556	15.196	22.91
Training set 2	108.2956	61.398	150.31	74.39
Training set 3	71.314	51.575	81.64	52.50

Table 5.8



Average way	Threshold way
	
Difference 65.7%	Difference 62.4%

Table 5.9 the segmentation result with different Mu generation.

We can see in the table 5.9 that these two methods generate different values of Mu and sigma. It is clear that the result that the Mu generates in the average way is smaller than that of the Threshold way. The result by Sigma is also smaller than the Threshold way. A low sigma indicates that the pixels in the training-set tend to be extremely closed to the Mu; whereas high sigma indicates that the pixels in the training set are spread out over a large range of values. It does not mean that Average way is better than Threshold way. For the simple image in frames, when the contrast are extremely high between background and foreground of the objects, it is better to use the average way to calculate Mu and sigma than that in the Threshold way.



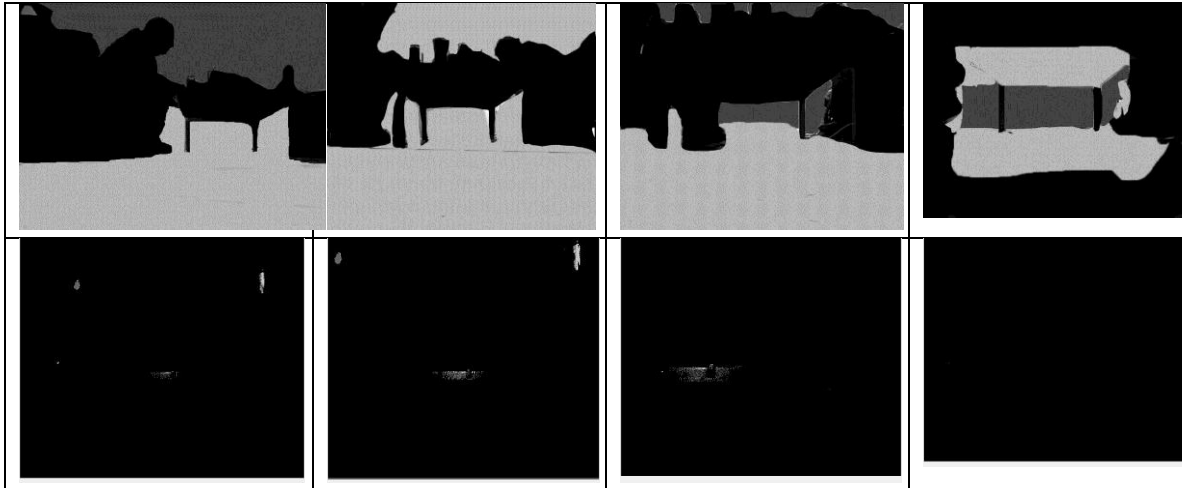
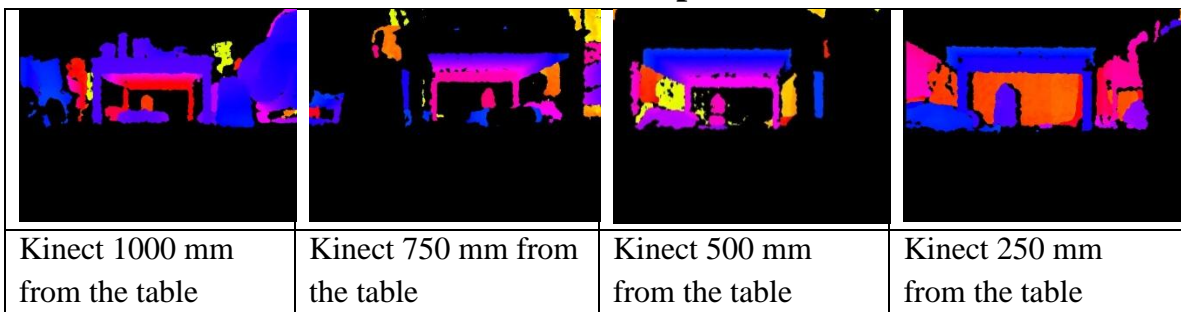


Figure 5.12

Figure 5.12 shows the results of image segment which are from the same training set in dark condition. The experiments are mainly practiced in the afternoon. And, at night, I just light lamp and get four frames in the same position as the ground truth. First we transform the images into grey scale, which is the first row in figure 5.7. However, the result is terrible. We almost see nothing, even with human eyes. Then we set the black area is the place where have obstacles. Therefore, the result (third row in figure 5.12) shows that robot cannot detect anything as well. It is regard as an obstacle and unable to move. Therefore, light is a big effect on the image segment with Gaussians. To be specific, if the selected training-set is on the different condition with the current frame, such as brightness, the result of segment of the current frame is terrible and hard to discern anything.

Combine the Gaussians model and the depth information of frame



The result shows the situation where the object is closer to the Kinect (the darker part of image). The result shows that the bottom of the image is always black, which is the floor, because the view of the Kinect has dark angles. Therefore, the floor, closed to the Kinect, may be detected as the object or the inaccessible area.

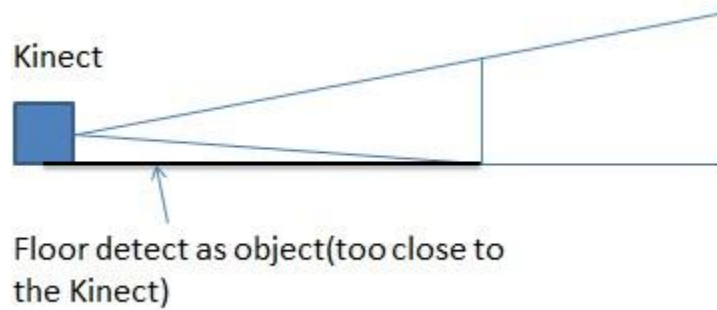


Figure 5.13

How to solve it? First, the easiest way is to increase physical height of the camera; therefore, the floor closed to the Kinect will not appear in the view field of the Kinect. Secondly, we only detect the top part of the frame, but the bottom part is accessible as default.

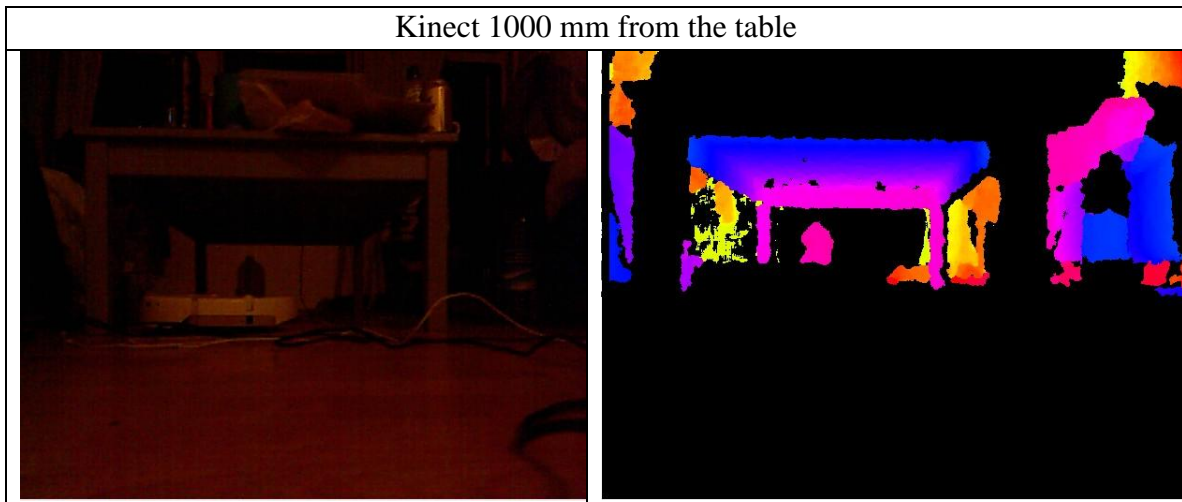


Figure 5.14 the left is the real view from Kinect and the right image is the color depth (with Gaussian) view

And the histogram of the image above is showed as follows:

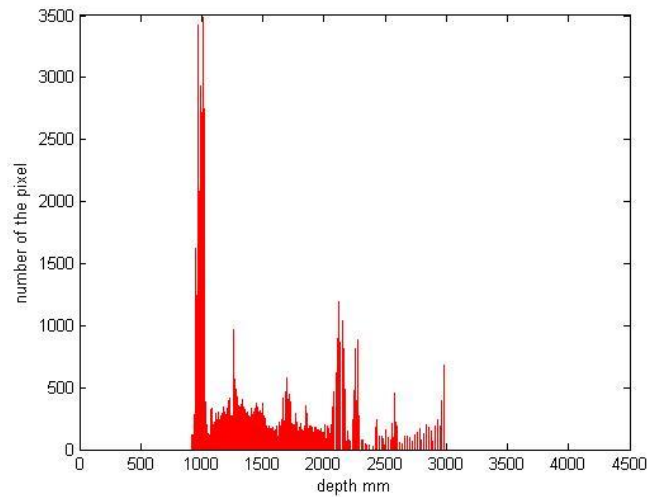


Figure 5.15

The figure 5.15 is the histogram of the image in figure 5.14. The x label is the depth of image; I set the depth range from 0mm to 4500mm. It is clear that most of pixel of which the depth is with the range between 1000mm and 3000 mm. Most parameters of depth are around 1000mm (mainly on the desk). We can see the first significant peak in histogram which represents the black part in the right image in figure 5.14. The middle part of the histogram represents the blue and purple part in the right image in figure 5.14. And the second significant peak and the rest part in histogram represent the yellow and orange part in the right image in figure 5.14.

The EKF-SLAM

The EKF-SLAM algorithm is very hard for me. I try to write the program to implement it, but my software still has some bugs. Therefore, I will analyze the results with the program which is not written by me. The program is named DiversityGridSlam. It is not an opensource software. But I use the .net reflector to decompile it and get the source code (maybe is not same as the author's code) but I decompile the code and run it to analysis the result.

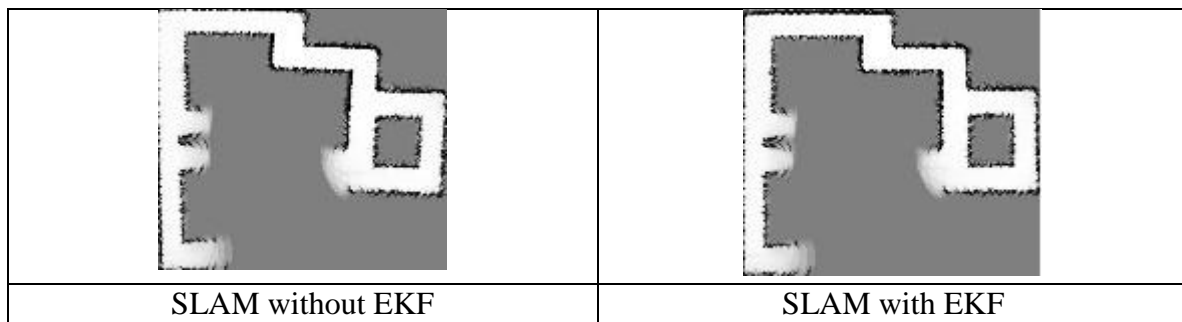


Figure 5.16

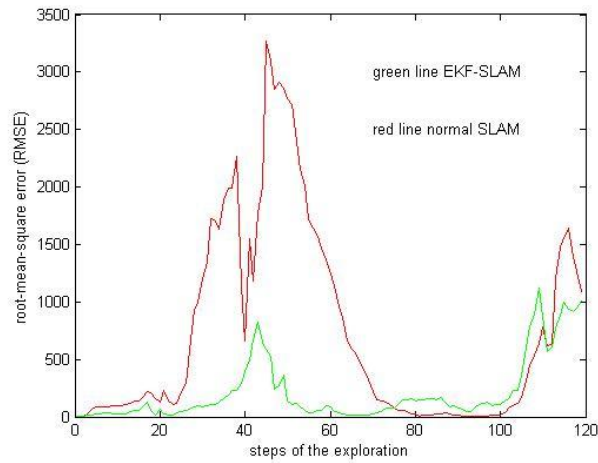


Figure 5.17 the RMSE on SLAM with different methods.

The figure 5.8 indicates the result between EKF-SLAM and normal SLAM. It is clearly that the EKF-SLAM has a better map than that by the normal SLAM. The figure 5.9 indicates the root means square error between the EKF-SLAM and normal SLAM.

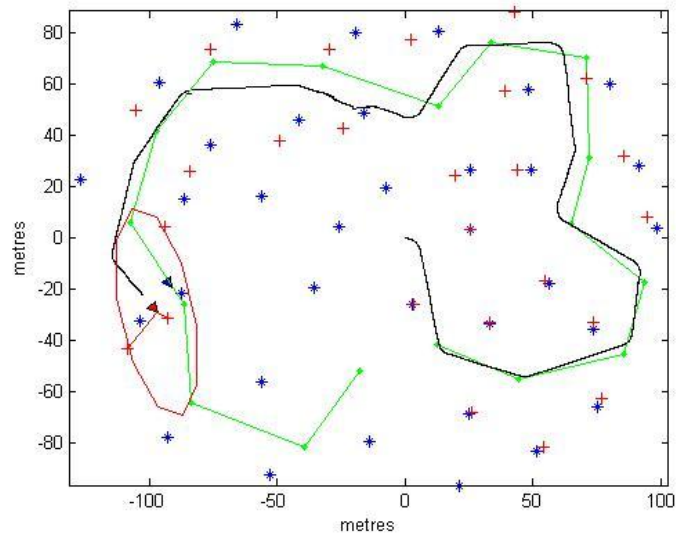


Figure 5.18

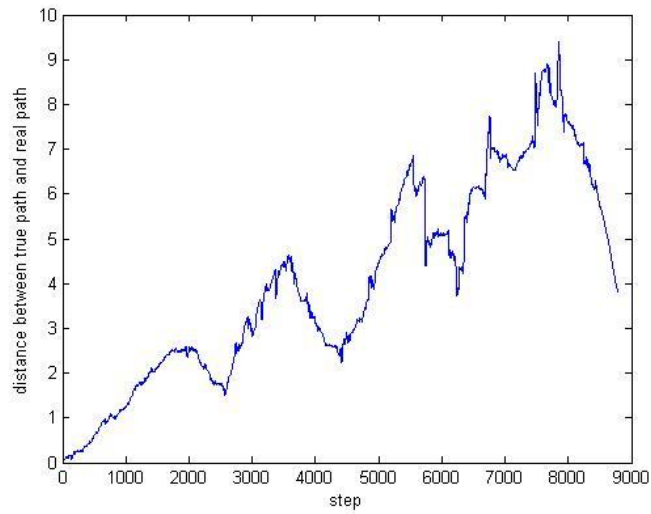


Figure 5.19 EKF-SLAM with 1 round

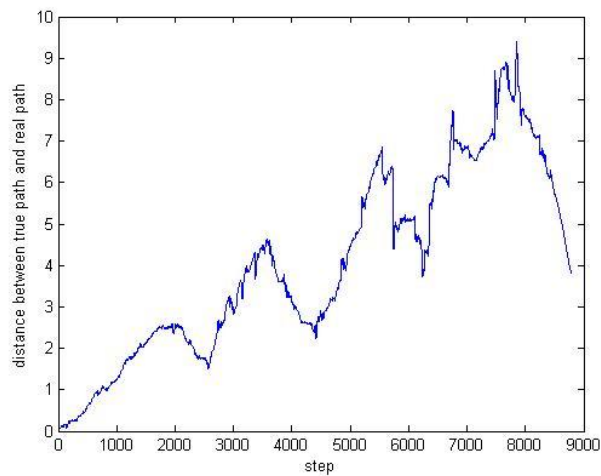


Figure 5.20 EKF-SLAM with 2 round

Figure 5.18 is the EKF-SLAM. The blue dots in the image are the object, and the red dots are the detected landmarks, the green line is the true path regarded to be the ground truth, and the black line is the real path. Figure 5.19 and 5.20 indicate the distance between the ground truth and the real path, as the distance becomes larger. It is easy to understand that if the robot keeps moving, the noise will be increased step by step. Therefore, the distance between ground truth and the real path is increased.

Discussion

The landmark detection

There are several features of the algorithm above. All the pixels are iterated only once. Therefore, the time complexity of algorithm is $O(n)$. This algorithm is improved, compared with other common landmark extraction methods which often run $O(n \log(n))$ [22]. The bad data are ignored, but if there are two points laying on each side of a section of bad data, they will be still treated as neighbors. The reason why I use this precaution is that the Kinect often regard the sharp and reflective surfaces as bad data. These bad data points are likely to be the results of the IR lasers hitting the surface at a glancing angle significantly distorting the Kinect's laser array making it un-interpretable [20]. These objects with sharp and reflective surfaces are exactly the type of surface which the algorithm attempts to detect. Unfortunately, many landmarks are undetected if ignoring of these points. In addition, this algorithm will ignore the landmarks which are far away from the Kinect, even though the landmarks are neighboring pixels. This review is to ensure that the landmark detected is on the confidence which is 1220mm to 3810mm (*Microsoft Kinect SDK 1.5 Document*).

Comparing with the result of MRPT, my algorithm generates fewer landmarks. The advantage of fewer landmarks is that it is easier to process landmarks, and to get the feature of the landmark. And also the complex of my algorithm is $o(n)$, it will process faster than that of other algorithm like $O(n \log(n))$ at the quickest [22] without the loss of the frame. Therefore, my algorithm will have a good performance when the resolution of the camera is high.

But there are also some disadvantages of my algorithm. The fewer landmarks have been detected, the harder the robot makes localization. Unfortunately, visual landmark detection will be easily effected by the environment Therefore, there is the possibility that different landmarks the robot detects are actually in the same position. Thus, this will affect the robot to make localization. Robot can easily fix the position with more detected landmark. My algorithm will be weak in this part. Secondly, I add the color information to detect the landmark. Kinect uses IR to detect the depth of the image, and it will not be affected by the environment. The color information highly depends on the condition of the environment. When the environment is getting darker, it will be harder for Kinect to detect the object with color camera. Therefore, my algorithm will have different performances in different conditions.

The Gaussian model

When I get the landmark, I will tell the robot which part is accessible. I try to use Gaussian model and Bayesian inference methods to segment each image of the frames. And I try to tell the robot to which part the robot move.

Gaussian model also has the advantage and disadvantage for the robot to detect the directions. The advantage is that we have to get a rule for robot how to classify the pixel in the frame each time before we choose the training-set for robot, because different pixels are classified to different classes. Therefore, based on the classification, robot will move to the accessible class, in the frames which means this part can be accessed. It is true that classification is highly dependent on the training result. In other words, it is highly depends on the training-set. Therefore, how to choose the training set is the key factor which affects the final accurate. In addition, Gaussian model processes the training-set with its RGB information. Therefore, the condition of the environment (e.g sun light) will have a large effect on the training-set. For example, if we train the image in the morning, and we use this rule to get classify of each frame in the night. The result will be inaccurate. Therefore, when using Gaussian model to classify the accessible area we must confirm that the conditions of the selected training-set, which is similar with that the robot needs to detect.

Using SGM and Bayesian inference methods are suitable for the segment the images, especially when the contrast of the image is high. But they also have some disadvantages. If the contrast of the image is not too high, and the histogram of the image has many peaks, it is hard for SGM to detect the foreground and background. So the accuracy will be decreased. Hence, if the image is too complex, GMM is the better choice to segment the image than SGM. Location prior is also an excellent way to segment the complex image. Location prior is not too general, but more specific. The segmentation of some specific images are excellent. This special location prior is hard to be used on other common images. It is hard to say which one is the best method for the image segmentation for the robot to choose accessible area. Different images are processed by different methods. Some images are hard to segment, even segment by human beings. Therefore, the combination of different methods will be a better choice for processing the image.

EKF-SLAM

SLAM can be solved by practicing these three steps iteratively. EKF-SLAM provides a system to solve the problem. And much work has been done on its convergence, the growth of map, the changes of uncertainty and etc. However, EKF-SLAM still has many problems in practice. First, the linearization of nonlinear function may result in errors in the system. The second problem is the Gaussian Model for noises and the irrelevant hypothesis are often unacceptable. Both of these two factors may cause the uncertain operation of EKF-SLAM. In the meanwhile, the algorithm depends on the correct motion model and observe model, so it is critical to model the robot and the environment. EKF-SLAM is usually suggested avoiding using for those applications where it is hard to model the robot and the environment.

Future work

As to work in the future, I am interested in the extension of my method to the large environment. My method highly depends on the environmental condition such as brightness of the environment, so the indoor area is more suitable for my work. Besides, I will try to fix the bug of EKF-SLAM software and to combine it with my landmark detection algorithm and accessible area detection algorithm.

Conclusion

In this paper, I try to implement landmark detection algorithm which is based on the depth image and to improve it with color information. Compared with other common landmark extraction methods which often run at $O(n \log(n))$, my algorithm is run at $O(n)$. The advantage of my algorithm is the time complexity. The higher resolution the camera has, the less possibility the frame has to loss by my algorithm. But my algorithm still has some shortages. If the threshold value is too small, too many bad points will be detected as the landmarks which will seriously affect the accuracy of robot localization. If the threshold is set to a big value, it is hard to detect the bad points. The number of the detected landmark will be decreased, so it is easier and faster for fewer landmark points to process. But the accuracy will be affected as well. And for using image segmentation method to detect the accessible area for the robot to choose the direction, I use Gaussian model to train the training-set. Unlikely, the fact is that the classification of Gaussian model is mostly affected by the training-set. Therefore, if the conditions which Kinect detects are different from that of the training set (like sunlight), there will be a significant effect on the result accuracy. And, GM is based on the RGB information of each pixel, so the change color on the object's surface will directly affect the classification results.

Acknowledgements

I would like to express my thanks to my supervisor Walterio Mayol-Cuevas, he gives me a lot of help on the project.

Thanks also to Xiaochuan Wu and Wei Wang for their help on my project and my English writing.

I would like to greatly thank my family and my girlfriend for their constant support and encouragement.

Reference

- [1] Wikipedia/Date [2012-8-17]. http://en.wikipedia.org/wiki/Mobile_robot
- [2] Frese U. A Discussion of Simultaneous Localization and Mapping. *Auton. Robots.* 2006, 20(1): 25-42.
- [3] Xiucan JI, Hui ZHANG, Dan HAI, and Zhiqiang ZHENG. An Incremental SLAM Algorithm with Inter-calibration between State Estimation and Data Association. RoboCup International Symposium 2008, July 15-18, Shuzhou, China. (Accepted and will be published in Springer LNCS/LNAI series)
- [4] Xiucan JI, Hui ZHANG, Dan HAI, and Zhiqiang ZHENG. A Decision-theoretic Active Loop Closing Approach to Autonomous Robot Exploration and Mapping. RoboCup International Symposium 2008, July 15-18, Shuzhou, China. (Accepted and will be published in Springer LNCS/LNAI series)
- [5] Frese U. A Discussion of Simultaneous Localization and Mapping. *Auton. Robots.* 2006, 20(1): 25-42.
- [6] Ray J. Distance Transform Based Path Planning for Robot Navigation. *Recent Trends in Mobile Robots*, Zheng Y, World Scientific, 1993, 3-31.
- [7]] Wikipedia/Date [2012-8-18].
http://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping
- [8] Frese U. A discussion of simultaneous localization and mapping. *Autonomous Robots.* 2006, 20(1): 25-42.
- [9] Hahnel D, Thrun S, Wegbreit B, et al. Towards Lazy Data Association in SLAM. *Robotics Research.* 2005: 421-431.
- [10] Neira Jos é Tard ós Juan Domingo. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation.* 2001, 17(6): 890-897.
- [11] Besl, P. J, McKay, N. D. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 1992, 14(2): 239-256.
- [12] Lu F, Milios E. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems.* 1997, 18: 249-275.

- [13] Thomas R. Using Histogram Correlation to Create Consistent Laser Scan Maps. Proceedings of the IEEE International Conference on Robotics Systems. EPFL, Lausanne, Seitzerland: IEEE, 2002. 625-630.
- [14] Biber P, Straer W. The Normal Distributions Transform: A New Approach to Laser Scan Matching. Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway, NJ: IEEE, 2003. 2743-2748.
- [15] JI Xiucai. Data Association problem for simultaneous Localization and Mapping of Mobile Robots. June , 2008
- [16] Andrea C, Gian, Diego T. Lazy localization using a frozen-time filter/Date [2007-5-24]. <http://purl.org/censi/research/2008-icra-fts.pdf>.
- [17] Monte Carlo Localization: Efficient Position Estimation of Mobile Robots," by Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun, *Proceeding of the 1999 National Conference on Artificial Intelligence (AAAI)*
- [18] Lu F, Milios E. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*. 1997, 4: 333-349.
- [19] Wikipedia/Date [2012-9-15]. http://en.wikipedia.org/wiki/Point_cloud.
- [20] Wikipedia/Date [2012-9-15]. <http://en.wikipedia.org/wiki/Kinect>
- [21] Richard Marron Advisor: Dr. Jason Janet C# Implementation of SLAM Using the Microsoft Kinect 4/18/2012.
- [22] V. Nguyen, S. Gachter, A. Martinelli, N. Tomatis and R. Siegwart, "A comparison of line extraction algorithms using 2D range data for indoor mobile robots," 8 June 2007.
- [23] Microsoft MSDN document for Kinect
- [24] Simulation localization and mapping with the extend Kalman filter, Joan sola August 26, 2012
- [25] M. Montemerlo and S. Thrun, FastSLAM - A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics, vol. 27 of STAR – Springer Tracts in Advanced Robotics. Berlin Heidelberg New York: Springer Verlag, 2007.
- [26] <http://www.bioss.ac.uk/people/chris/ch4.pdf> [2012-9-16].
- [27] <http://www.ece.uvic.ca/~aalbu/computer%20vision%202009/Lecture%2010.%20Segmentation-edge-based.pdf>
- [28] *Digital Image Processing*, Addison-Wesley Publishing Company, 1992, Chap. 7.

[29] E. Davies *Machine Vision: Theory, Algorithms and Practicalities*, Academic Press, 1990, Chap. 4.

[30] D. Vernon *Machine Vision*, Prentice-Hall, 1991, pp 49 - 51, 86 - 89.