Local navigation using features in images

SeungMin Baek

The Graduate School
School of Electrical and Electronic Engineering
Yonsei University

Local Navigation using Features in Images

A Masters Thesis
Submitted to the Department of
Electrical and Electronic Engineering
and the Graduate School of Yonsei University
in partial fulfillment of the
requirements for the degree of
Master of Engineering

SeungMin Baek

December 2015

This certifies that the masters thesis
of *SeungMin Baek* is approved.

Thesis Supervisor: DaeEun Kim

Sanghoon Lee

Euntai Kim

The Graduate School
School of Electrical and Electronic Engineering
Yonsei University
December 2015

Abstract

The navigation of mobile robot is challenging problem due to insufficient information of surrounding, error from controlling the mobile robot, and error from estimating its current location. The simultaneous localization and mapping problem is one of popular concept, and solution for problems of navigation task. This concept declare the localization problem, and mapping problem cannot be separated independently. To localize itself, it requires the map to express its current location while a mapping problem requires current location to build a map. With this map information of surrounding environment, and current location from localization problem, a navigation task can be achieved.

To achieve navigation task for mobile robot, it requires abundant information of an environment, and the most popular type is a visual information. This provide information, e.g., visual features, hue, depth, and extracted specific object. Some type of information can be easily extracted by simple filter, but some require sophisticated vision technique. This complex technique also requires complex computational load, and it induce the amount of time to extract meaningful visual information. To overcome this shortage, there is another approach which is based on navigation ability of animals, and insects.

There are many experiment on insects, and animals to find out their navigation ability and its characteristic. There are meaningful results with insects navigation. The insects has a navigation ability to find a direction to their living area with visual information after foraging. It is known that many insects including ants can use visual snapshot around them for homing navigation. Inspired by this navigation ability of insect, many navigation algorithms have been suggested. The average landmark vector (ALV) algorithm is one of famous algorithm to find a direction to the target location relative to the current location. This algorithm is based on the observation of landmarks from visual information. Observing and identifying landmarks in surrounding environment is a challenging problem. For the snapshot model, the feature extraction from the visual image plays an important role, and it also requires a technique to extract features.

There are many different ways to extract visual features, e.g., using SIFT, and SURF descriptors, vertical lines and its surrounding image, and any other feature extraction algorithm. We apply these different algorithm s to extract visual features on image, and use these features as landmarks in navigation task. With these information, we set

a target location arbitrary, and we show how well it can find a direction toward to target location relative to its current location with snapshots from both locations. With these algorithms, we implement them to a mobile robot, and control this robot with these algorithms.

These algorithm which are based on navigation ability of animals, and insects have one purpose that control a mobile robot to target location. This task is not enough to solve the SLAM problem. From this concept of the snapshot model, we suggest a SLAM algorithm to solve this problem. We depends on visual information, and distance information to solve these localization and mapping problem. With this algorithm, we show it can build map correctly, and show its performance have more advantages than other basic SLAM algorithm.

Acknowledgements

This thesis could not have been achieved without the support of many people. My first thank for my supervisor, the Prof DaeEun Kim, who always guide me. Prof Sanghoon Lee, and Prof Euntai Kim, the professors who volunteered to be my thesis committee. They gave me advices, and these bring enormous improvement of my thesis. I also show my gratitude to the members of the Biological Cybernetics Lab. Especially, I give thanks to Changmin Lee, and Eunseok Jung who are always beside me, and support anything they can. I also give my thanks to my technical supports, Youngseo Cha, Hyungu Yim, Joongbo Shin, Jaehong Lee, Wonjun Lee, Seungbae Son, Jaehyun So, Sanghun Lee, Junghun Chun, Wonki Lee, Seulgee Kim, Changmuk Kim, and Yunkyung Ju. Finally, I put my all thanks to my family who support me for my entire life.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*SeungMin Baek*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Some insects have navigation ability to return home after foraging. They have incredible navigation ability even they have simple neural structure, and limited information of surrounding environment. There are many different analysis of their navigation ability, and the most plausible hypothesis is the ”Snapshot hypothesis”. This hypothesis implies that they rely on their visual information to decide direction to their home by comparing visual information from their home, and current location.

From this hypothesis, there are many different derived algorithms to solve the navigation problem. There are two different approaches, one is using entire visual information to find direction to home, and another is extracting features. First one have relatively simple, but this algorithm can be affected by noise on image, and occlusion. Navigation algorithm using extracted features have more complex calculation to extract valid feature, but its performance can be more reliable. We suggest this feature based algorithm can be a solution for navigation problem, and we use many different type of features. There are many different features which have unique characteristic on image, We compare these feature extraction algorithms by comparing their performance.

This feature based algorithm provide proper direction to home relative to the current location. This estimating of a heading direction can be enough to achieve navigation problem, but some cases require more information of surrounding information to avoid obstacles, and plan proper route to reach a target location. This problem is same to the SLAM problem, and we suggest an algorithm to solve this problem.

1.1 A bio-inspired feature based navigation

There are many studies on navigation ability of insects to solve navigation problem. One of the interesting approach is navigate by comparing images from a current location, and a target location. This approach is originated from a study of movement of honeybees (Collett et al., 2006). The behaviour of bee when they are foraging have its own characteristic. Otherwise, there is another insect whose navigation ability is proven is desert ant (Andel and Wehner, 2004), Müller and Wehner (1988). This ant spread chemical trail to find out its way and navigate. They have certain chemical route to food, and use this path. Even they losing their way, they trying to find out this path to continue foraging (Kohler and Wehner, 2005).

There are several possible navigation to use of insect like bee. One is path integration. Insect know its direction and it movement by observing optical flow or difference of observed landmarks. By integrating its own movement, it knows it location related to their nest. In their retina, they use optical flow from ground they observe, and same thing from surrounding area. Another navigation method in bee is found (Collett, 2008). In this theory, bee rely on visual information while they combining this information with from sky, sun-compass. They can estimate their current direction by using visual information. They know how they rotate by observing optical flow. This combining of information allows them to navigate with more accuracy. They communicate by dancing to other companion with these navigation information. They found that the bee and ant not only use their visual information to navigate, but also can learn about landmarks. This concept of landmark is very important to following navigation algorithm (Graham and Collett, 2006).

From this concept of landmark of navigation ability of insects, there is a simple, but robust navigation algorithm, the average landmark vector algorithm (Möller, 2000) (Lambrinos et al., 2000). This algorithm have a concept of the average landmark vector. The landmark vector is defined by a unit vector from current location to observed landmark. Average of these landmark vector can represent characteristic of each position. The difference of average landmark vector for each position indicate home direction which is direction toward target location. This algorithm just consider only change of bearing which means this algorithm considers limited information of observed landmarks. This algorithm is valid and can indicates home direction with high precision. Thia algorithm also guarantee its convergence to target location.

On the other hand, there are different approach to understand navigation ability of animals which is study their brain activity to find unique characteristic during navigation task (Andersen et al., 2006). This approach based on brain activity of mammal, and it shows there is close relationship between activity of specific part of their brain, and its current location in environment. This specific part of their brain is the ”place cell” in their hippocampal cells (O'Keefe and Dostrovsky, 1971). This observation implies that these mammals remember characteristic from each location in environment, and rely on this information to find their current location, or localize themselves.

In the next section, we introduce detailed motivation and objectives for the models proposed in this paper.

1.2 Motivation and objectives

We suggest a model from bio-inspired visual system with features in visual information. With this algorithm, we also show this algorithm can be a solution for navigation problem. Furthermore, We show our algorithm can be a solution for the SLAM problem using features. The detailed objectives are as follows:

Navigation with landmark vector algorithm based on feature extraction There are many bio-inspired approach to solve the navigation problem with the snapshot model. From this approach, we suggest a feature based navigation algorithm. This algorithm is based on the average landmark vector(ALV) algorithm(Lambrinos et al., 2000). This algorithm require a specific landmark to estimate direction to target location relative to the current location. This extracting landmarks have to be robust, and we solve this problem with different feature extracting algorithm. The histogram of oriented gradients (HOG) descriptor, the scale invariant feature transform (SIFT) descriptor, and the speeded up robust feature(SURF) descriptor are most popular feature extraction algorithms. We show this algorithm can be a solution for landmark problem, and this algorithm can direct proper direction to target location by comparing snapshots.

Homing with a mobile robot, and route following We propose an algorithm to find proper direction to target location. We implant our algorithm into a mobile robot to move target location. In previous algorithm, we set only one target location to find direction, but we set serial waypoints to follow a route. We previously

set multiple waypoints to follow these waypoints. With snapshots from these waypoints, we can control a mobile robot along waypoints. We also propose another algorithm to follow waypoint in dynamic environment. This algorithm is based on the conditional probability to find its associated image. We have a learning system to remember images from these waypoints. We use current image as input of the system. With corresponding output of the system, it can find its current location, and navigate to target location.

Probabilistic approaches of local visual navigation There is another approach from bio-inspired navigation. This approach focus on solving the SLAM problem. The 'Rat SLAM' is one of them (Milford and Wyeth, 2010). This approach is from brain activity of brain of rats during navigation task. We simulate this algorithm with visual information. From this idea, we also propose another SLAM algorithm which can solve mapping problem, and localization problem. We use not only visual information, but also distance information from laser sensor. We conbine this information to find its current location, and map information of surrounding map. We compare this algorithm into the 'RAT SLAM', 'Laser only SLAM', and dead reckoning.

1.3 Organization of the dissertation

In chapter 1, we introduce the motivation, and history of our work from the concept of the bio-inspired approach. We also define the problem we have to solve from our works. The next chapter, chapter 2, we introduce related work with many different point of view. Bio-inspired navigation, dealing with visual information, traditionanl navigation, and the SLAM problem.

In chapter 3, we show bio-inspired navigation algorithms based on features. We show extracted features can be a landmarks for navigation problem, and this can be a proper solution for navigation problem. We also simulate this algorithm in static environment. We take snapshot images on given environment, and we show these algorithms can find proper direction to target location.

Chapter 4 presents control of a mobile robot with proposed algorithms. We show our algorithm can find target location. We set multiple waypoints on static environment, and we also take snapshot images on these waypoints. With these information, we

show how accurately a mobile robot can follows properly. In this chapter, we also propose another navigation algorithm to follow previously set waypoints. This algorithm is based on rememeberance of animal, and rely on the conditional probability. With this algorithm, we show this can be a solution for navigation problem to follow given multiple locations.

In chapter 5, we apply our algorithm into the SLAM problem. In previous research, we focus on navigation to target location. The localization, and mapping problem have to be solved to satisfy the SLAM algorithm. We propose an algorithm to solve these problem based on a concept of the feature of landmark. There is a similar approach which is based on bio-inspired algorithm which is called the 'RatSLAM'. We simulate this algorithm, and compare our algorithm.

The last chater have concolusions of our works. We show our importance of our works, and possibility to improve. We also conclude shortage, and advantage of our works.

Chapter 2

Background

Navigation technique is very complicated algorithm by unexpected scene (Brady and Wang, 1992) (Vasudevan et al., 2007). This complex thing is need to be simplified to apply in real problem (Kuipers and Byun, 1988). To prove this complicity problem, there is a new approach. This approach is inspired by insect navigation ability. Some insect can return home after foraging in a straight line (Wehner, 1987). These insect have no complex calculation capability at all. Even they have this limitation, they can return home successfully citepcollett1986landmark. Collett observe their behavior of navigation to find out its characteristic (Collett, 1996). In this observation, these insect use many navigation clues such as magnetic information of earth (Collett and Land, 1975), polarization of sky (Lambrinos et al., 1997), visual cue around insect (Wehner and Müller, 1985), celestial compass (Wehner, 1997), path integration (Wehner and Wehner, 1990) (Wehner and Wehner, 1986) and chemical trail for some insect. Another observation is that these insect like ant use landmark to find out home direction (Collett, 1992). Without any distinguishable object, they cannot find home easily (Graham and Collett, 2006). This concept of landmark is one of most important in local navigation algorithm.

Another similar observation, suggest snapshot hypothesis (Wehner and Räber, 1979). This explains the behavior of insect that is insect estimate home direction by comparing image from home location and current location. By comparing these images, insect can know home direction. From this hypothesis, many navigation technique is improvised by contribution of great researchers.

From previous work in this field (Mallot et al., 1996), one of the important work is done by Franz (Franz et al., 1998). He take these hypothesis and model this algo-

rithm (Franz et al., 1997). He suggest how the landmark is observed in both location by modeling it. This model is fundamental of navigation algorithm based on snapshot hypothesis. This also is define problem of this vision based algorithm that is lack of distance information. He also suggest how can we solve this problem from work about motion of robot (Nelson and Aloimonos, 1988). Most of later work is to solve this problems. He also conduct navigation experiment with real robot in artificial environment. With conical camera, he observe surrounding area (Srinivasan et al., 1997) (Yagi and Yachida, 1991). He prove this model is valid and its performance is good, and can be applied in on-line situation. By this means, Franz open new era of local visual navigation field.

With this model, Moller R. and D. Lambrinos suggest another way to find home direction (Möller et al., 1998) based on snapshot hypothesis. He uses vertical edge to distinguish landmark (Möller et al., 1999). With this detected landmark, he suggest two new algorithm one is the proportional vector model, and another is the average landmark model.. One is draw vectors toward observed landmark from center of agent, and difference of sum of these vector at both location direct home direction. In this algorithm, the magnitude of vector is proportional to observed size difference of landmark in both location. The average landmark vector algorithm is same but this draw unit vectors. He suggests these algorithms to find out home direction. He also prove its validity, and perform experiment with real robot. He also suggests using another information like polarization in sky is also another way to find home direction.

Another approach of navigation is the simultaneous localization and mapping(SLAM) (Betke and Gurvits, 1997) (Booij et al., 2009) (Booij et al., 2008). In this problem, the localization is one of the most important issue (Cozman and Krotkov, 1995). This problem is knowing its current position relate to original position of panoramic image (Gonzalez-Barbosa and Lacroix, 2002). This problem is same in local navigation problem (Burke and Vardy, 2006). The robot need to remain constant orientation (Collett and Baron, 1994). In this problem, position information include angle data related to home location. In traditional way to find this angle information is using magnetic sensor or odometer information (Barshan and Durrant-Whyte, 1995). This two method is useful, but vulnerable to external noise. To improve this shortage, Labrosse (Labrosse, 2004) suggest one way to find out this angle information without using magnetic sensor (Frier et al., 1996) or odometer information (Nistér et al., 2006) (Chen, 2004). This algorithm called visual compass that is just using visual information to find angle difference. He compare images pixel by pixel by rotating

current image in outdoor scene (Cozman et al., 2000). This comparing algorithm is inspired from face recognition (Bichsel and Pentland, 1994), (Etemad and Chellappa, 1997). He find minimum distance of two images and the following angle is difference between home and current location (Makadia and Daniilidis, 2006). He compare this result with traditional way, using magnetic sensor. The comparing result show these two methods have similar performance. With this result, he shot its credit.

Meanwhile there is some improvement of computer vision technique. This technique is also highly related to this vision based navigation. Moller R. apply one of the computer vision technique to visual navigation (Möller et al., 2010). The applied technique is warping (Binding and Labrosse, 2006). The warping technique is originally used in computer vision field to align two images with different point of view. This image registration technique is similar to find relationship two image from both location (Giachetti, 2000) (Labrosse, 2007). First of all, he model this warping equation by movement of robot. With taken snapshot images in both location, he find minimum warping line (Liu et al., 2010). By fitting warping equation to given warping curve, he can decide home direction, angle difference in same time. He claim this algorithm have this advantage that we can get home direction and align angle, but this algorithm also have disadvantage its calculation load is heavy.

There is more improvement of computer vision technique. That is the optical flow technique (Barron et al., 1994). This algorithm made of two major components. One is find feature points, and another is find correspondence between images (Harris et al., 2007). With these steps, we can estimate movement of object (Lehrer and Bianco, 2000). This algorithm is similar to visual navigation. Finding feature point is kind of extracting landmark, and find correspondence is finding relationship between two images (Fischler and Bolles, 1981) (Jogan and Leonardis, 1999). With this similarity of these two methods, Moller R. conduct a experiment that of finding home direction using optical flow (Franz and Krapp, 2000). This optical flow is robust to image distortion. He shows that this can find home direction with optical flow.

From the observation of behavior of insect, there is development by contribution of many researcher. They observe navigation ability of insect, define problem of navigation algorithm, and improve this algorithm by combining with other theories from other field. Due to this problem is highly depends on visual information, the computer vision technique is suggested by solution of visual navigation problem.

2.1 Biologically inspired navigation

2.1.1 Insect navigation

Navigation from visual image includes many many approaches. One of the interesting approach is navigate by comparing images. This approach is originated by one observation of Thomas S. Collett. He is interested in movement of honeybees (Collett et al., 2006). He focus on behavior of bee when they are foraging. This intriguing navigation ability of bee has one special feature. Another insect whose navigation ability is proved is desert ant (Andel and Wehner, 2004), (Müller and Wehner, 1988). This ant spread chemical trail to find out its way and navigate. They have certain chemical route to food, and use this path. Even they losing their way, they trying to find out this path to continue foraging (Kohler and Wehner, 2005). There are several possible navigation to use of insect like bee. One is path integration. Insect know its direction and it movement by observing optical flow. By integrating its movement, it knows it location related to their nest. In their retina, they use optical flow from ground they observe, and same thing from surrounding area. Another navigation method in bee is found by Thomas S. Collett (Collett, 2008). In his theory, bee use visual information while they combining this information with from sky, sun-compass. They know their current direction by using visual information. They know how they rotate by observing optical flow. This combining of information allows them to navigate with more accuracy. They communicate by dancing to other companion with these navigation information. They found that the bee and ant not only use their visual information to navigate, but also can learn about landmarks. This concept of landmark is very important to following navigation algorithm (Graham and Collett, 2006). They conduct an important experiment. That is the ant use and learn landmark as an indicator. To prove this, they set an arena. This arena have some special characteristic that is in Figure 2.1. The arena consist of a tube and a cylinder. This tube force ant to follow through this route to find their food. At the end of this tube, there is some food for ant, and this is tagged by 'S' in below figure. At the opposite end of tube, there is a cylinder means only landmark that they can detect. When ant through the tube and return to home, they cannot observe this landmark at all. Oppositely, when they foraging, they can observe and use it as indicator to food.

They can use landmark only when they foraging to food. In this experiment, they use

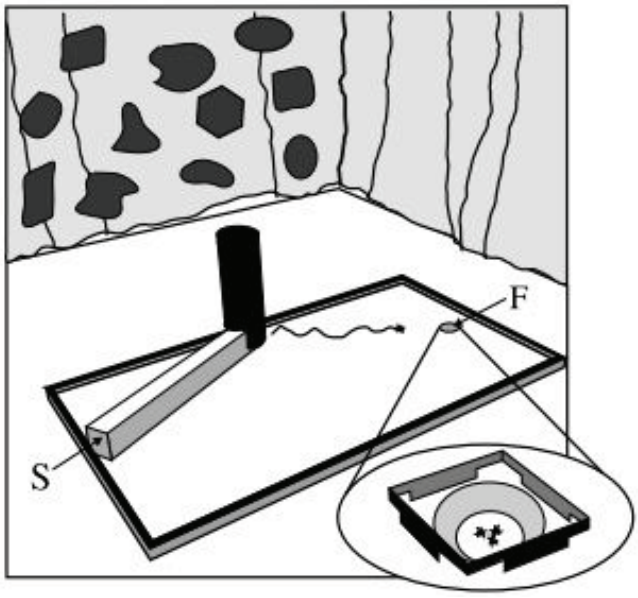


Figure 2.1: Arena to find out effect of landmark existence. Ant travel to source(S) to home(F) or reverse of it. At the source, there is only route to home, and one landmark represented by black cylinder is not observed when they return home. Only they can observe this landmark when they foraging food. Reprinted from (Graham and Collett, 2006).

this different observation. They train the ant. One group is repeating return home. They are kidnapped to food location when they reach their home. By observing path of this trained ant, we can conclude the effect of existence of landmark. Another group is just repeating foraging. When they reach food from home, they are kidnapped to home. The last group is do this both without kidnapping. This different condition of each case can occur difference of path they travel. They observe this path by the training.

This result is interesting. When they foraging and returning, they can learn about environment more faster. This group has straight trajectories with less training. Another group, just foraging group can observe landmark. This group is learn more faster than returning group with landmark existence. Last group, returning group is cannot use major landmark, large black cylinder. Even they cannot use this landmark, they are available visual cue from surrounding arena. With this surrounding cues, they can learn about environment with slow speed. They need more training time to find out straight way to home. This result implies the ant can use landmark existence. This concept of landmark is base of other algorithms. After they proving ant use landmarks

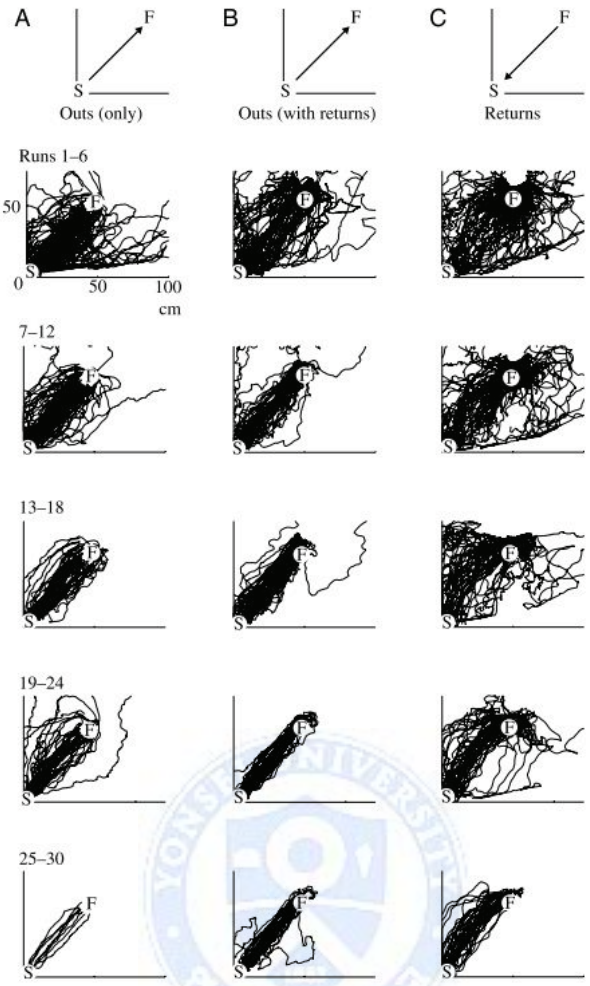


Figure 2.2: The path of ants by the training. Group A allowed foraging, group B is allowed foraging and returning while group C is just allowed returning. Reprinted from (Graham and Collett, 2006).

as indicator when they find food, they explore about this characteristic. To find out the characteristic of landmark, they conduct another experiment (Harris et al., 2007). When they observe the landmark in previous experiment, the black cylinder is seems like just black plane without any pattern on its surface. In this case, the landmark is just black plane. There is no information in the middle of plane without any pattern. The informative area of landmark is the edge of it, so they focus on this effect of edge. Ordinary landmark have two edges in left, and right side. They also consider upper edge and bottom edge by changing height of landmark. To find out the effect of edge, they apply gradient to surface of landmark to one side. This gradient is also not distinguishable. This landmark has only one edge due to its gradient. This gradient difference

of edge will explains about the effect of edge. To remove one edge of landmark, they apply gradient. This gradient can also affect the result. To find out this effect, they do this experiment by changing density decreasing of gradient.

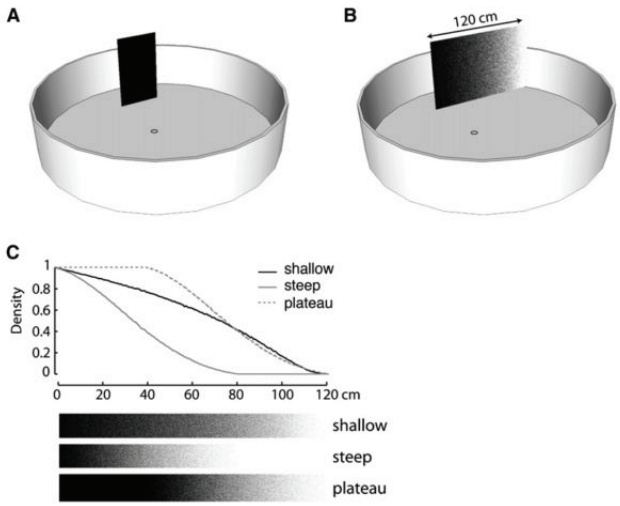


Figure 2.3: A. Arena with two edge landmark. B. Arena with one edge landmark with gradient. C. Types of gradient applied in B. Reprinted from (Harris et al., 2007).

They use two kinds of arenas. One is use two edge landmark, and another is one edge landmark with gradient. By comparing these two types of arena, we can know the effect of edge of landmark. In training phase, they train some specific condition. There are four cases to train. One is using gradient landmark, and place food at edge. Another case is using same landmark and place food at 10 Cm from edge. Rest of cases are using two edged landmark. They record trajectory of ant in training phase. The training result is in Figure 2.4.

In this figure, middle of graph implies that the location of landmark related to ant heading. Last of graph for each case is fixation relative to food. In this training phase, we can conclude that when they foraging, they use edge as landmark, and when the edge is far from food, they have some error to find food. Oppositely, when the food is just at the edge, they travel straight to the food. Their biased fixation relative to food is one of the evidence that they use edge as landmark. There is another factors when we apply the gradient to landmark. One is the gradient starting point. If this starting point is changed, and ant use it as landmark or evidence to food location, ant will go wrong way. With trained ants, they record the trajectories of ant with changed landmark. The result is interesting. Even the landmark is changed, they can find food

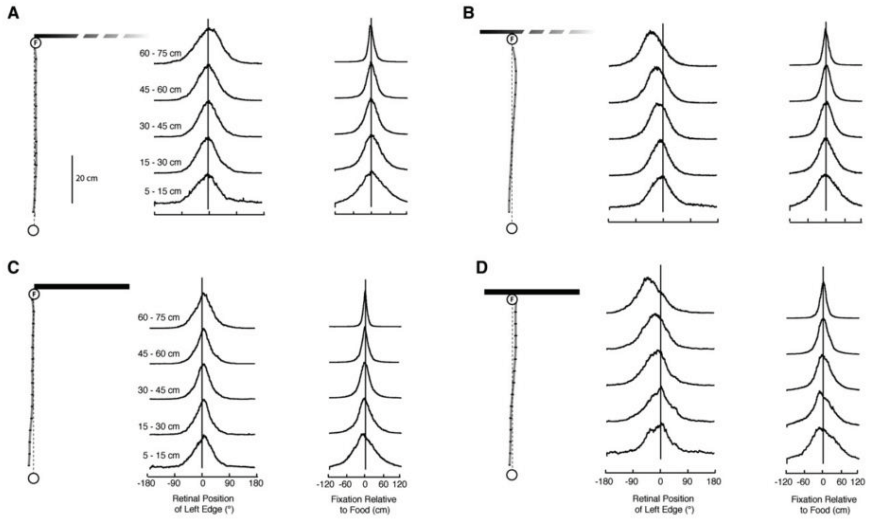


Figure 2.4: Training conditions. A. using gradient landmark, and place food at edge of it. B. using gradient landmark, and place food at 10 Cm from edge. C. using two edged landmark, and place food at edge of it. D. using using two edged landmark, and place food at 10 Cm from edge. Reprinted from (Harris et al., 2007).

location directly. They doesn't affected by starting point of gradient. Another factor, depth of gradient, is also possible factor to confusing ant's navigation algorithm. To compare this, they do same thing by changing depth of gradient. Also, they can find food location directly. With these experiment by changing starting point of gradient and depth of gradient is not affect ant. In another phrase, ant does not consider gradient at all. This means they mainly detect the edge of landmark.

2.1.2 Snapshot hypothesis

Wehner suggest the "snapshot hypothesis" (Wehner and Räber, 1979), (Wehner et al., 1996). This hypothesis explains how to animal can return their home or target location. This hypothesis claims that the animal can store the visual cue of surrounding environment, and they can compare this visual cue to find out direction. From this hypothesis, many research about this hypothesis are happen. One of the interesting work is from franz (Franz et al., 1998). He assume that there just one landmark observed in current and home view, and we know observed angle. With this assumption, we uses ring sensor model described in Figure 2.5. By adding up these associate displacement vectors, we can get the

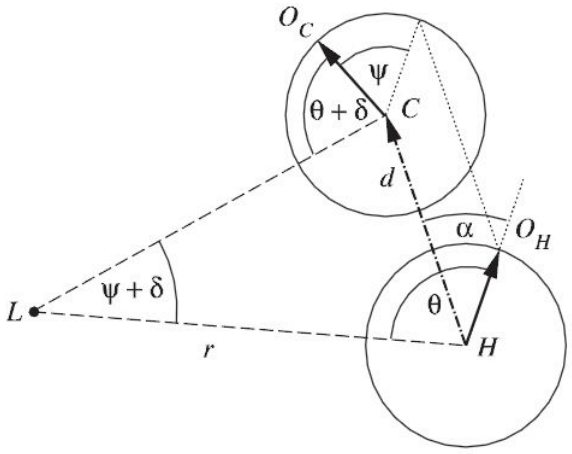


Figure 2.5: Senor difference at home and current position. Reprinted from (Franz et al., 1998).

With this model, he suggest mathematical relationship between these variables. To solve this equation, there are two major condition. One is the correspondence problem. In this model, we assume there is a landmark. In real problem, a landmark observed in home location is not always distinguishable in current view. There is change of point of view to occur disrupt of image. With this problem, we must know this correspondence of landmark in images from both location. Without this assumption, find out home direction is very challenging problem. There is another critical condition to solve this problem. If we don't know the distance from landmark, this information must be compensated by other information or by some assumption. These are major challenging problems of local visual navigation. We will introduce how they overcome this problem later. Some use estimation of distance, and some made tremendous assumption to solve this problem. To overcome this problem, he suggest two assumptions. One is isotropic distance assumption. This assumption is that each landmark is independent, and distance of landmarks are not affected by viewing direction. He use the concept of associated displacement vector (Hong et al., 1992). This displacement vector is defined by $\theta_i + \delta/2 + \pi/2$ for $\delta > 0$, and $\theta_i + \delta/2 - \pi/2$ for $\delta < 0$. In this definition, index i represent landmark index. By normalize and adding up these displacement vectors for every observed landmarks, we can estimate home direction. This scheme have advantage that is this displacement vectors are robust sensor noise. This algorithm is basic form of later work of Rofer (Röfer, 1995), Wittmann (Wittmann, 1995), and Moller et al (Möller et al., 1998). He suggest weighted version of this algorithm. Thus, he calculate its error and analysis about this error. He also suggest its convergence that

this algorithm can direct home direction. There is another way to compensate lack of distance information, the equal distance assumption. This assumption is consider landmarks are far from agent enough to assume the distance between robot and landmark in both locations are equal. This assumption made relationship between home and current location more easier. This assumption is just valid when there is no object in close range. To complete and determine these variables, this needs at least three landmarks. The solution of this relation is in form of field. This field can be described as matched filter. This matched filter is researched by Krapp and Hengstenberg (Krapp et al., 1996), and established theoretically by Nelson and Aloimonos (Nelson and Aloimonos, 1988) and Mallot et al. (Mallot et al., 1996).
This algorithm is based on the assumption that is landmarks are equal distance. This assumption cause some error. Franz analysis this error mathematically (Franz et al., 1998). He also conduct an experiment with real robot. Figure 2.6 describe the arena he execute experiment.

Figure 2.6: Arena of conducted experiment by Franz et al. Reprinted from (Franz et al., 1998).

He set the toy houses. This cue of houses is landmark the robot can observe. He uses modified Khepera robot. He uses conical mirror to get panoramic image (Gaspar et al., 2000) (Goedemé et al., 2005). From image, the intensity of mirror by horizontal position is input of this system. This conical mirror imaging technique is suggested by Chahl and Srinvasan (Chahl and Srinivasan, 1996) and Yahi, Nishizawa, and Yachida (Yagi et al., 1995). The horizontal resolution is 4.6 degree. He applied Wiener lowpass filter (Goldman and Johansson, 1978). To reduce the effect of illumination change, he remove background, and apply histogram equalization to maximize contrast. He draw the vector fields by different home location. The result is on Fig-

ure 2.7.

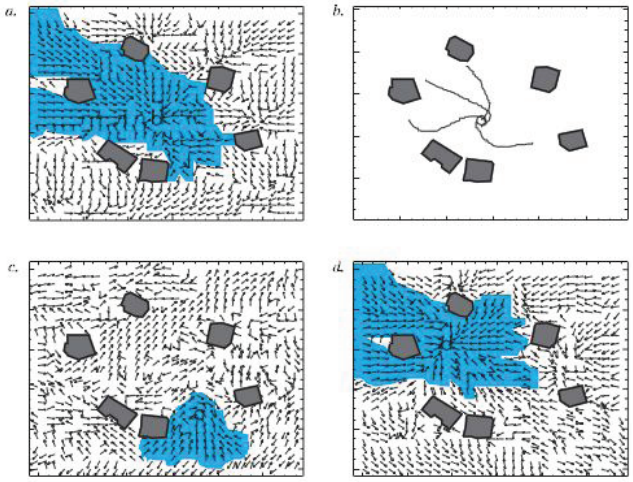


Figure 2.7: Vector fields in arena with different home location. Shaded area means catchment area. Reprinted from (Franz et al., 1998).

In the result figure, shaded area implies the catchment area. Figure 2.7B. is drawing of exact trajectory of robot from other location. With this catchment area, he evaluate

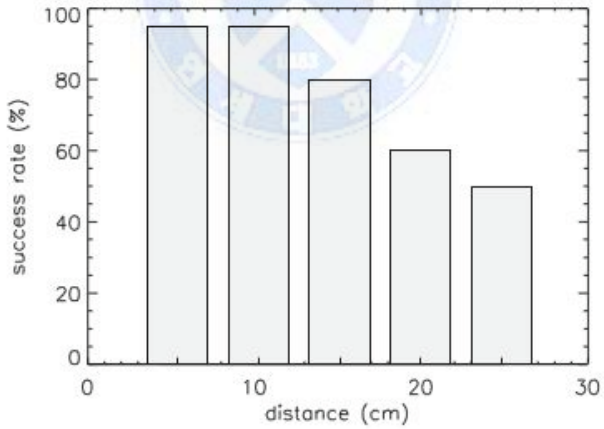


Figure 2.8: Success rate by the distance from home location. Reprinted from (Franz et al., 1998).

the performance of this algorithm by success rate by the distance between home location and current location. The evaluation result is on Figure 2.8. The success rate is decreasing by the distance.

He suggest a hew way to find home direction with snapshot image. This algorithm is based on ”snapshot hypothesis” from insect navigation ability like ant or bee. This

algorithm inspire many researchers. From this idea, there is great one who contribute to this field.

2.1.3 Navigation based on landmarks

As explained above, insect has amazing navigation behavior. Previous work is just observing insect behavior to find out insect's navigation ability and its characteristic. Dimitrios Lambrinos apply this navigation algorithm of insect to real robot (Lambrinos, 1998), (Lambrinos et al., 2000). Another work for navigation of ant using polarization is included his work. He combine this polarization information and visual data to find out home direction. In this paper, we focus on their work of visual information. Many experiments of navigation of bee are conducted in previous work (Cartwright and Collett, 1983). This previous work provide a model by comparing snapshot from home and current view.
He follows snapshot model in Figure 2.9.

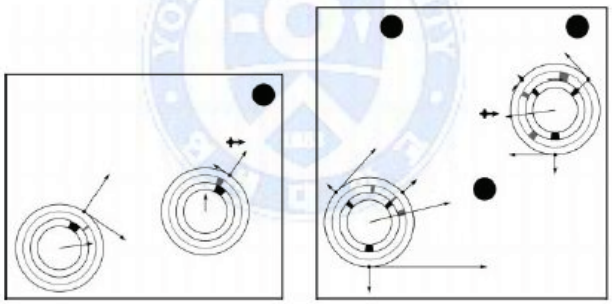


Figure 2.9: Proportional vector model from snapshot model. Reprinted from (Lambrinos et al., 2000).

He improve snapshot model, and suggest one model. This model is called Proportional Vector model(PV model). Original snapshot model has one problem as we provide in previous section. The lack of distance information can occur error to estimate home direction. To overcome this, he consider change of detected width of landmark as distance. It seems quite fair assumption. If the object is near, landmark must be detected wider, while far object is looked narrower. This difference can be a indicator of distance. With this information replacement of parameter, it can be more accuracy. He suggest this model. He conduct an experiment with a robot. The experimental environment is described on Figure 2.10.

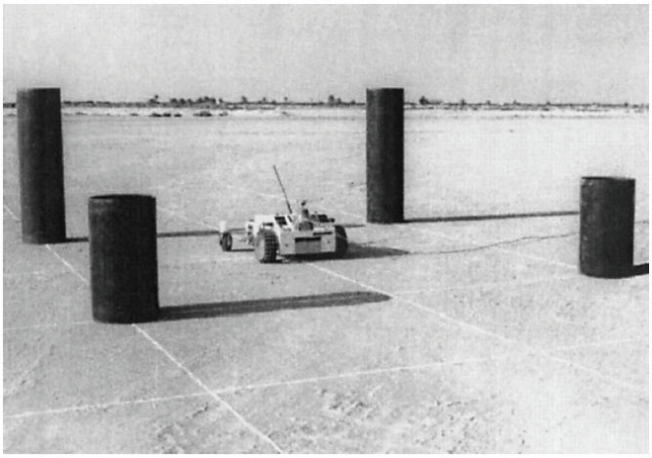


Figure 2.10: Experiment environment. The position of robot is home location. Reprinted from (Lambrinos et al., 2000).

In this environment, he uses four landmarks. He change arrangement of landmarks. The home location is registered position in figure. In this environment, he conduct navigation experiment. To apply the model, he need to extract landmark location or width of landmarks. Following figure explains how he can extract landmark.

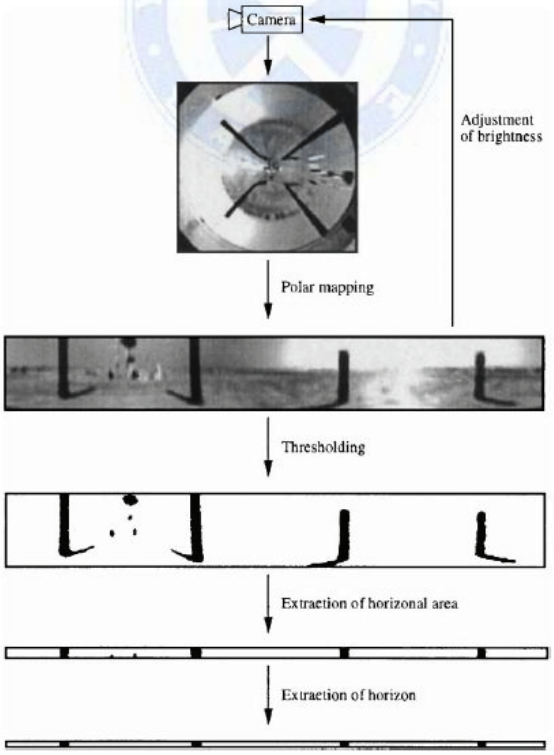


Figure 2.11: Image processing for landmark navigation. Reprinted from (Lambrinos et al., 2000).

He uses omni camera to get the image around agent (Winters and Santos-Victor, 1999). In this arena, or environment, there is artificial landmarks whose color is black. There are four black artificial landmarks. These beams can be extracted by clearly distinguishable contrast relate to environment. He extract these landmarks by thresholding. There is only landmarks and noise factor like black object in far range or shade. He manually select the band that can extract landmarks. This horizontal area is picked and we can extract location of landmark by this information. In this extracted landmark, we also can know its width to apply proportional vector model.

He suggest another algorithm, the average landmark vector algorithm. This algorithm is basically based on work of Franz et al. This algorithm is using average landmark vector. The landmark vector is defined by unit vector from agent to observed landmark. Average of these landmark vector can represent characteristic of each position. This algorithm is explained graphically in Figure 2.12.

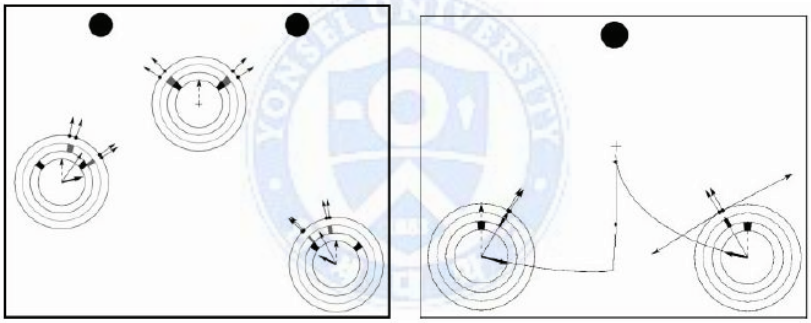


Figure 2.12: Average landmark vector algorithm. Reprinted from (Lambrinos et al., 2000).

The difference of average landmark vector for each position indicate home direction. This algorithm also can provide home direction well. This algorithm is one that just consider only change of bearing. He suggest 2 algorithms, proportional vector algorithm, and average landmark vector algorithm. These algorithms are valid and can indicate home direction well. To prove this stability of algorithm, he compare these two methods and traditional one, snapshot model. He set 27 landmarks. These landmarks are very many to recognize. Each landmark has different size. In this environment, we simulate how the agent will move by comparing two image from home and current location.

In this result, we can confirm the convergence of these three methods. In this simula-

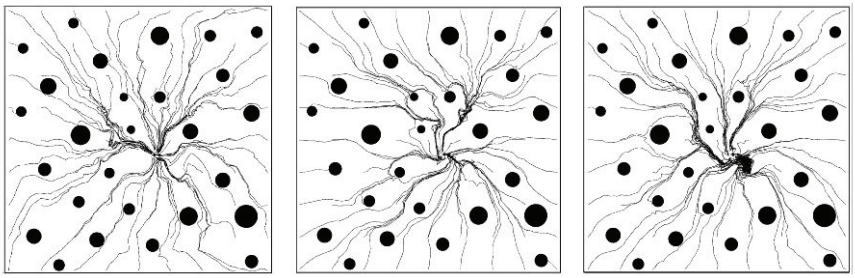


Figure 2.13: Snapshot model(left), PV model(center), and ALV model(right). Reprinted from (Lambrinos et al., 2000).

tion environment, there are many landmarks. The conductor of this experiment expect this result is not good by overcrowd environment. Even this environment, the result is very good, and the show the robot can find home location with these methods.

The coworker of D. lambrinos, R. moller, study further of ALV algorithm (Möller, 2000). In his study, he show the advantage of using average of vectors rather than using sum of vectors. This averaging of vector is robust to occlusion. When a landmark observed in home location is missing in current location, averaging can guarantee that direct home location with smaller error while homing direction of summing method is diverge.
He also conduct a experiment with simple robot. He set the arena with some landmarks on the wall. He uses photo diode to sense landmarks. This robot have 32 photo diodes to sense landmarks around robot. With this detected landmarks, he apply average landmark vector algorithm to find out home location. He vary number of landmark. He uses just one landmark to find out home direction. In previous section, Franz (Franz et al., 1998) already show that the solution is needed at least three landmarks. This one landmark condition violate this condition. Due to this lack of landmarks, the home direction is not converge. This home direction is just direct landmark position. He also conducted with two landmarks. This case also didn't show convergence of homing vectors. To compare this condition, he conduct three landmarks. This result show homing vector direct home location. He evaluate his experiment with how the robot close to home location. He also show the convergence of algorithm by mathematical proof.

Optical flow is also kind of image processing to detect movement of object in two scene (Fennema and Thompson, 1979), (Horn and Schunck, 1981), (Nagel, 1982). With some constraint like there is less change of brightness or a gradient assump-

tion (Brox et al., 2004), (Papenberg et al., 2006), this algorithm provide velocity of object well (Hatzitheodorou et al., 2000).

This algorithm is consist of major two parts (Shi and Tomasi, 1994). One is tracking features from image (Harris and Stephens, 1988), (Vardy and Oppacher, 2004), (Vardy and Oppacher, 2003), and another is finding correspondence. To finding correspondence of two images, some researcher try to extract distinctive features (Se et al., 2002), (Lowe, 2004). Some researcher try to find this correspondence just using brightness of block (Weber et al., 1999), or color regions (Gourichon et al., 2003), (Goedemé et al., 2004), (Gourichon et al., 2002). This optical flow and visual homing methods with snapshot hypothesis are very similar problem (Gluckman and Nayar, 1998). Traditional optical flow is block matching (Jain and Jain, 1981). This algorithm is compare distance of block from two images in certain radius. With this block matching method is applied local navigation (Vardy and Moller, 2005).

In his experiment, he assume that orientation is not changed in current location. His test image set is already aligned, and he assume these images are aligned. He also provide that using visual compass, magnetic sensor, or polarization is one way to find its aligned angle. In his previous suggested algorithm, the average landmark vector provide homing direction with its approximated distance. Magnitude of difference of average landmark vector is also indicate distance from home location. In this optical flow technique, this homing vector is normalized and magnitude of it means nothing. He uses image database of the robotics laboratory of the computer engineering group of bielefeld university. He apply this image database two homing algorithms, one is warping, and another is block matching algorithm he suggested. The result of this simulation is on Figure 2.14

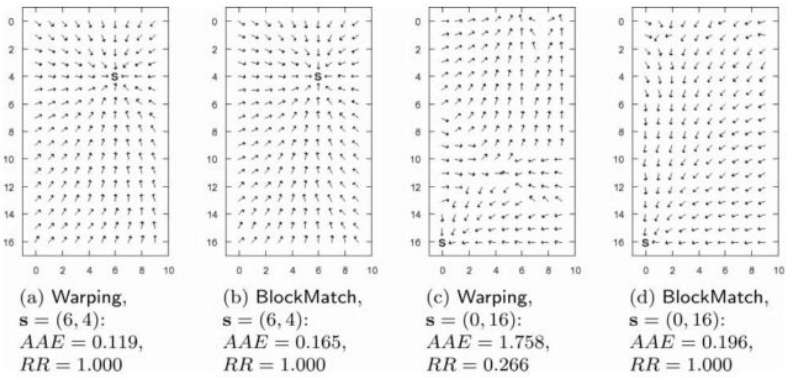


Figure 2.14: Result of warping method and block matching method with different home location. Reprinted from (Vardy and Moller, 2005).

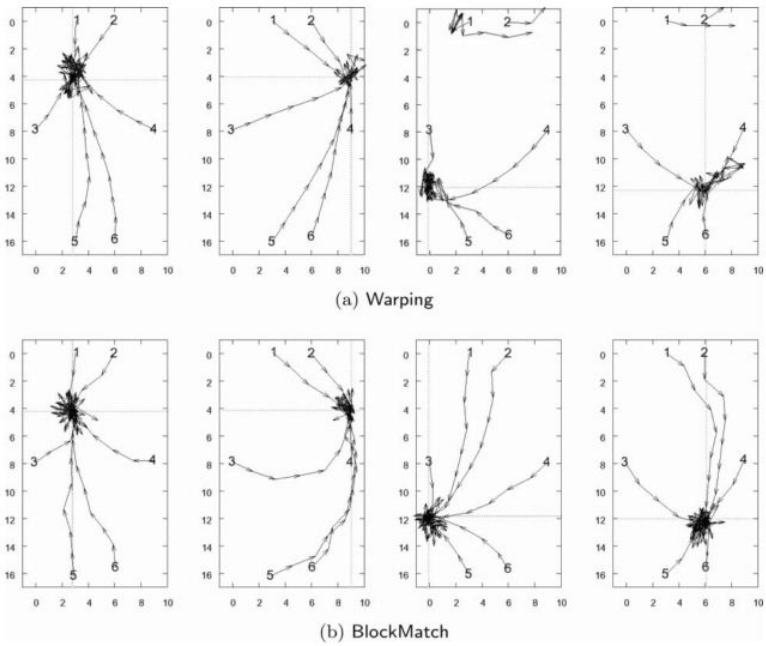


Figure 2.15: Robot trajectory of two methods. Reprinted from (Vardy and Moller, 2005).

In this result, both methods are converge with snapshot position at (6, 4), but at (0, 16), the performance of warping method is decreased. This snapshot point have close large object that cause break of equal distance assumption. This lack of assumption distort the performance of warping method. Meanwhile, block matching method have perfect return ratio that means its performance is best. This simulation is off-line test. To prove its validity in on-line situation, he apply this methods to real robot. He draw its trajectory in Figure 2.15.

There is another contribution of Cha. Y (Cha and Kim, 2012) using optical flow technique. In his work, he use another kind of optical flow technique from Lucas and Kanade (Lucas et al., 1981). In his work, he combine the average landmark vector algorithm from D. Lambrinos (Lambrinos, 1998), and pyramid LK algorithm. He extract rotational component of optical flow, and find out aligned angle with intensity of generated optical flow. Another experiment with real robot is conducted. With optical flow, robot estimate its movement and integrate all movement it made. With this movement, it return home location. The result is on Figure 2.16.

He uses three kinds of movement, and confirm this agent can return to home well. He evaluate this homing performance with angle error and distance error. He also show its robustness in low resolution image case. This optical flow algorithm have many common problems and characteristic in visual homing navigation. This can be another

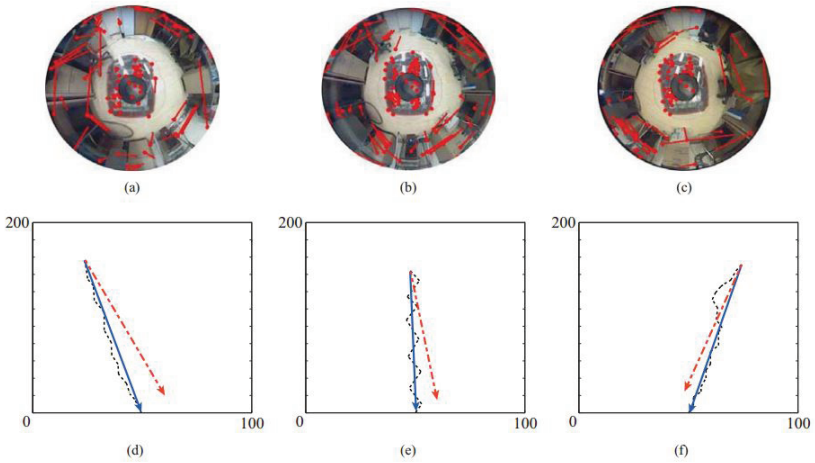


Figure 2.16: Example of robot trajectory using optical flow algorithm in different kinds of movement. Reprinted from (Vardy and Moller, 2005).

solution to local visual navigation problem.

2.1.4 Place cells

There is another different approach to understand a navigation ability of animals. Especially, the rodents have better performance to find their home after foraging. There are many experiments on these rodents to find out their characteristic, and change when they find their home or their current location. In previous approach, try to find their navigation ability by changing environment or condition while this approach observes their activity of brain directly. This approach analyzes brains of these rodents, and find out specific part of brain involves a navigation task which is called the Hippocampus. These cells are located in CA3 and CA1 pyramidal cells. Figure 2.17 (a) shows anatomical structure of the Hippocampus and its connections.

The Hippocampus is located in the medial temporal lobes, and it is connected to the entorhinal, parahippocampal, and perirhinal cortices (Andersen et al., 2006). This Hippocampus involves long term memory, and there are many explanation for this part of a brain. This organ involves not only long term memory, but also a spatial domain which concern problems like forgetting place where they been. There are many reports on damage of the Hippocampus in humans cause spatial memory problems. This reports implies that this area of a brain involves spatial tasks like navigation problems. In human, this Hippocampal area helps to navigate, and there is a discovery of 'place cell'

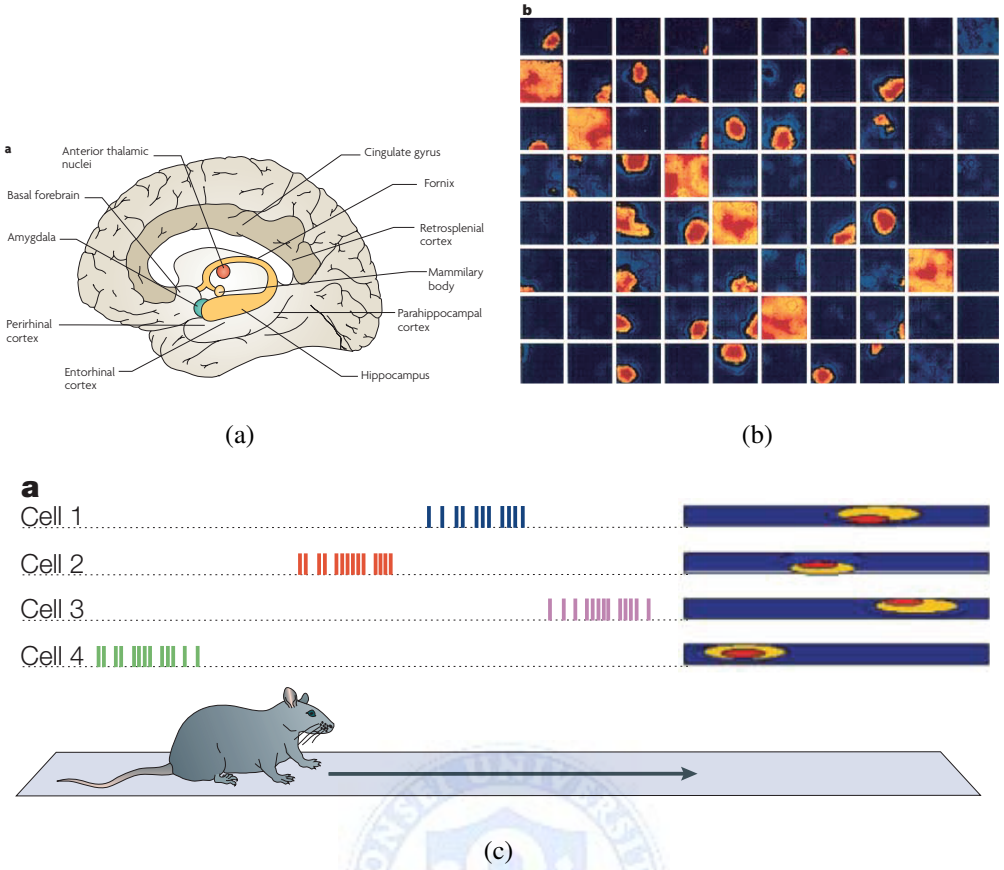


Figure 2.17: (a) Anatomical model of the hippocampus, (b) Place cell activity by its location, (c) Place cell of rodents (Reprinted from (Nakazawa et al., 2004), (Bird and Burgess, 2008))

has same role in rats (O'Keefe and Dostrovsky, 1971). The place cells fire by location of an animal in environment. As they explore an environment, these place cells are trained, and it provide reasonable spatial representation of itself. These place cells are not direct response of their sensory input. If it was direct response of sensory input of their visual information, it is changed by its direction which decide overall scene of own (Cressant et al., 1997). In experiments on these rats, it is not changed even they had different scene, but it has fixed location. This experiments implies that these place cells fires by its location it is not dependent on its orientation. Rather than it depend on a visual information directly, it depend on its own movements, and this process is known as the 'path integration' (Hafting et al., 2005). There is another cells involves the navigation tasks which is called the head direction cells. These cells are activated by its orientation, and this activity can provide the clue of orientation problem. With these cells, there is a model to explain how it works together.

The Byrne, Becker, and Burgess (BBB) model is a plausible theory to explain mechanism of place cells (Addis et al., 2007). This model involves a anatomical structure of the Hippocampus, and its connection. When we remember something, we can remember details of environment. Boundaries of environment, distance, geometric characteristic can be these details. These details can be classified into two different classes. One is environment information which represent geometrical information, and overall scene around it. This information is processed by the parahippocampal cortex (Epstein and Kanwisher, 1998). Another is feature information which is distinctive characteristics of scene including edge, distinctive colored, and uniquely shaped object. This information is processed in the perihinal cortex (Buckley and Gaffan, 1997). The BBB model combines this perceptual information from these parts of brain. This model remember features from environment, and it also remember detected direction using the head direction cells. This geometrical information limit the environment, and direct relative location of these detected features. In this environment, there are many features in their sight. This distinctive features can provide clues to find their location. The perihinal cortex remember these features, and its detected angle. It remember many features, and it is activated by current visual information. If it observe features in their memory, the perihinal cortex remind its detected angle by location. By comparing these detected angles of features from memory and visual information from current location, it decide a current location relative to given environment.

To support this model, there are many experiments on rodents. Figure 2.17 (c) shows that recording of extracellular action potentials of the hippocampal cells by moving of rodents. These place cells are composed of several independent cells which are activated by its location. One of researches to find out mechanism of the brain is observing this activity directly using probe. In this experiment, a probe is located in a brain of a rodent, and let him move freely. During exploring, and moving, activity of the hippocampus is recorded. From this recording, there is activity depending on its location in figure 2.17(b).

This activity of place cell is simple model to solve a navigation problem, and finding its own location. We take this model which is activated by its location, and finding its current location. With this model of brain, we suggest a solution of localization problem.

2.2 Feature matching algorithm

2.2.1 The scale invariant feature transform

By moving camera, we detect an object with different scale. An closer object is seems bigger, while far object seems smaller. This scale problem is one of the important factor of feature matching problem. There is a feature matching algorithm robust to this scale problem, the scale invariant feature transform (Lowe, 1999). This algorithm uses multiple scale of image. Different octaves of image can have resolution issue. To overcome this issue, this algorithm also apply the Gaussian mask to cut off image frequency. With this different size of image, we find local maximum or minimum points to obtain scale invariant points. With these points, we calculate SIFT keys, and these are on Figure 2.18. This SIFT keys can describe features of image. They compare

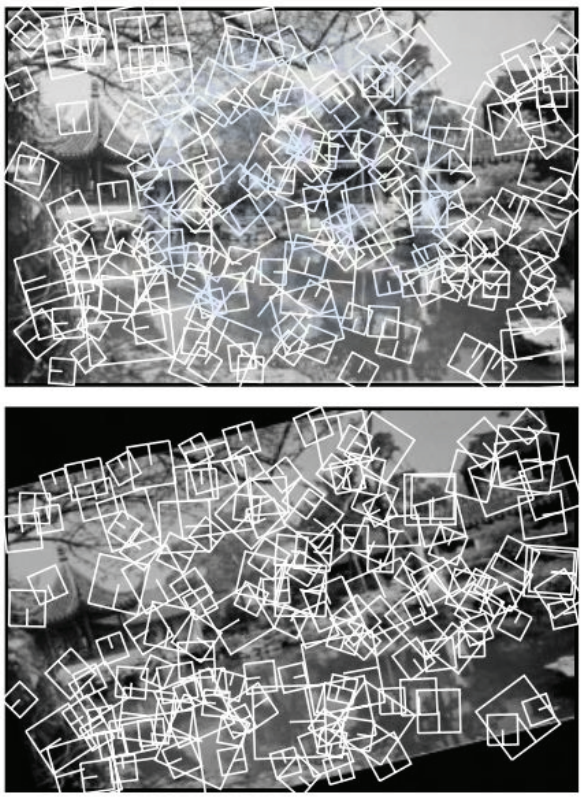


Figure 2.18: The SIFT keys for original image(upper) and transformed image(lower). (reprinted from (Lowe, 1999))

these keys from original image and transformed image. They generate transformed image by rotating, stretching, rescaling, and changing brightness. 78% of keys are

matching even they transform image. They show that keys are robust to transform of image. With these keys, we can detect object with robustness to image transform. In this paper, we do not need to recognize an object. We can use these keys as landmarks. Each keys have its own characteristic, and these are also considered to landmarks.

2.2.2 The speed up robust feature

The SIFT algorithm is robust and have good performance, but there is a shortage. That is this SIFT algorithm have heavy computational load. This shortage lead to its running speed. To overcome this shortage, there are many alternative algorithms such as PCA-SIFT. This algorithm reduce the dimension of SIFT keys to reduce computational load. There is another feature matching algorithm with advantage of running speed, the speed up robust feature algorithm. This algorithm uses the fast Hessian matrix to find out interest points. The Hassian matrix includes second derivatives of Gaussian. To reduce this calculation, they approximate this operation by box filter on Figure 2.19.

Figure 2.19: Approximation for Gaussian second order partial derivatives. (reprinted from (Bay et al., 2008))

Above derivatives of Gaussian is approximation along y direction and xy direction. By applying this matrix to integral image, we can obtain interesting points.
After we find interesting points, we need to calculate descriptor for each points. To achieve this, we apply the Haar wavelet filter. This process is on Figure 2.20.

2.3 The SLAM problem

The simultaneous localization and mapping (SLAM) is a problem of concurrently the structure of the surrounding environment with perceived sensor data, while simultane-

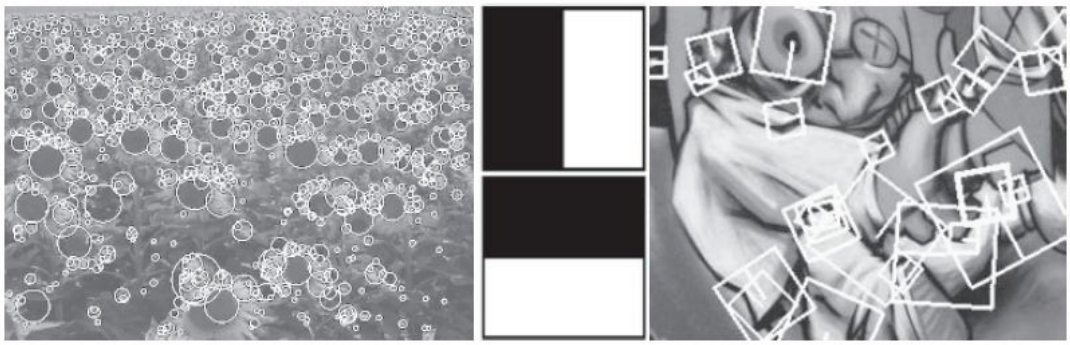


Figure 2.20: Interesting points(Left). Haar wavelet filter(Middle). SIFT descriptor(Right). (reprinted from (Bay et al., 2008))

ously getting localized in it. This concept was originally developed by Hugh Durrant-Whyte **?**, and Cheeseman **?**. This SLAM problem concerns building a map from unknown surrounding environment by mobile robot, and localization itself at the same time. This algorithm have many parts to solve this problem. First of important step of the SLAM algorithm is a state estimation. This part estimate its current position information from state updating. For instance, a mobile robot moves with given velocity for a second. In this case, we can estimate its position after moving by solving the kinematics of this robot. This position information have same meaning of the state in the SLAM algorithm. This estimation of position have same meaning with the prediction in the Kalman filter. Another important part of the SLAM algorithm is the estimation. This estimation part is find its position using the map, and sensor data. This estimation process also can be divide into some specific techniques by the way to solving the problems. For instance, we consider that this SLAM algorithm uses landmark feature to find its current location. For this case, extraction of these landmark feature can be a first step to find its current location. This extraction can be any sensing mechanism. There are most commonly used sensors which are the visual camera, and laser sensor. Visual camera provides the visual information, and it does not provides landmark feature itself. It requires extra feature extraction algorithm from visual information. Visual template, vertical line, horizontal line, and corner based feature are most popular feature extraction from visual information. When this algorithm is depending on laser sensor, it use distance data itself to construct the map. With this information, it associate current sensor data, and previously built map. With this association, it can find its current location from sensor data. We have two estimation from state estimation, and sensor data. There is final step of the SLAM algorithm which is combine

these two estimation. This step is also called state update its state, and a map including landmarks. For each step, there are many ways to solve, and this solution is highly related to applying environment.

In the SLAM algorithm, it represent all parameters with probabilistic expression (**?**). It does not decide its current position, and heading direction in deterministic express. To express possibility of position, it assume that this distribution is the Gauissian. This distribution is represented by its mean, and covariance. With these mean, and covariance information of current position, we can find its possible location. This assumption is one of important assumption to apply the Kalman filter. If actual system does not satisfy this assumption, it cause error to estimate its current location.

There is another important assumption that to find its current location which is the 1st order Markov chains. It is memoryless conditions. This assumption that current probability of current state is only related to previous state. This assumption is expressed by following equation.

$$Pr(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, ..., X_1 = x_1) = Pr(X_n = x_n | X_{n-1} = x_{n-1})$$

This assumption implies that the current position is only affected by proximately previous step. This assumption allow us that we don't need to consider older previous steps to estimate current position. This assumption cause drastic calculation reduction. Most of SLAM algorithms are based on this assumption.

With the 1st order Markov chains, it apply the Bayes filter to estimate current location. The Bayes filter is following equation.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

This form is for descrete event, and this expression is for general probability. We have to apply control factors, and previous state to estimate current state. In above expression, an event A is that probability distribution of possible location when a mobile robot was at x_{t-1}, and there is known conditional probability to have new state from control at time t, $p(x_t|u_t, x_{t-1})$.

$$\overline{bel(x_t)} = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

This equation express of P(A) in the Beyes filter equation. This term also called the ''belief" in SLAM algorithm. This equation implies that possiblity of state after control action. This equation does not consider the observation of sensor data. This equation consider all possible state which is expressed in above equation as integration

range. For each possible state at time t-1, it calculate all possible state at time t after control action. This control action is in $p(x_t|u_t, x_{t-1})$. This term implies that relationship between current current state, previous state, and control action. By integrating all possible estimated current state from previous state, it can find estimation current state by using previous state, and control action. In general mobile robot case, state implies that its current poses which include coordinate, and heading direction. A conditional probability term, $p(x_t|u_t, x_{t-1})$, implies that a kinematics of the mobile robot. If this robot depends on two wheel, this control factor can be velocity of these two wheels. With this kinematics, we can estimate its possible location from previous state. With this estimation, we can find its current state just using its control action, and previous state. To compensate this estimation, it also use observation. This observation equation is in following equation.

$$bel(x_t) = \eta p(z_t|x_t)\, \overline{bel(x_t)}$$

This equation implies that we already observe surrounding environment with sensors. With this sensor data, and map information provide abundant information to estimate its current position. In this equation, it also considers all possible state which is expressed in $\overline{bel(x_t)}$. For each possible state, it calculate conditional possibility to obtain observed data with given sensor which is expressed in $p(z_t|x_t)$. In above equation, a term η is for normalization, and this one is in following equation.

$$\eta = \int p(z_t|x_t)\, \overline{bel(x_t)}\ \ d\overline{bel(x_t)}$$

With these calculations, we can estimate its current state by compensating estimated state, or the belief, with observed data from sensor. This approach have one critical shortage which is this filter have too much calculation load to apply in real problem.

2.3.1 The Kalman filter

In the Bayes filter, it consider all possible previous state, and its conditional probability. This considering of state increases computational load too much. To overcome this shortage, the Gaussian assumption is applied in this algorithm. This applying of the Gaussian assumption is the Kalman filter. To apply this assumption, there are many calculation procedures, but these are not considered in this paper. With this applying of assumption, we can have few simple equations which is the Kalman filter in following

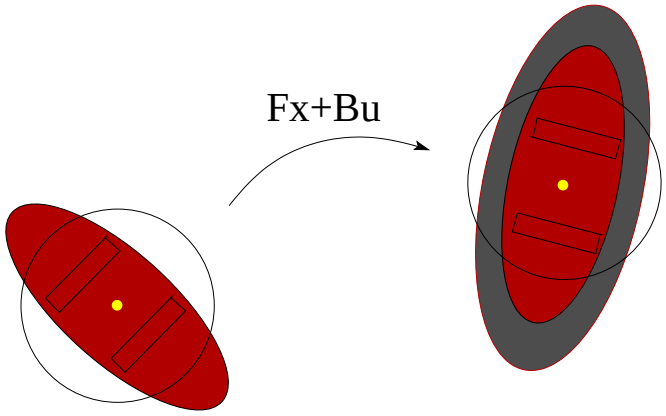


Figure 2.21: Relationship between the kinematics, and state transition of the Kalman filter.

equations.

$$\widehat{x}_{k|k-1} = F_k \widehat{x}_{k-1|k-1} + B_k u_k$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

These equations are first two equations of the Kalman filter which is called the prediction. This equation is basically same procedure of calculating the belief in the Bayes filter. In the Bayes filter, it considers all possible state. The Kalman filter solve this problem with applying the Gaussan assumption. It consider only mean of all possibility, and covariance of it. In other word, the Bayes filter express the belief with probability distribution function while the Kalman filter express that with its mean, and covariance. In above equations, parameter k implies that index of steps. k-1 implies a previous step, and k implies a current step. A matrix F is state transition matrix which implies transition of state by step. In some case, the state can includes velocity information of robot. In this case, position is always changed without any control action by step. This matrix consider this change for some case. u_k implies that control action. In two wheel driven mobile robot case, this matrix is velocity of two wheels. A matrix B implies change of state from control action. This matrix is a kinematics matrix which implies change of position, and heading direction by velocity of two wheels. If there are transition matrix between two probabilistic parameter, there is one important relationship between its covariances with an following equation.

$$Y = FX, \quad Cov(Y) = F * Cov(X) * F^T$$

$$Y = X + C, \quad Cov(Y) = Cov(X), where, C\, is\, a\, constant$$

A second equation of the Kalman filter implies this property. A control action is added to state transition, but this adding is not change its covariance because this state transition can be considered as constant for that step. From these property, we can find estimating covariance with a second equation of the Kalman filter. In this equation, there is a matrix Q to add movement noise. Figure 2.21 shows a relationship between the kinematics and state transition of the Kalman filter. In previous state at time k-1, its distribution is expressed by its means and covariance. A yellow dot of left robot is mean of previous state, and red ellipse implies its covariance. From this distribution of previous state, and control action, we can estimate its possibility distribution of the current state. In this case with a mobile robot with two wheels, a kinematics is solution of finding state transition, and control matrix. This ellipse is not changed when we just apply the kinematics. A red region of right robot is estimated covariance of current state. This current state does not consider control error. In above equation, a term 'Q' is considering this control error. By adding this term into covariance, a size of ellipse is increased, and it means its distribution is enlarged by control error. This consideration of control error allow compensate this error. If this error is not enlarged, or considered, control error is remaining by steps. If it happens, control error is accumulated as it happened with the "dead reckoning". A magnitude of control error term affect overall performance. If it is too small, this system has accumulated error. Otherwise, when it is too small, this system depends on sensing data. It also can cause estimating error by a type of sensor. With this prediction, we can estimate its current state with control action.

There are other equations of the Kalman filter in following equations.

$$\widetilde{y}_k = z_k - H_k \widehat{x}_{k|k-1}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

These above equations are the innovation of the Kalman filter. These equations implies estimation of sensor data. A term z is observed sensor data at time k, and a term H implies that sensor data that would be detected on a state from prediction process. Difference between observed data in real, and one from prediction can be a measurement residual. A term S implies covariance of the measurement residual. This covariance of the measurement residual. The observed sensor, a term z, is constant at time k. It does not change at that time, and this property cause this covariance of measurement residual is only depending on estimated state from prediction, and observation matrix H. For this reason, covariance of the measure residual is similar form of covariance of

current state from prediction. A term R is a factor to compensate observation error. If observation is always perfect, this term is not needed. In real case, there is some limitation of sensor, and this cause degrade of observation of data. For instance, a laser sensor can detect distance from obstacles, but there are severe error when these obstacles have transparent. This cause severe degrade of observation, and it also cause severe degrade of estimation of current state. From this measurement, the Kalman filter calculate the Kalman gain as following equation.

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

This Kalman gain implies that inverse of observation matrix, and uncertainty ratio of prediction, and innovation. This matrix convert a sensor data into state by calculating inverse of observation matrix term. This term also considers covariance of current state from prediction, and innovation. With the Kalman gain, there are update equation to make final estimation of state by combining these two information from control, and observation in following equations.

$$\widehat{x}_{k|k} = \widehat{x}_{k|k-1} + K_k \widetilde{y}_k$$

$$P_{k|k} = (I - K_k Hk) P_{k|k-1}$$

Final estimation of state is linear combination of estimated state from prediction, and innovation. This ratio of coefficient is from the Kalman gain which consider its covariance. This Kalman filter also update covariance of state with these two types of information. These equations are from mathematical analysis of the Bayesian filter with the Gaussian assumption. With these consideration, most possible state can be calculated by above equations. This update process is in figure 2.22. In above figure, there is estimated state from prediction of the Kalman filter in left of the figure. This prediction also has its mean and covariance to express its distribution. On the other hand, there is another estimation from the observation, or measurement. There are observation from sensor date on right side of the figure. It detects surrounding environment, and finding its possible current state. This is also expressed with its mean and covariance of estimated state. In update step, it combine these two Gaussian distribution of estimation of state to find optimal solution. There is an optimal mean, and covariance with highest probability, and use this point as final decision of current state. The Kalman filter combines two different estimations to find optimal current position. This concept is very quite fit to solve the SLAM problem for some reasons. There are

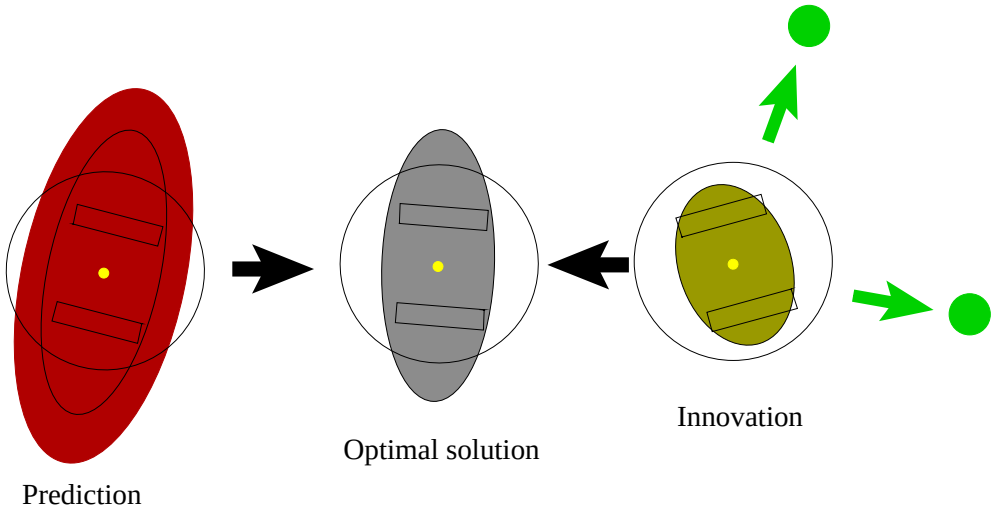


Figure 2.22: Innovation, and update of the Kalman filter.

typical measurement to find its current state, or position which are accumulating control action, and measuring surrounding environment. These two estimation has its own characteristic. Accumulating control action, or the ”dead reckoning”, has accumulated noise. When there is any noise, and control error, this error propagate to next step of estimation. This error is increased by step, and overall performance of estimation of state is also decreased. This characteristic guarantee performance in close range, but it is not work with wide range. On the other hand, sensor measurement also have one shortage depending on type of the sensor. In general case, this sensor data is not sufficient to determine its current state uniquely. For example, a case that environment has symmetric structure causes a vague decision to find its current position. When there are many similar structures in environment, find its current location with only using the sensor data is not possible. These characteristics of two different measure of position can help each other to estimate its current state. This concept is a basic of the Kalman filter in the SLAM problem.

There is another important step to implement the Kalman filter into the SLAM problem which is decide the state. In previous equations, there is a state variable ’x’. We have to decide what is this variable, and what it includes. For simple mobile robot, coordination of the robot, and heading direction can be the state variable. If there are more complex case, more variables can be included in this state variable. In general case, it assume that this robot is moving on plain environment. It has x, and y coordinate to express its current location. With this coordinates, a heading direction is also can be an important factor to decide its position, and it is also can be included to the state variable. This case have another problem to implement the Kalman filter

directly into the SLAM algorithm which is absence of the map. When we previously have entire map, this additional expanding of state variable is not needed, but if it is not, there is one serious problem. This problem is it cannot decide its current state with measured sensor data. In this case which does not have the map, a mobile robot build the map with their sensor. They have to travel surrounding area, and observe the environment. These observation is also not always perfect, and these information of the map have to be in its memory. To solve this problem, it build its own map in the state variable. In this state variable, it includes position of observed landmarks, or features. By setting this state variable, it update its own position while it also update location of detected landmarks. This updating process cause improvement of overall performance of observation. This technique is commonly used in many different types of the SLAM algorithm based on features in environment. This technique have another problem, which is it increase size of state variable, or dimension of state variable. A size of this state matrix is highly related to number of landmarks in environment. It means that when we target large environment, this size is too large. In calculation of the Kalman filter, there is inverse operation of matrix which size is same to the state variable. This inverse matrix cause inaccuracy to estimating current state. There are many different technique to solve this problem by managing considering landmarks in given environment. This management is add newly observed landmarks, and remove some landmarks which was detected far before. This technique have to be attached to the SLAM algorithm to implement in real problem.

This using of Kalman filter have another problem to implement in real robot. To solve this SLAM problem with the Kalman filter, we assume that distribution of state is the Gaussian. On the other hand, there is state transition matrix, and observation matrix to find next step, and observation. This matrix cause a problem to apply directly which is that these matrix operation is not linear. When we apply non-linear operation into the Gaussian distribution, its result is not the Gaussian. Every time we apply this non-linear operation, it cause estimation error by violating the Gaussian assumption. To overcome this problem, there is two commonly used algorithms. One is the extended Kalman filter which approximate this non-linear transformation by the tailor series. This non-linear transform can be approximated by the tailor series. This extended kalman filter depends on the Jacobian matrix, and use this matrix as transform matrix, and observation matrix. This technique is most commonly used Kalman filter for many applications. There is another type of kalman filter to overcome this non-linear problem which is the unscented Kalman filter. This algorithm suggest another method

to express covariance. We can transform a point with non-linear transformation easily, and we can find a mean of transformed points by transforming a mean point. Estimating proper covariance matrix after transformation is challenging problem. This algorithm suggest the Sigma vector concept to simplify this problem. Its covariance matrix is square matrix, but it can be expressed by these Sigma vectors. These Sigma vectors are vector points from mean of data distribution. A direction of each Sigma vector is same to each Eigen vector of covariance matrix. A length of each Sigma vector is related to its corresponding Eigen value. This algorithm transform these Sigma vectors, and it decide proper covariance with these transformed Sigma vectors.

2.4 Summary of Chapter 2

In this chapter, we introduce different background ideas. Our basic motive is based on behaviour of animals, and there are many analysis of them. From this behaviour, there is a commonly accepted hypothesis which is the 'snapshot hypothesis'. It implies that they rely on their visual information, and they have to find direction to target location by comparing two snapshot images. From this hypothesis, there are many derived algorithms to explain, and simulate animal navigation.

We also introduce another approach which is based on their anatomical characteristics. The place cells are most important results to explain this characteristic.

Chapter 3

Local navigation using HOG descriptors

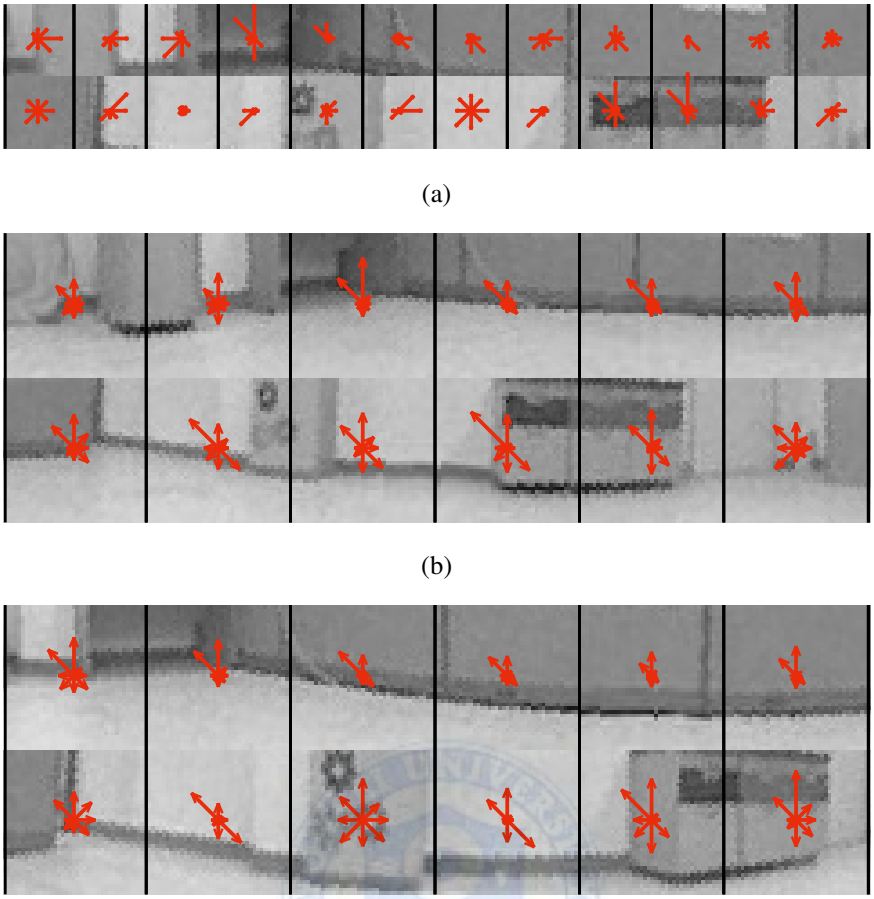
Many algorithms are derived from the navigation ability of animals using visual information. Animals compare images between their home and the current location and estimate the direction to home relative to their current location. This comparison requires proper landmarks to extract valuable information from the visual information. Many methods are used to extract landmarks; we use HOG descriptors as landmarks in an image. In this section, we present details on the extraction of the HOG feature; and use specific bins as landmarks. This section is prepared for submission as a scientific paper (Baek and Kim, 2015b).

3.1 Navigation using HOG descriptors

3.1.1 Section based on HOG descriptors

We select a narrow band of an image that contains meaningful landmark information 3.1. This region includes most objects in the environment, and is not drastically affected by image warping. We ignore the rest of the image to estimate homing direction. Images of both the current, and home locations are compared to determine thehoming direction. The band image is then divided into many subsections (as shown in figure 3.1).

A centered gradient filter is used to calculate the oriented gradient vectors for each



(a)

(b)

(c)

Figure 3.1: The histogram of oriented gradient vectors of each section of band image. (a) with small cell size(20 by 20), (b) with large cell size(40 by 40), at home location, (c) at current location.

pixel of an image. Each cell contains HOG vectors. In figure 3.1, the number of bins is 8, and we vary it to compare the performance. This HOG descriptor for each cell is a fundamental element of our algorithm; we depend on the information obtained from the descriptors. The cell size has a considerable effect on the overall performance because it decides the amount of information in an image. When the cell size is very large, details in an image merge and mix into one histogram thereby degrading the overall performance. On the other hand, when the cell size is very small, details in an image do not sufficiently express the characteristics of cells. Thus, we have discussed an optimal cell size that delivers the best performance when locating the homing direction presented in this section.

These images shown in figure 3.1 include many objects in the environment Each object

has a unique gradient pattern; this pattern is similarly observed in images from different locations. This consistency of the HOG descriptor provides meaningful information for use in navigational tasks.

3.1.2 Aligned angle of panoramic image

To realize the accurate homing direction, we need to determine the required orientation of the mobile robot, or camera used and ascertain the angle difference between the camera's direction at home; and current locations. Without determining this angle difference, bearing information cannot be extracted from the observed landmarks. To solve this problem, we use the idea of a visual compass and apply the HOG descriptors to compare. HOG descriptors are obtained from panoramic images, and are compared with those in an image taken at a different orientation at the same location shown in figure 3.2. We then determine the distance between the descriptors at far locations and those at the center location and the Euclidean distance between the corresponding descriptors. Thereafter, we rotate one of the images, and calculate the distance by the rotation angle; this distance is at a minimum when these images are aligned. With this image property, we can determine the angle difference, or orientation difference between the images. We can then align these panoramic images, and deteremin the aligned bearing information from the observed landmarks.

3.1.3 Homing direction using HOG descriptors

HOG descriptors are obtained for each section of a panoramic image. With these descriptors, we can determine the appropriate homing direction by comparing two images (one from the home location and another from the current location). We summarize our algorithm using equations as follows :

$$
\begin{aligned}
HOG\ descriptors : & \overrightarrow{H_{11}}\ \overrightarrow{H_{12}}\ \overrightarrow{H_{13}} & \cdots & \quad \overrightarrow{H_{1P}} \\
& \overrightarrow{H_{21}}\ \overrightarrow{H_{22}}\ \overrightarrow{H_{23}} & \cdots & \quad \overrightarrow{H_{2P}} \\
& \vdots & & \\
& \overrightarrow{H_{Q1}}\ \overrightarrow{H_{Q2}}\ \overrightarrow{H_{Q3}} & \cdots & \quad \overrightarrow{H_{QP}}
\end{aligned}
$$

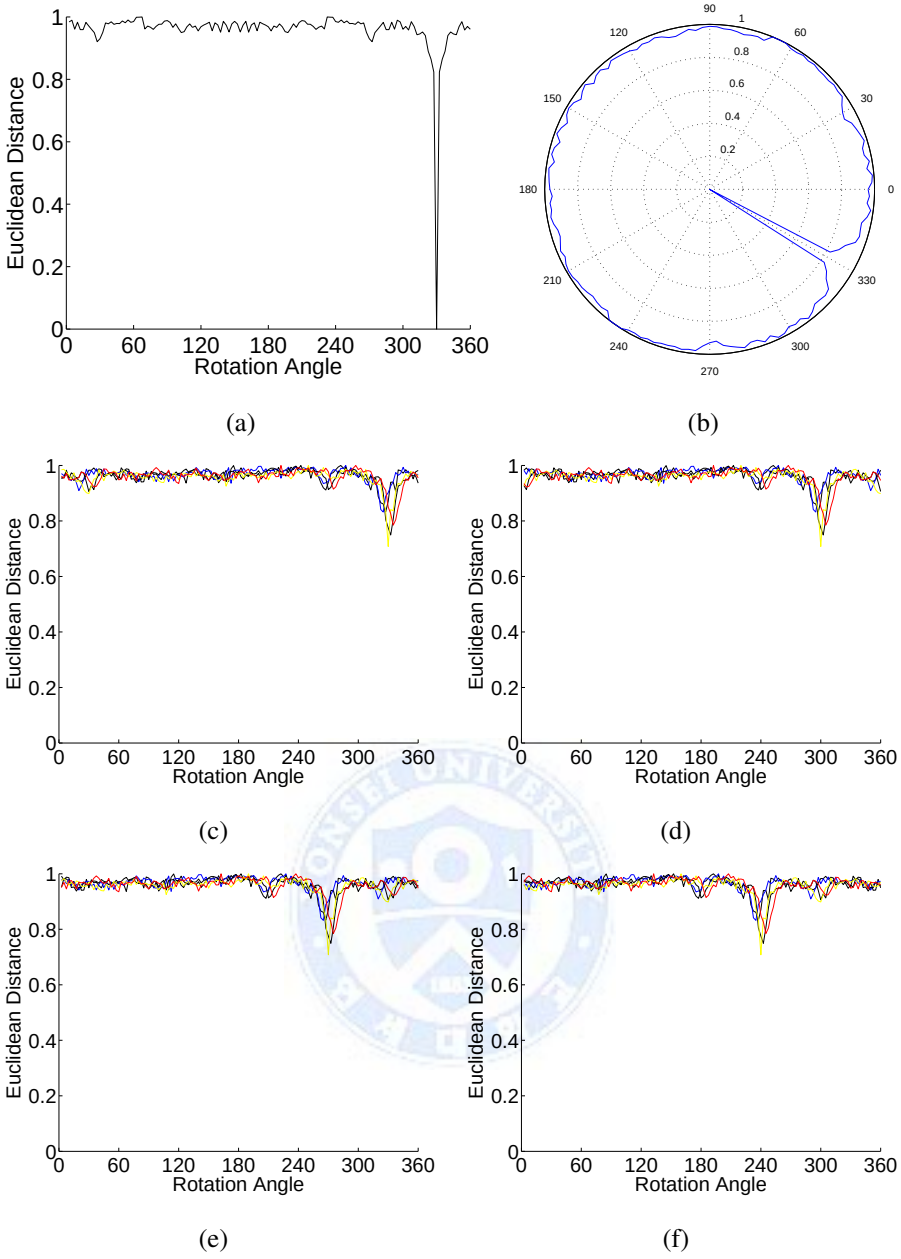


Figure 3.2: (a) The distance between an image from center and its shifted image by calculating the Euclidean distance of HOG descriptors by rotation angle. (b) The result in polar form. Distance of the HOG descriptor between center and far locations by rotation angle. (c), (d), (e), and (f) represent images obtained at different angles, i.e., 30°, 60°, 90°, and 120° respectively.

where P is the number of sections in the panoramic images, and Q is the number of bins for the HOG descriptors. Equations used to decide the average landmark vectors

for each bin at the home location are as follows :

$$\overrightarrow{ALV}_{Home1} = \sum_{p=1}^{P} \{|H_{1p}| cos(\frac{2\pi}{P}p),\ |H_{1p}| sin(\frac{2\pi}{P}p)\}$$
$$\overrightarrow{ALV}_{Home2} = \sum_{p=1}^{P} \{|H_{2p}| cos(\frac{2\pi}{P}p),\ |H_{2p}| sin(\frac{2\pi}{P}p)\}$$
$$\vdots$$
$$\overrightarrow{ALV}_{HomeQ} = \sum_{p=1}^{P} \{|H_{Qp}| cos(\frac{2\pi}{P}p),\ |H_{Qp}| sin(\frac{2\pi}{P}p)\}$$

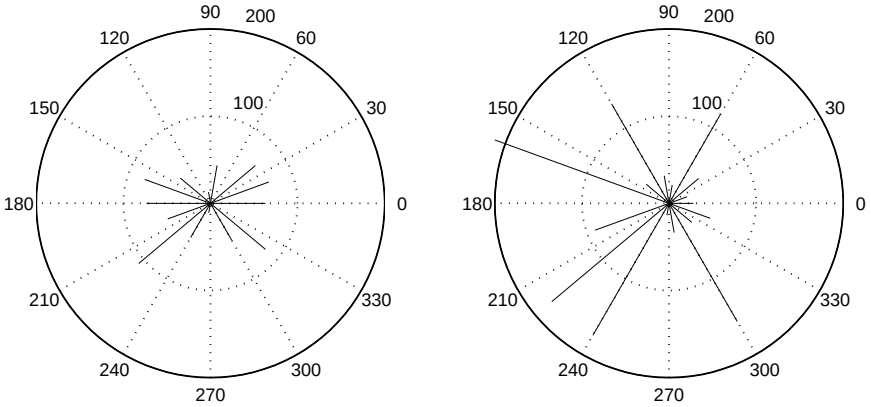
The average landmark vectors from each bin are merged as follows :

$$\overrightarrow{ALV}_{Home} = \sum_{q=1}^{Q} [\sum_{p=1}^{P} \{|H_{qp}| cos(\frac{2\pi}{P}p),\ |H_{qp}| sin(\frac{2\pi}{P}p)\}]$$

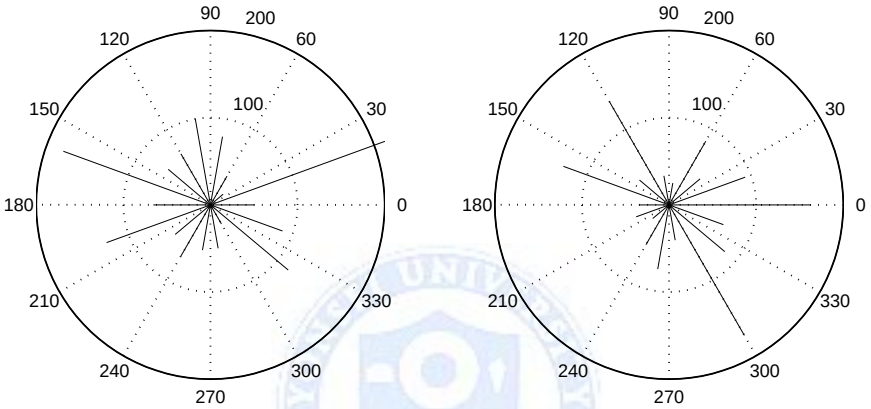
Using this equation, we then obtain the average landmark vector of an image from the home location. We repeat this for an image from the current location. With the average landmark vectors from both locations, it is possible to determine the homing direction using the equations as follows :

$$\overrightarrow{Homing direction} = \overrightarrow{ALV}_{Home} - \overrightarrow{ALV}_{Current}$$

Most well described features are detected at the edge of objects. Using these features, it is possible to recognize the correspondence of vectors, and with the correspondence, we can estimate the homing direction. First, in two images, we assume that the same object occurs at a similar oriented gradient vector. Although the oriented gradient vector is not exactly the same, the angle and magnitude will be the same. We then quantize these vectors in 36 levels (figure 3.3 shows an example of quantized vectors). We gather vectors in a range of 40° - 50° and plot them at points where they were detected. The left side of figure 3.3 represents an object at 130° from an agent, and the object generated vector in a range of 40° - 50°. The right side of the figure 3.3 represents the same object detected at 120° from an agent. This level shows the landmark location and its change clearly, although most levels cannot represent objects or landmarks.



(a)



(b)

Figure 3.3: Examples of quantized vectors of home (left) and current (right) locations. The angles in the figure imply the positions of the images. These vectors have angles in the range of 40° - 50°. This level represents the landmark locations and the difference between the two images.

Figure 3.3(below) shows an example where no distinctive landmark exists within the range; such a case is thus ignored. This quantization enables the correspondence between a landmark in two images to be recognized. Each distinctive vector represent the landmark location of average landmark vector method. It is possible to determine the homing location by calculating the difference between the average vector of the landmark location at the home location and that at the current location (figure 3.4). We thus calculate the average for each level at the home and current locations, and then we subtract them. This subtracted vector will be the direction of the home location. The result is shown in Figure 3.5 (a). The result is obtained from the current location at left-top location and the home location at center of the environment. The optimal angle

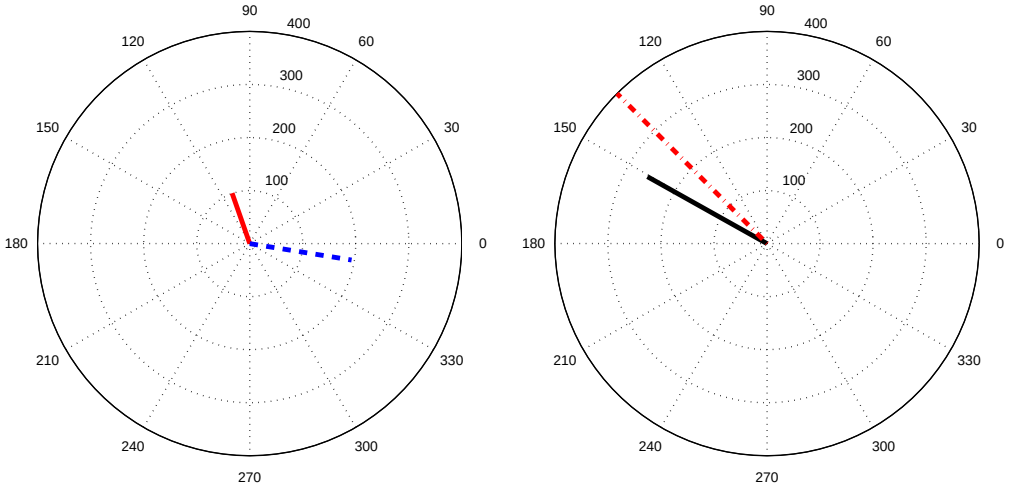


Figure 3.4: Averaging vector at home(left) view and current(right) view(above). The homing direction of this range. This vector can present the homing direction.

of the homing direction is -45°. In the figure, we plot the calculated homing direction using the difference between the average landmark vectors of the home and current locations for each bin. The result shows the dominant vectors representing the homing direction with a small error. It is necessary to navigate several dominant vectors that have large magnitudes; however, many irrelevant vectors also exist. This is caused by undetected ranges in the quantization levels, where there are meaningless vectors with uncertain locations. To reduce the effect of meaningless vectors, we thresholding these vector with the threshold value decided by maximum vector magnitude. We remove the meaningless vectors cannot reach its magnitude to the half of maximum vector. The filtered vectors are shown in Figure 3.5(b). By summing all these dominant vectors, we can estimate the homing direction.

For using the average landmark vector method, minimum three landmarks are required. The features obtained from the HOG algorithm indicate landmark information. We have shown that the arrangement of an average landmark vector array can be used as a distinguishable feature when acquiring landmark location information from two images and have previously suggested that the quantization method can be used to indicate the home location. In the previous section, we observed that many meaningless vectors exist in the quantized average vectors and undetected levels cause such irrelevant data. However, we cannot be certain that the meaningless vector can cancel each other or that the performance will be maintained even if meaningless vectors are re-

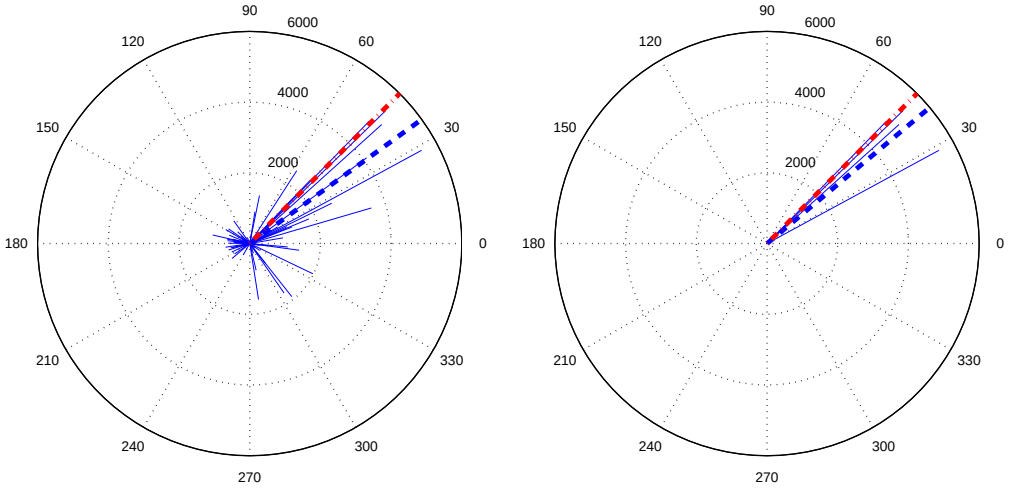


Figure 3.5: Average landmark vectors. (a) Difference among average landmark vectors of all bins; red line indicates the homing direction obtained by adding all average landmark vectors. (b) Threshold difference among average landmark vectors of dominant bins; red line indicates the homing direction obtained by adding the dominant average landmark vectors.

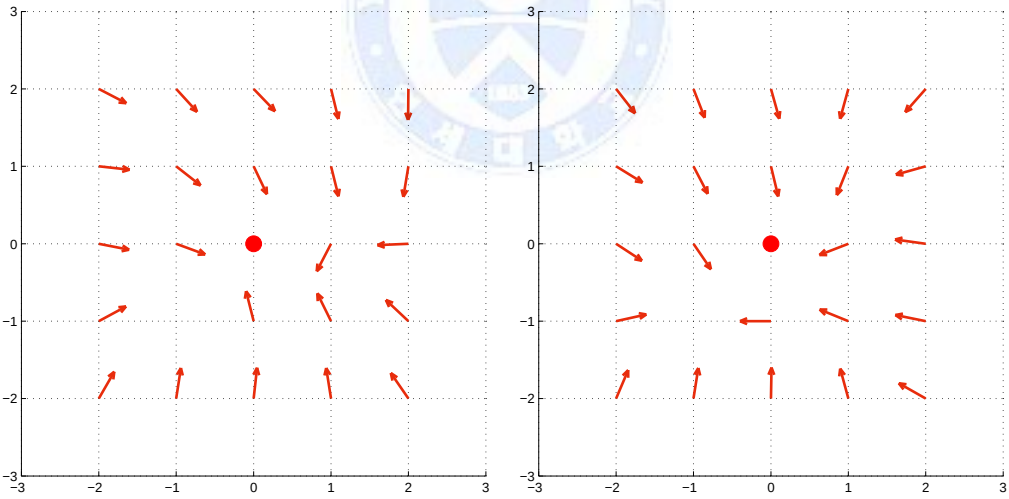


Figure 3.6: Homing direction obtained using given algorithm in a 5 $\times$ 5 environment: (a) from the algorithm and (b) with 0.4 threshold.

tained. To understand the potential effect, we sum all observed data to check whether the meaningless vectors subdue the performance level. The comparison data used are thresholded, and we check whether the thresholding can remove the meaningless data

and improve the accuracy. The hvectors, i.e., the difference between the home and current locations, can be distinguished at the home location. We simulate the hvectors at different locations and figure it out there is quite relationship to homing direction.

3.1.4 Local navigation performance

In figure 3.6, the simulation result at different position around the home located at center. Most vectors point to the homing direction, but there are many meaningless vectors that need to be ignored. Therefore, we first include all meaningless vectors to compare their performance by comparing the angular error of the difference between the observed angle and optimal homing angle. This acts as a concrete parameter to evaluate the method's performance. The average angular error is 18.11°. We also simulate this estimation at different indoor environments, covering a wide range (as shown in figure 3.7). We vary the size of each cell and determine that there is an optimal size for which the appropriate direction is found. When the 20×20 cell size is used, it is not possible to describe the characteristics in each cell because of the small cell size, and thus, it is considered to be very small to find the correct direction. The 80 80 size delivers a poor performance than the other cell sizes because much meaningless information from the surrounding environment is included. In addition, the area and floor and ceiling of the environment in this image contain no specific pattern or characteristic, which causes huge errors in the gradient vectors and degrades the overall performance. Figure 3.7(d) shows the average angular error by case.

Many distinctive features can be observed on the vertical edge of the images. From this observation, we select oriented gradient vectors with a vertical direction. We then divide the range of HOG descriptors by their angles, shown in figure 3.8(c). We divide these descriptors into two ranges to extract the vertical and horizontal descriptors. We then split the orientation into 8 sections and consider that there are vertical components in sections 2, 3, 6, and 7 and horizontal components in the other sections. To determine the optimal cell size and type of descriptors, as shown in figure 3.9, we compare the average angular error by case and by different measurements. Figure 3.9(a) shows the average angular error by case from the overall environment, and figure 3.9(b) shows the average angular error by case from locations near the home location. Figures 3.9(c) and (d) show the result when a threshold is applied. We remove the HOG bins that have a magnitude below 30 % compared with the most dominant one. We determine

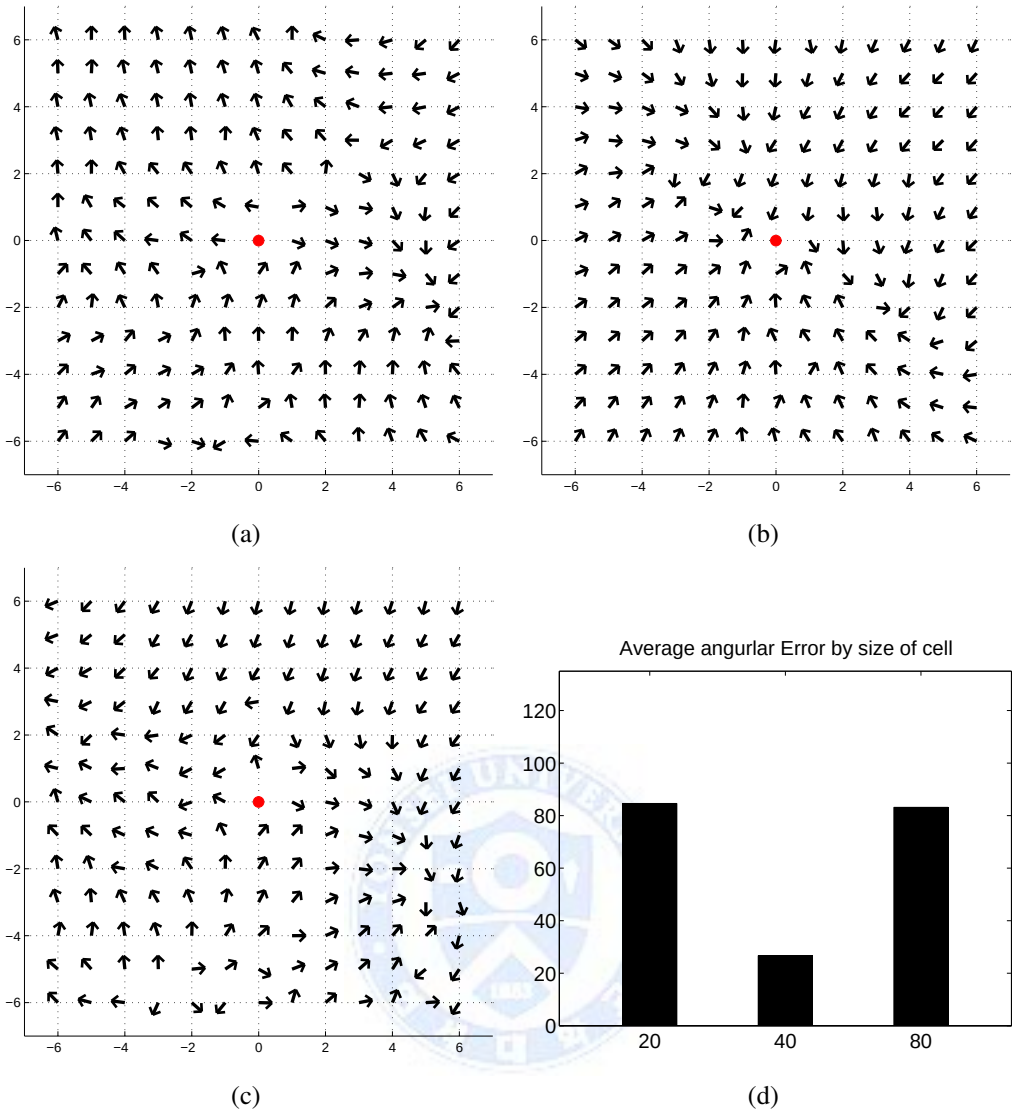


Figure 3.7: Homing direction obtained by the given algorithm in a 13 $\times$ 13 environment using different cell sizes: (a) 20 $\times$ 20; (b) 40 $\times$ 40; (c) 80 $\times$ 80. (d) Average angular error by case.

that there is an optimal cell size and that the type of descriptor does not decide the overall performance. We divide the types of descriptors with associated angles, as shown in figure 3.8(c), and set a specific angle range to decide vertical and horizontal descriptors.

We can find there are optimal size of cell, but types of descriptors does not decide overall performance. We divide type of descriptors with its angle as in figure 3.8(c). We set a specific range of angle to decide vertical and horizontal descriptors. This range can be too wide to express proper vertical descriptors. About 45 degree can be

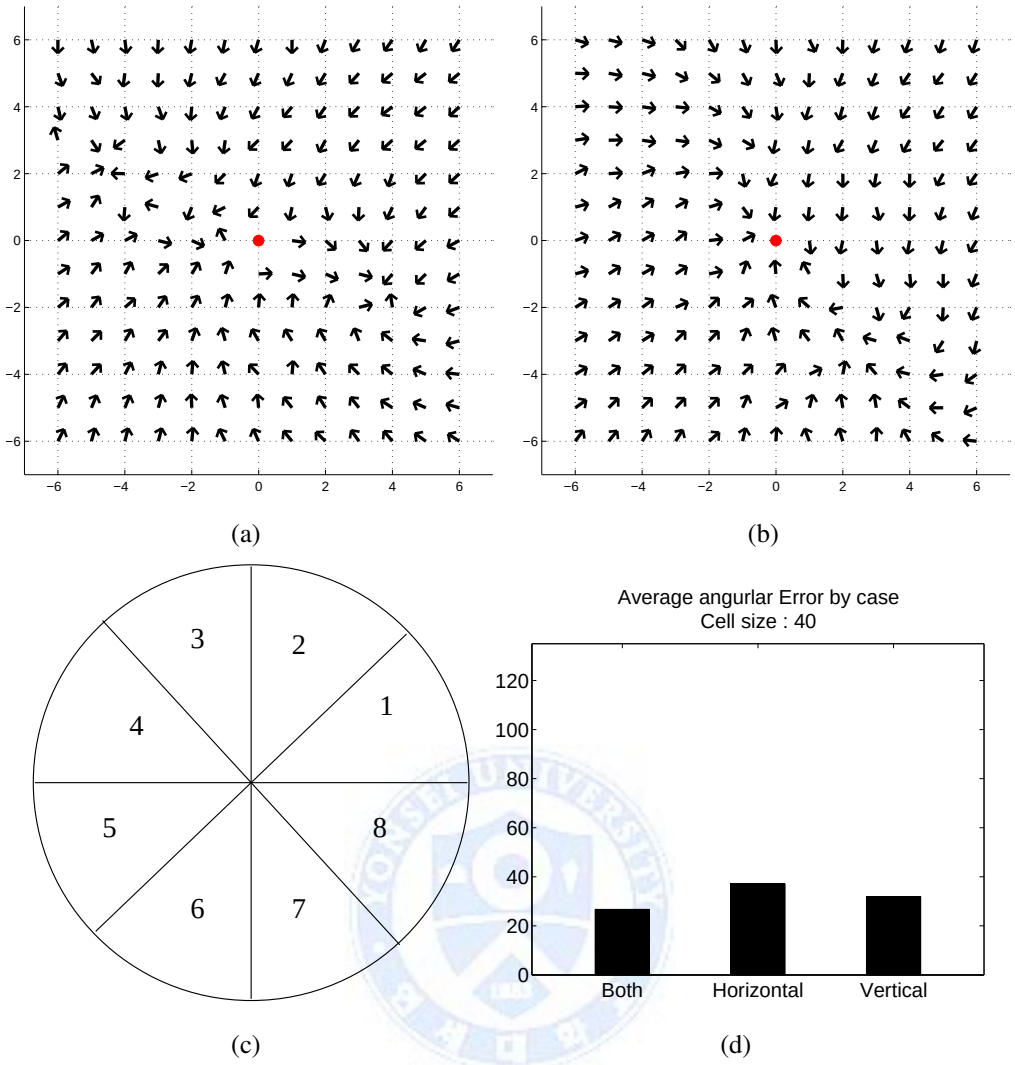


Figure 3.8: Homing direction obtained by the given algorithm in a 13 $\times$ 13 environment with different ranges of HOG vectors: (a) horizontal range and (b) vertical range. (c) Vertical range: 1, 8, 4, and 5 and horizontal range: 2, 3, 6, and 7. (d) Comparison of average angular error by case.

considered to vertical descriptors. To avoid this, we specify the range. We use narrow range of angle to extract vertical and horizontal descriptors in figure 3.10.

However, this range can be very wide to express proper vertical descriptors, and therefore, the range should be around 45°. To avoid this problem, we specify the range and use a narrow angle range to extract the vertical and horizontal descriptors, as shown in figure 3.10. As shown in figure 3.10(a), vertical descriptors are used to determine the homing direction.

When finding vertical descriptors with a wider range, a small area exists wherein it

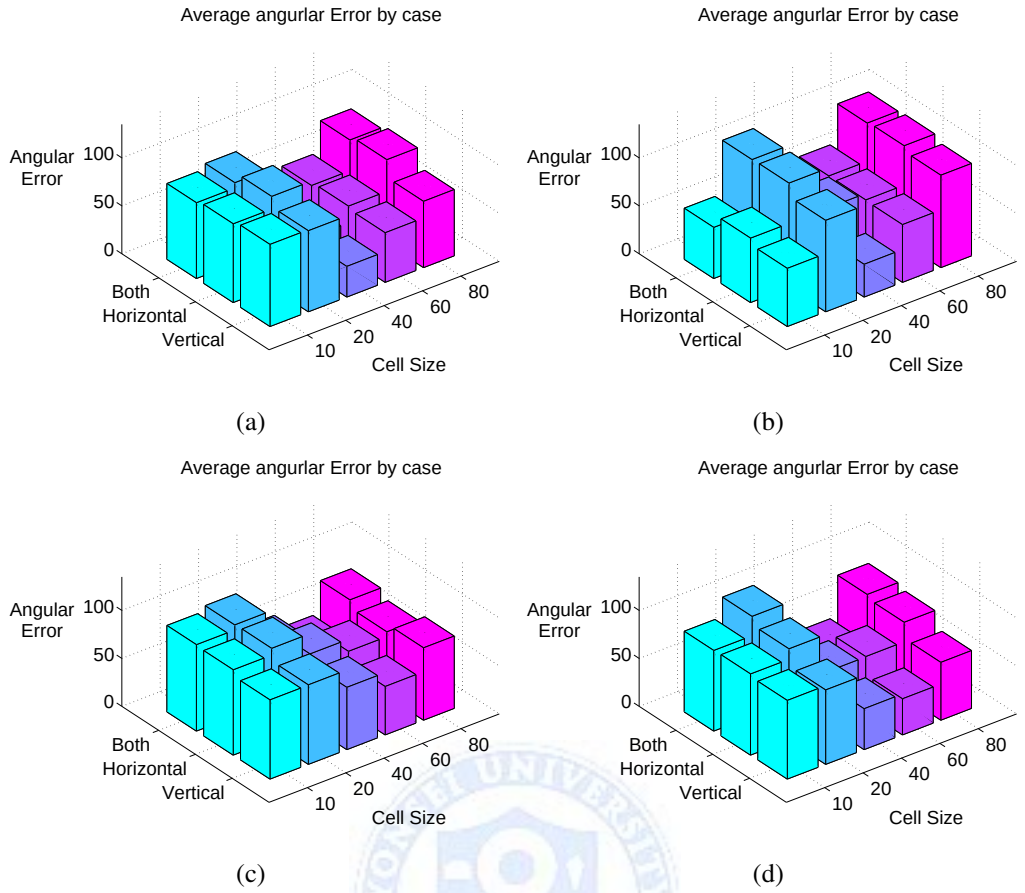
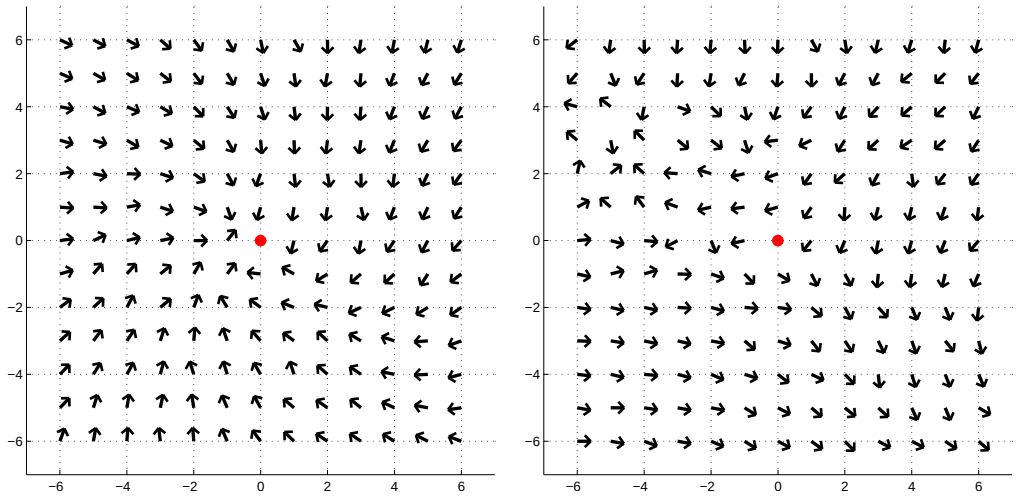


Figure 3.9: Homing direction obtained by the given algorithm in a 13 $\times$ 13 environment by descriptor type and cell size. (a) Angular error by descriptor type and cell size. (b) Average angular error in a close range within the environment. (c) With 0.3 threshold value. (d) With 0.5 threshold value.

is not possible to accurately determine the homing direction, but this does not occur in this case. The homing direction in most of the areas can be determined, and it is merged to home. The overall performance using horizontal descriptors is not sufficient for determining the homing direction. We have an omnidirectional image with warping along the y axis. A horizontal straight line actually appeared as a curve in the panoramic image. Thus, the horizontal descriptors cannot work accurately, and it is necessary to rely on the vertical descriptors to determine the homing direction. In Appendix A, we test this local navigation task with a 13 $\times$ 13 environment and simulate a 11 $\times$ 11 environment that includes many objects in the target area. Using this environment, we simulate the local navigation task shown in figures 3.10(c) and (d). The result shows that the algorithm cannot be used to accurately determine the homing direction when there are many objects in the field. There is a small area which is opposite side



(a) (b)

Figure 3.10: Homing direction obtained by the given algorithm in a 13 $\times$ 13 environment using (a) narrow vertical descriptors and (b) horizontal descriptors.

of object relative to home location. In this area, it is possible to accurately determine the homing direction, which shows that it is valid to determine the homing direction in this environment. We conduct experiments using different types of environments (Appendix A). Figure 3.11 shows the results on a vector map, where the overall performance is observed to depend on the characteristics of an environment. The indoor environment gives a better performance than others, and this is related to the clear and concrete features on an image. Otherwise, outdoor environment have pattern in far distance. The observed bearing angle of these patterns does not change by location, but their HOG descriptors vary between locations because of the noise in the camera. Such a difference causes a bias in the homing direction. In this environment, the top side is open but the bottom side is closed, and such asymmetry causes the biasing problem. Other environments lack clear features. Most images have pattern-less walls, which causes the algorithm to perform poorly. Furthermore, images in this environment have many errors or noise on the images, which causes a degradation of the overall performance. As there is noise on the images, we conduct experiments with noise. We add some noise to the images from the indoor environment and attempt to determine the homing direction. The average angular error by noise level and type is shown in figure 3.12. First, a blur effect is added to the images because this effect occurs when the focus of the camera is not appropriately set. We thus modify the snapshot images using a blurring filter and compare the effect by covariance of the filter. This blurring filter

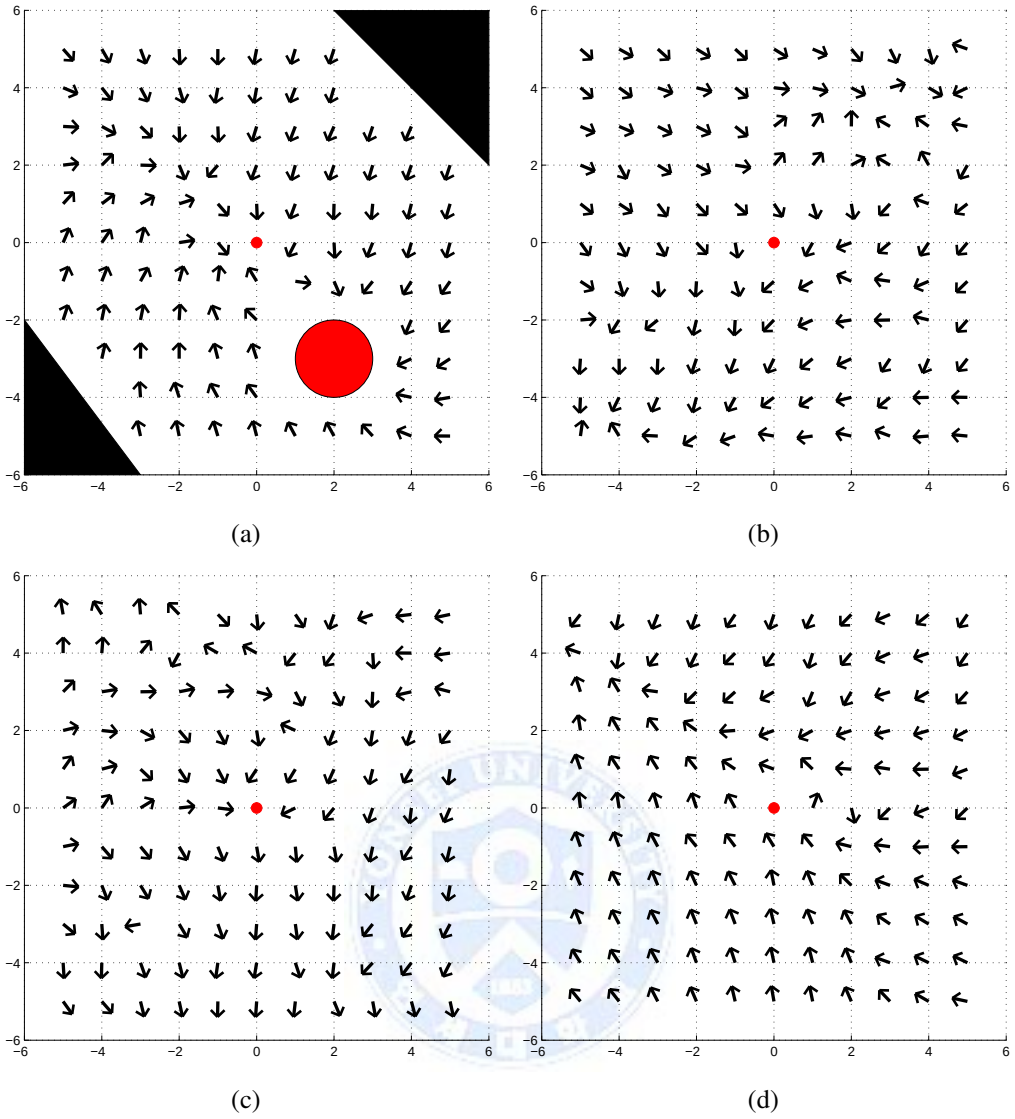


Figure 3.11: Vector map in different environments: (a) indoor environment; (b) outdoor environment; (c) moving people; and (d) featureless environment.

is obtained by a mathematical operation and does not change the HOG descriptors. This implies that the algorithm is robust against the blurring effect on an image when distinct gradients exist. We conduct another experiment in which we resize images to a certain percentage of the original ones and apply the nearest interpolation rule. With these images, we determine the homing direction using the HOG descriptors, and the performance is not highly degraded in accordance with the size; the overall performance is retained when using an image size that is 20 In a previous experiment, we found that there is an optimal cell size in which the correct homing direction can be found. To verify this, we compare the entropy level by sizes shown in figure 3.13 and

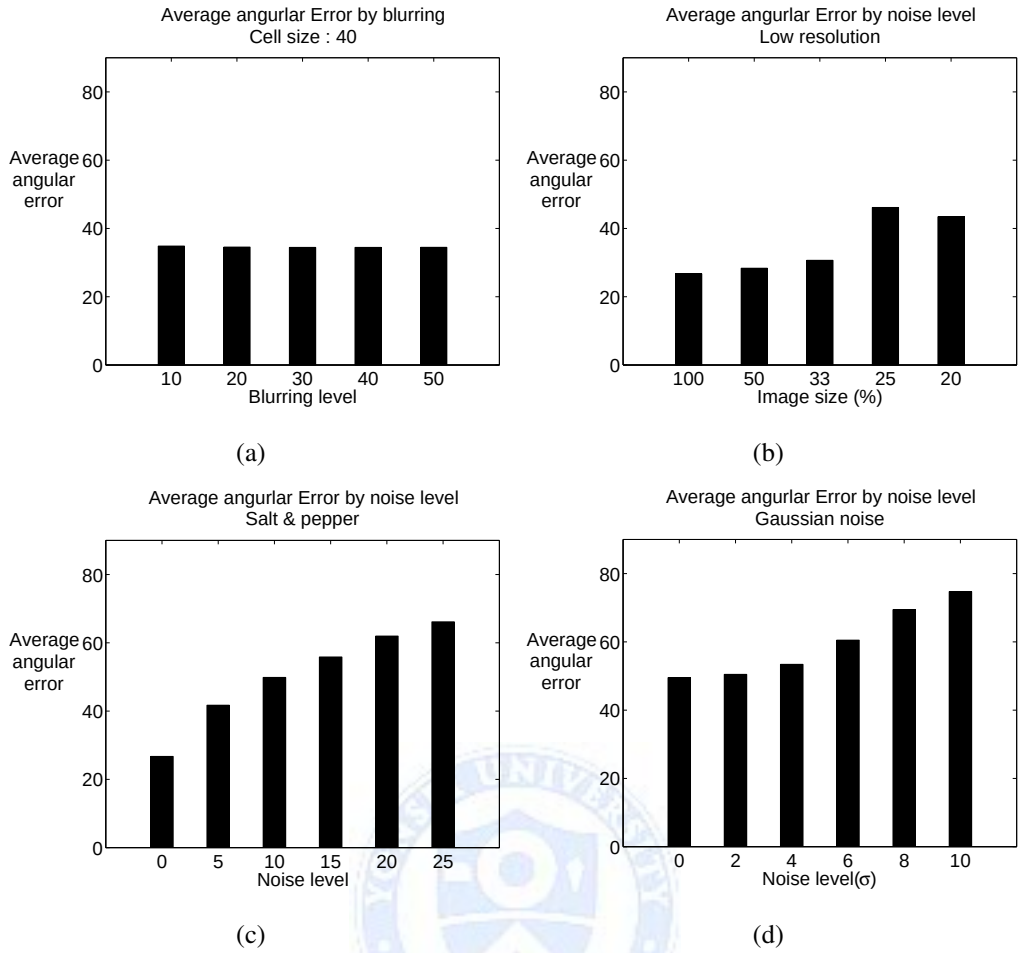


Figure 3.12: Average angular error by noise level with different noise types: (a) Gaussian blurring; (b) low resolution; (c) salt and pepper noise; and (d) Gaussian noise.

determine the maximum entropy level with a size of 40 or 60. There is no pattern in images when a small size is used, and the images become almost random. However, when a very large size is used, the bottom and top of the environment are included, and the images are pattern-less, causing a lower entropy level. Thus, as previously mentioned, there is an optimal cell size that can be used to determine the appropriate direction.

3.1.5 Illumination change

This algorithm using gradient feature can be affected by change of illumination. We gather snapshot images in an outdoor environment. We gather images for a long time, and it changes the illumination of environment. Mostly, existence of cloud cause a

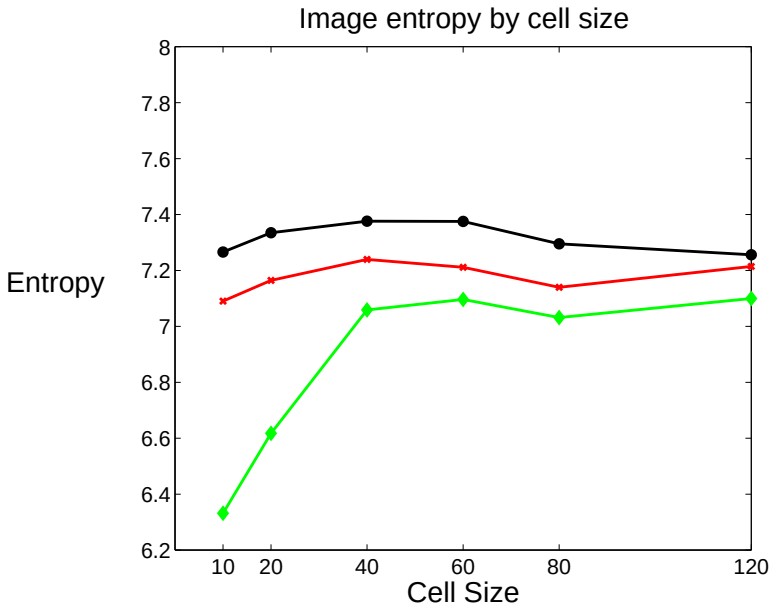
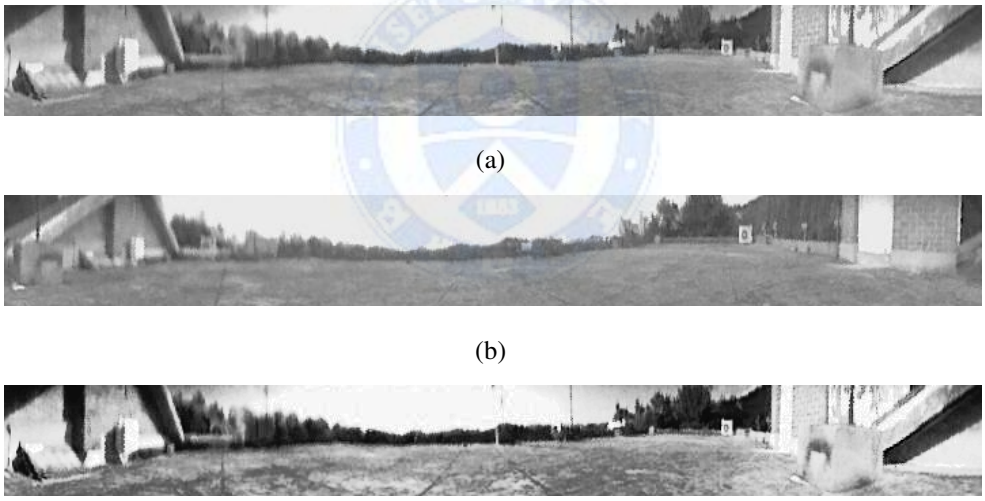


Figure 3.13: Entropy by size of cell in different environment: (yellow) indoor environment; (red) indoor environment (objects); (black) outdoor environment.



(a)

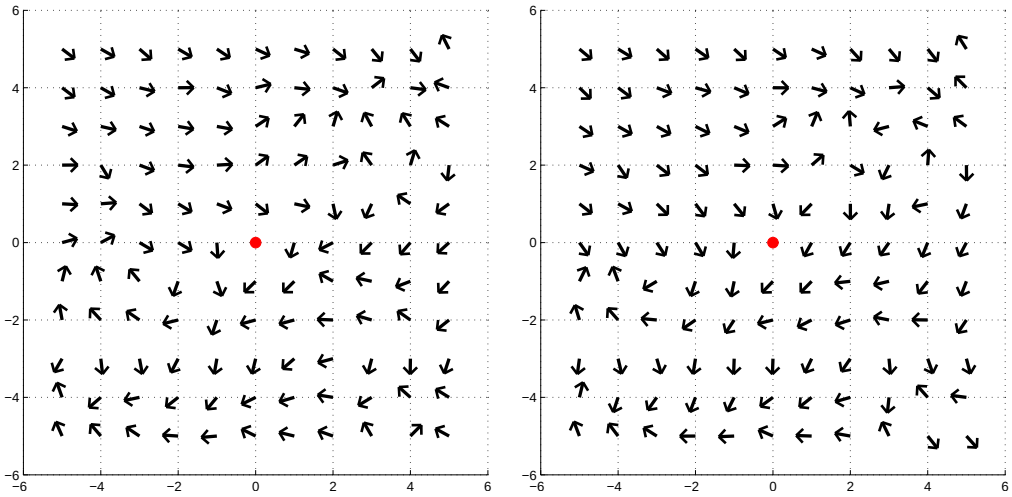
(b)

(c)

Figure 3.14: Illumination change of outdoor environment. (a) Environment with cloud, (b) Environment without cloud, (c) Applying the histogram equalization algorithm.

severe difference of illumination. Following figure shows this difference.

This environment has a clear change of sunlight. Without a cloud, most object has clear shape, and its shadow. When it is shaded, the overall scene is blurred. These shaded snapshot images are taken at left bottom of the overall environment. To reduce this difference, we apply commonly used algorithm, the histogram equalization algorithm.



(a) (b)

(c)

Figure 3.15: Average angular error by case. (a) Using the histogram equalization algorithm, (b) Using the Gaussian filter, (c) An example of defects from the histogram equalization algorithm.

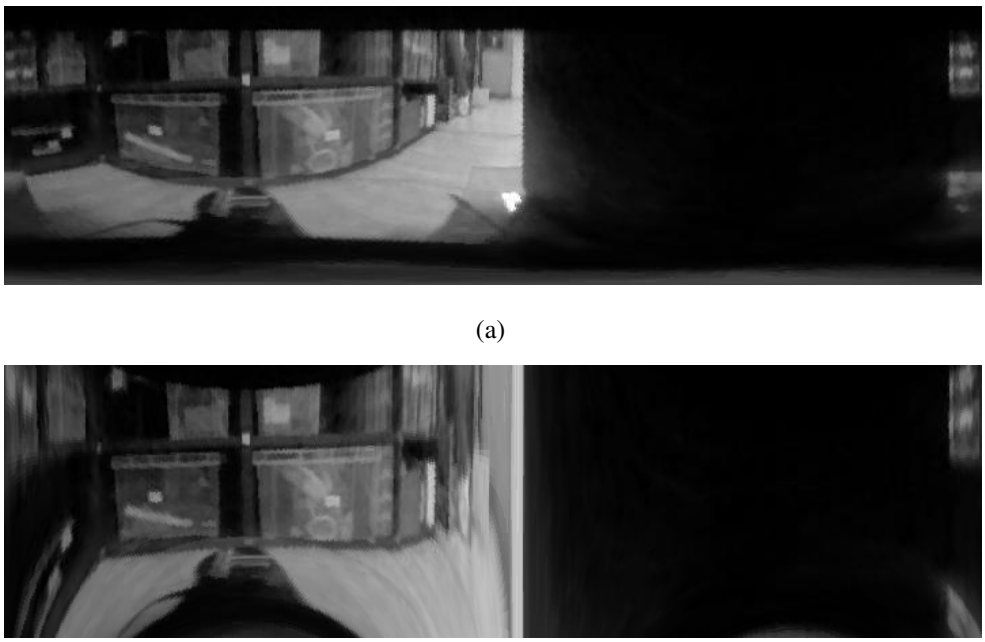
This algorithm compensates the illumination change, and enhance comparison of an image. We compare one using the histogram equalization algorithm in figure 3.11, and another without the algorithm in the figure 3.15.

We simulate this environment with illumination change with, and without the histogram equalization algorithm. It shows one without using the algorithm has better performance. When we apply the algorithm, its average angular error is 31.95°. Otherwise, it is 21.72°. We find why the degradation of overall performance is caused by using the histogram equalization algorithm in figure 3.15.

It shows details of a snapshot image. It shows a defect of the image is increased, and it also causes degradation of overall performance.

3.1.6 Image warping compensation

The omni directional camera uses parabolic mirrors to obtain surrounding environment. It has skewed image in figure 3.16(a). It shows straight parallel lines as curved



(a)

(b)

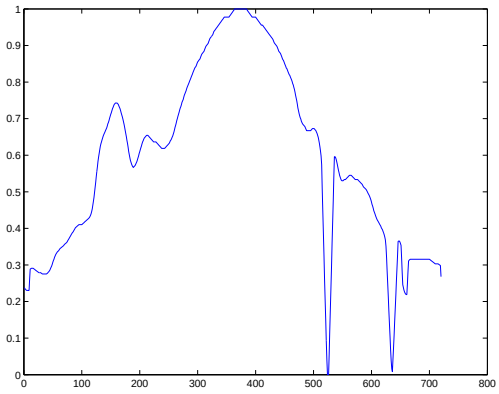
Figure 3.16: Warped image(a), and calibrated image(b).

lines. This warping relation of original straight lines, and skewed lines is depending on the angle difference between an object angle and focal angle, and distance between the object and the camera. A pixel in the image warped along y axis from warping effect. We have to find this relation between original heigh, or observed angle, of the original image, and warped image. In this figure, we can see a cabinet which has distant for the camera. It has a focal angle which is not changed parallel line by warping. It has unique focal angle decided by the curvature of a parabolic mirror. The difference of the focal angle, and pixel decide warping effect. If a pixel has distance to the focal angle in the original image, it moves far from the focal angle in warped image. There is another factor which decides the warping effect is the distance between the object and the camera. When it is closer than the others, it appeared bigger. This distance is critical to decide warping effect. This relation between pixel is following equation.

$$V_o = f(V_f - V_w, D)$$

Where V_o is observed angle of a pixel in an original image, V_f is focal angle, V_w is the observed angle of a pixel in a warped image, and D is the distance between the object and the camera. In the general case, we cannot find the distance to object, and we estimate this distance from image in figure 3.17.

(a)



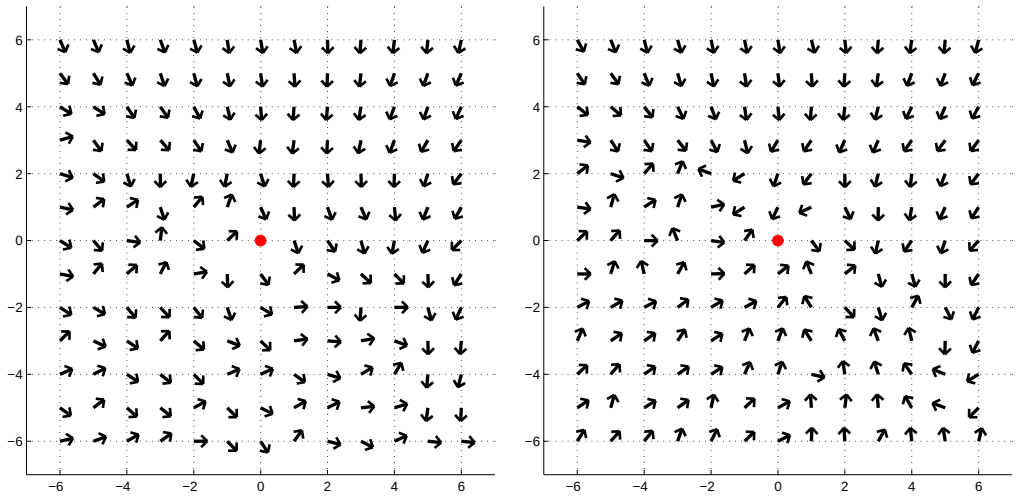
(b)

(c)

Figure 3.17: Warping process. (a) Edge of a snapshot image, (b) Estimated distance, (c) Calibrated image.

For each snapshot image, we find the edge image using the canny edge detection with blurring. It can provide bottom line of the overall object. We assume that this bottom line can represent the distance to object. From this bottom line, we estimate the distance to objects in figure 3.17(b). With this estimated distance information, and pixel information, we can find the calibrated image in figure 3.17(c). We can find a curved bottom line from the wall is compensated. We apply this algorithm to all snapshot images in a given environment. With these images, we apply our algorithm in figure 3.18.

In this figure, overall performance is degraded when we use 40 cell sizes. A case we use 60 cell sizes, we confirm overall performance is remaining. This degrade of performance is from the error of estimating the distance. For the same region of an image,



(a) (b)

Figure 3.18: Average angular error using calibrated images. (a) Using 40 cell size, (b) using 60 cell size.

we cannot find an edge, or bottom line properly, and it causes severe degradation of overall performance. This compensation has to be more accurate to find distance from object.

3.2 Summary of Chapter 3

In this section, we suggest navigation algorithms that have different feature extraction properties. In accordance with this, we suggest that HOG descriptors, SIFT, and SURF descriptors can be used as solutions for navigation problems. We consider these descriptors as landmarks, and show that it is possible to find the correct direction relative to the current location. We find that using specific bins for landmarks can deliver a better performance, and that vertical bins have a robust characteristic. We simulate this algorithm with different parameters. The size of the band image decides the overall performance, and we find that there is an optimal size when there is maximal information. In this section, a visual navigation algorithm is suggested using HOG features.

Chapter 4

Navigation with gradient feature on image

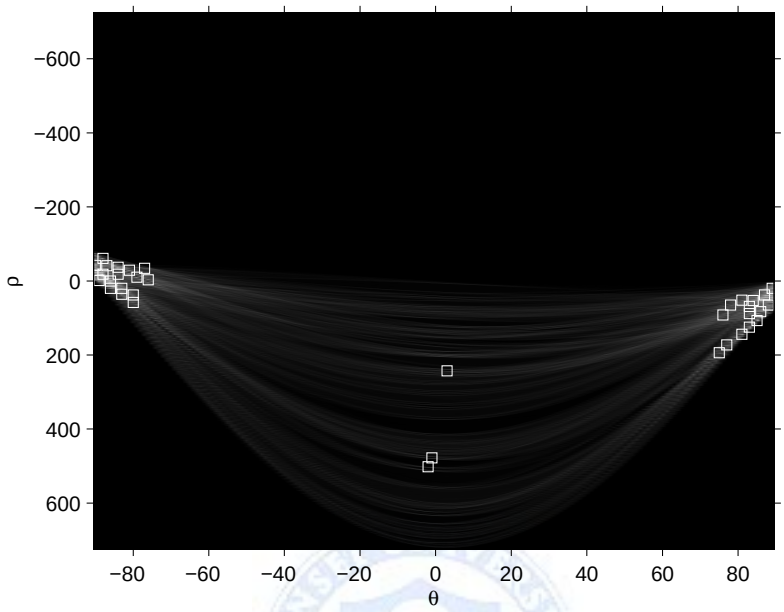
To find the homing direction, it is necessary to extract landmarks and corresponding landmarks. We suggest that the feature matching algorithm can be used in this respect. We simulate different feature matching algorithms to find the correspondence between landmarks, and apply the SIFT, the SURF, and vertical line features to find corresponding landmarks using visual information (Baek and Kim, 2014). This section is being prepared for submission as a scientific paper (Baek and Kim, 2015c).

4.1 Navigation with vertical line features

4.1.1 Vertical line detection from omnidirectional camera

A panoramic image is warped along the y-axis. The line at the bottom center of the image is originally straight line, but in the image it is warped. We thus consider the warping effect when we select the landmark (a line). Firstly, we obtain the edge of the image, and map this into the Hough coordinate (an example of this is shown in figure 4.1). It is necessary to have an edge image, and as such we apply canny edge detection to obtain the image in figure 4.1(a), which shows the edge image well. Many line segments are included here, and we thus map this image into the Hough coordinate. Figure 4.1 (b) shows the mapped edge on the Hough coordinate, where each line implies one edge point. With these mapped edges, we apply clustering and find the

(a)



(b)

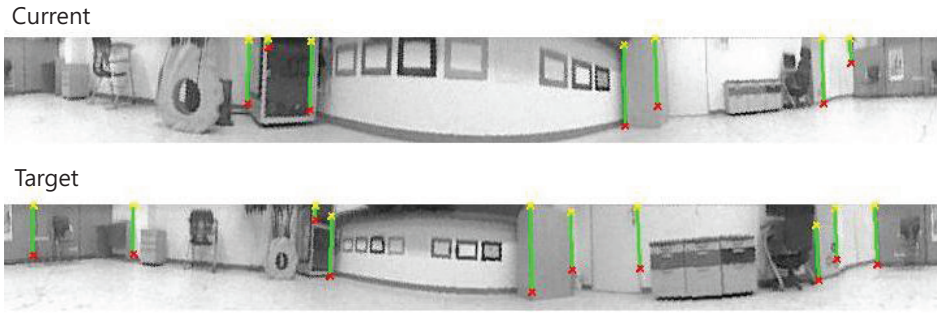
(c)

Figure 4.1: (a) Edge of image; (b) Mapping result and clustering; (c) Detected lines

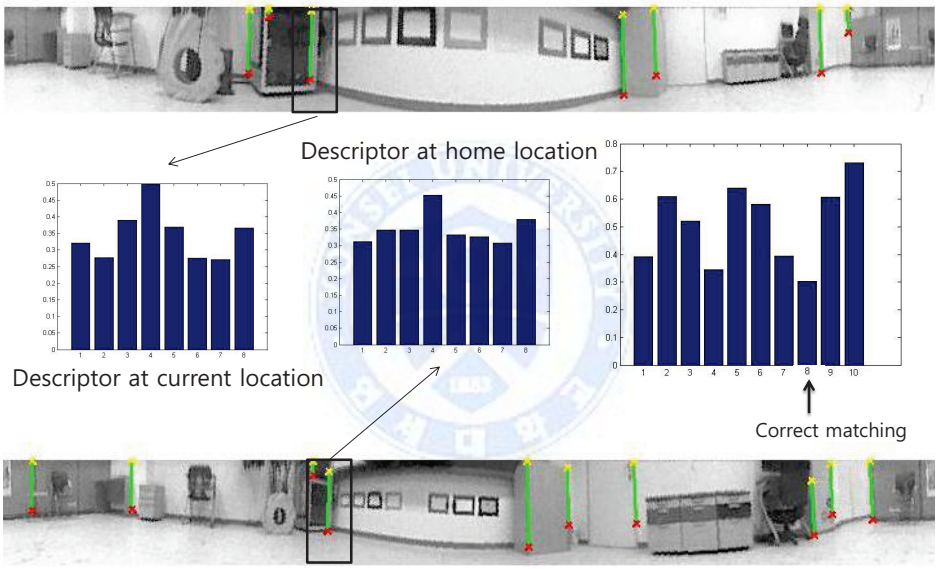
local maximum of the graph. The white boxes represent the local maximum points of the graph, where a number of line segments can be found. The result of line detection is shown in figure 4.1 (c), where it can be seen that too many line segments are included to use as landmarks. The landmarks need to have their own characteristics, but the lines are overlapped or not located in the right place.

In the previous section, we mentioned the warping effect of an image. The warping effect disrupts parallel lines and causes them to curve, which cannot be detected by the Hough transform. We thus need to find vertical lines rather than parallel lines. We threshold the lines by the theta value (the tangential angle). Through thresholding, all lines that disrupt the performance are removed. We need to determine whether line de-

tection is possible, and thus apply this algorithm into the image of the current and target locations. The detection result is shown in figure 4.2 (a). The result has many line seg-



(a)



(b)

Figure 4.2: (a) Detected vertical lines at current location, and target location. (b) Generated descriptor for corresponding lines; and finding minimal matching.

ments, and is thus shown as a long line divided into many line segments. We need to clustering these by its location. We have just one line for a reasonable line. These lines are considered to be related to the landmarks, and we can find detected lines within the corresponding lines. The lines at the side of electric box, and lecture desk. Some of the lines at the current location have disappeared, or have popped up in the home location. These differences can only be refined with knowledge of the landmark, and they are

caused by differences in the observing location where the observing angles of objects are altered. To determine the correspondence between landmarks at both locations, we apply the histogram from the oriented gradient technique. This technique generates descriptors of each landmark. A window is arbitrarily set for each detected line, with a width of 40 pixels and a height that is the same as that in the panoramic image. For this image, we apply the R-HOG filter to generate descriptors for each line, but do not divide the window into sub windows. Figure 4.2 (b) shows the clustered vertical lines at the current and home locations. We arbitrary pick a vertical line at the current location, and set a window around the line (which is represented by the black box on the panoramic image). For this region, we apply the R-HOG filter, and first histogram is generated descriptors for the line. There is a corresponding line at the home location, and we also generate the descriptors for this line. By comparing these descriptors, we ensure that the response is similar. We check that the approach is right for the line at the current location by comparing the descriptor for this line and for other lines from the home location. The distance between the descriptors is shown in the third histogram of figure 4.2. There is a minimum at the corresponding line (at 8). The index of the line is arbitrarily set, but the index of the corresponding point is at 8. There is also minimal point at 4, which is a line on the another side of an electric box. Such a similarity of windows can induce a degradation of the performance. However, we are able to find the corresponding landmarks using these descriptors, and the distance between other landmarks. We are also able to estimate the landmark correspondence between images.

4.1.2 Navigation with vertical line

If we construct the environment and take images from it, we can find vertical lines using the Hough transform and use these as landmarks. We apply the HOG descriptors to find the correspondence between landmarks. With these landmarks, we apply the average landmark vector to estimate the direction toward the target location, at the center of grid.

However, this result is not reliable at all and it implies that there is no information to use. More information is required, and we thus expand the window.

Figure 4.4 (a) shows that the result with the expanded window is better than that with the smaller window, although the performance is still not reliable. Figure 4.4 (b) shows

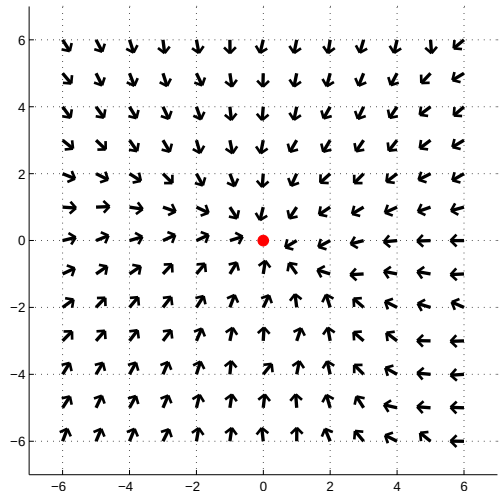
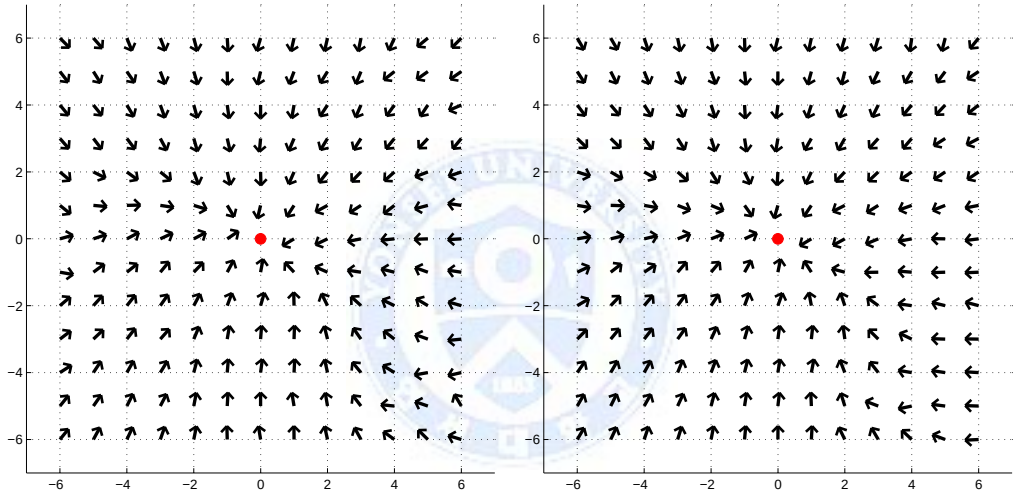


Figure 4.3: Using R-HOG response; window size is 40 pixels.



(a) (b)

Figure 4.4: (a) Using R-HOG response; window size is 80 pixels. (b) Using a different window scale.

the result using a different window scale, where a 40 pixel window was used and then resized. This difference in scale makes the change in resolution robust. However, although the result is improved it still includes a large number of wrong directions. We thus apply the different HOG filter, as the diagonal HOG filter is able to provide us with better landmark information. We find a landmark in the current location and pick a corresponding landmark in the home location. In the real image, a number of landmarks have disappeared, and some have popped up. As this operation contains weaknesses we therefore need to refine it, and to achieve this we pick a corresponding landmark at a minimal distance from the home location. For this chosen landmark, we again the

line with minimal distance among the current image. We compare these landmarks pick each other, than we think this matching is correct, and use this matching. If they are not choose each other, we waste these landmarks. With this, we expect that the result is improved. However, there is another available refining process. The correct matching of lines have not much difference of position along the x-axis, and most of wrong matching has a large displacement. We find the differences along the x-axis, and threshold them. If there is a matching line that has a displacement over the threshold along the x-axis, such landmarks are discarded. Using these refining processes we are able to obtain an overall better performance than before, and most the vectors point in the right direction. At some locations however, the vectors are pointed upwards. This is a default value, which appears when no matching lines exist, and can occur when a landmark from a current location has a corresponding landmark at the target location. To refine this result, we determine the minimal matching between the landmarks from the current location and the corresponding landmark, and this therefore represents the landmark at the current location. If another minimal landmark exists, it is not considered to be matching. However, this refining process can sometimes remove correct matching and can induce the default direction for some points. Another result using threshold shows that refining is able to improve the result. We have the problem that we already mentioned in relation to the refining process, which cannot be solved using the thresholding technique, although the overall performance is much improved. We therefore need greater accuracy to obtain robust navigational features, and thus apply another technique from the HOG descriptor. Generally, the HOG descriptor divides the window around interest points into many sub windows. We therefore apply this dividing process, which takes a little longer than before. The window is divided into six sub windows, and we have a total of 48 dimension descriptors for each line. In this case, we do not use any other features, such as the diagonal HOG or a different scale. This results show a considerable improvement in the performance. By expanding the dimension of the descriptor the process is considerably improved. We thus apply the refining process, and a thresholding technique with an expanded process. These processes are much more complex than when only using the HOG descriptor, but the performance is drastically improved. However, the result still shows a default direction at some points, particularly on the right hand side of the environment. Therefore, to overcome this problem we loose the refining process. For a landmark in the current location, we find a corresponding landmark in the target location, and for this landmark we find the minimal distance between it and the current location. We

then check matching on both sides. However, this refining process is too strict to find matching lines in some locations, and therefore we loose this refining process. If the corresponding points has second minimal matching at current location, we also consider this matching is correct. We have accurate performance to find the right direction toward target location. Find the corresponding landmarks and direction delivers some reliability.

4.2 Navigation with SIFT features

4.2.1 SIFT features & navigation

We firstly obtain images from within the environments. From the images of the home and current locations, we find matching SIFT features, as shown in figure 4.5. From

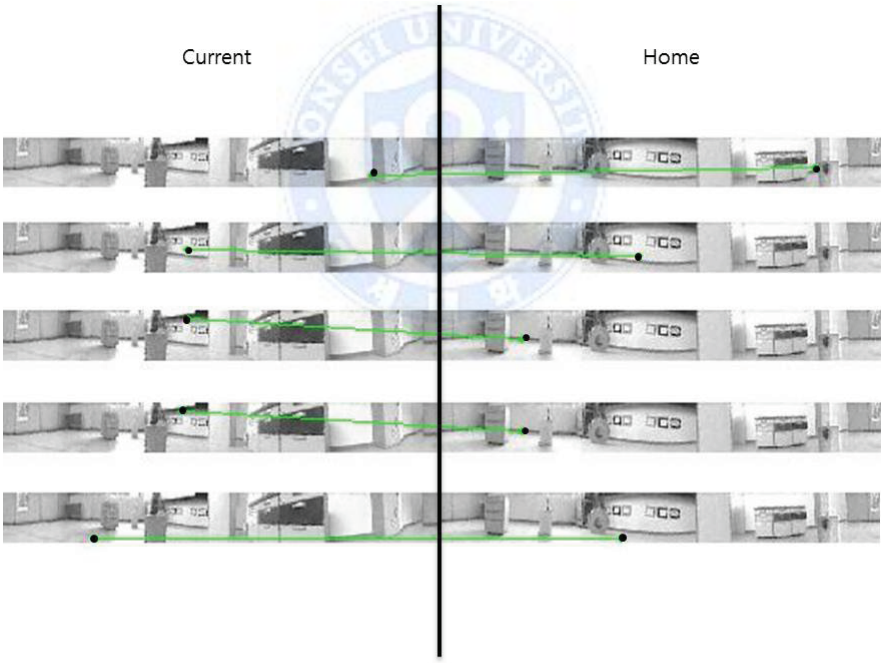


Figure 4.5: Descriptor matching by SIFT algorithm.

the images, we choose five matching features where each matching features implies a corresponding landmark. However, not all matching features are correct. We consider these as observed landmarks using the ALV algorithm, and we estimate the homing direction (figure 4.6).

Results are shown in vector map form. We calculate the homing direction for each

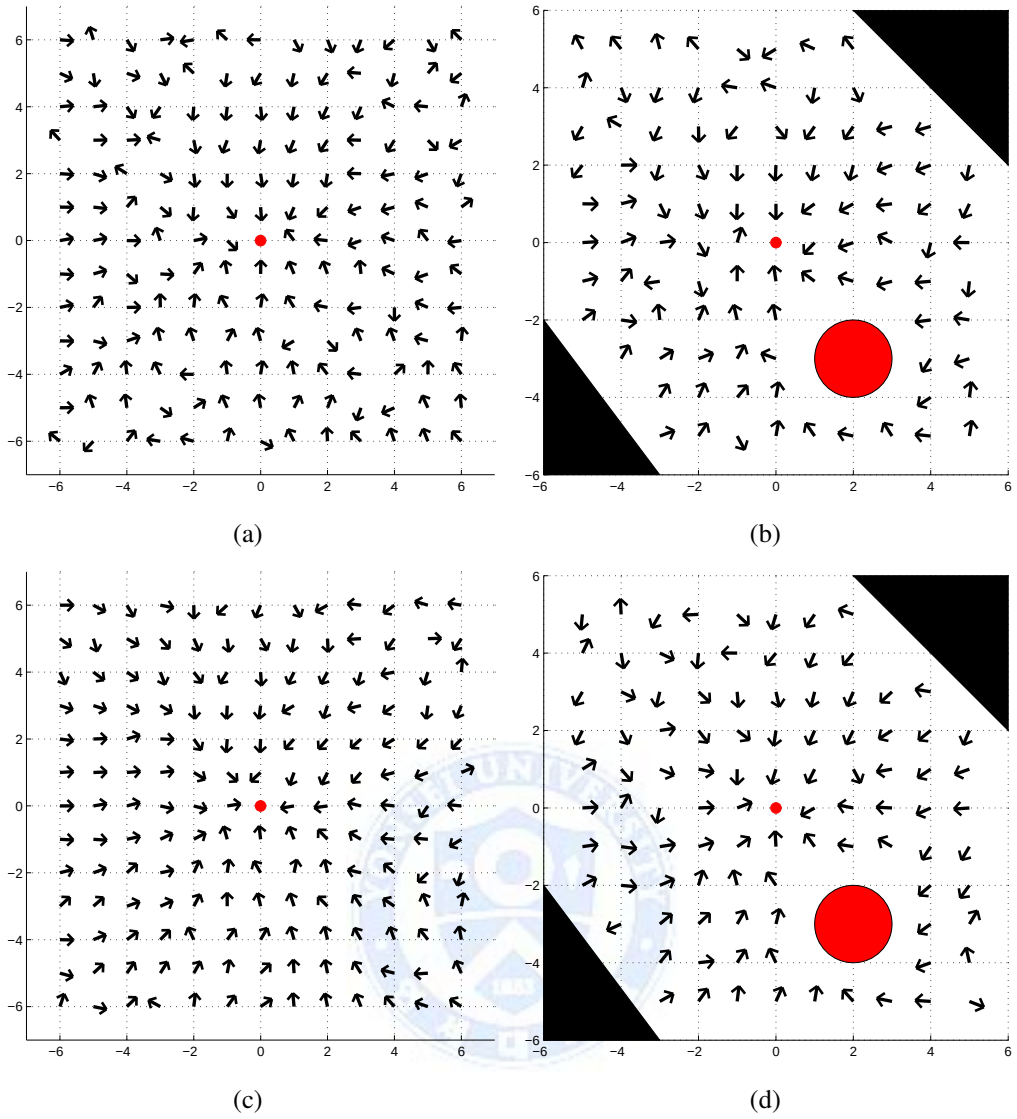


Figure 4.6: Performance of navigation using the SIFT algorithm shown in a vector map: (a) in environment; (b) and environment2 with five landmarks, and (c) in environment; and (d) environment2 using all landmarks with the SIFT algorithm.

point and draw this with an arrow. The home location is set at the center of the field, and thus the optimal vectors are directed toward the center of the field. The navigation result is not adequate to determine the homing direction. In homogeneous environment, this navigation works around the home location, and at close range this algorithm can work. The arrow directed toward home is directed well with a tolerable error. When there is a large distance to the home location, the algorithm performs badly and there are some errors related to mismatching by the SIFT algorithm, which occurs when an object is severely distorted by the difference in location and can de-

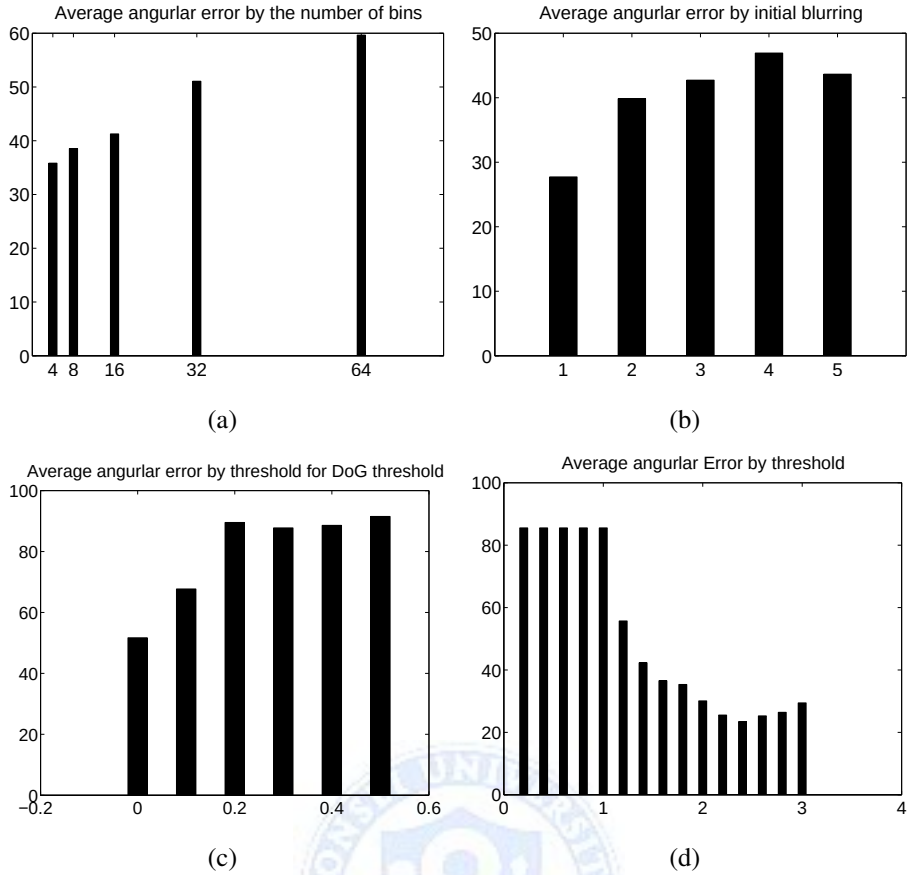
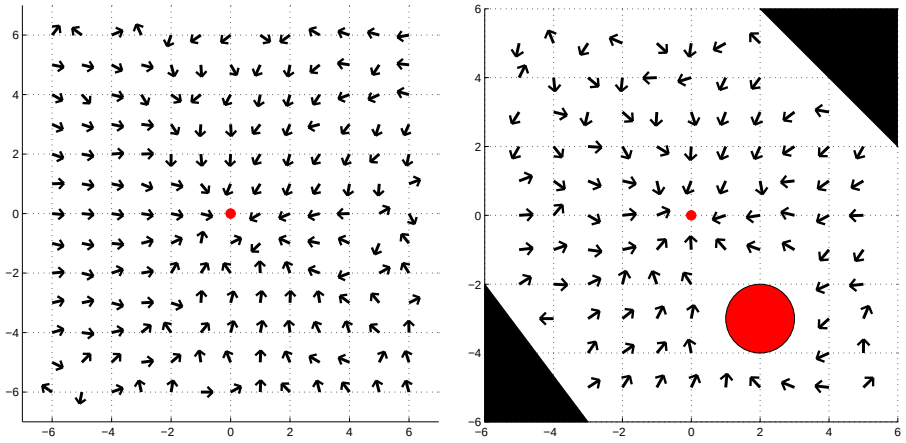


Figure 4.7: Navigation performance by parameters. (a) Number of bins for SIFT descriptors; (b) Gaussian blurring level; (c) descriptor threshold value; (d) matching threshold value.

grade the performance.

There are many objects inside Environment, which causes an occlusion and a number of mismatching features. Ultimately it causes a degradation of the overall performance. A different method of navigation is required. We need to find some other way to navigate. We arbitrary pick just five landmarks, and the mismatching of landmark cause severe degrade of performance. A mismatching have much proportion of total matching. Thus, it is confirmed that when a few landmarks are used and mismatched, tremendous errors can occur at each point. Thus, more landmarks are required for navigation. We calculate the difference in the descriptor for each matching, set a threshold for each matching, and use all landmarks above the threshold. We set optimal parameters for the SIFT feature which decide whether mismatching features exist or not. Some parameters remove noise on the image (their overall performance is shown in



(a) (b)

Figure 4.8: Navigation performance in vector map: (a) in environment1; and (b) in environment2 using the weighted algorithm.

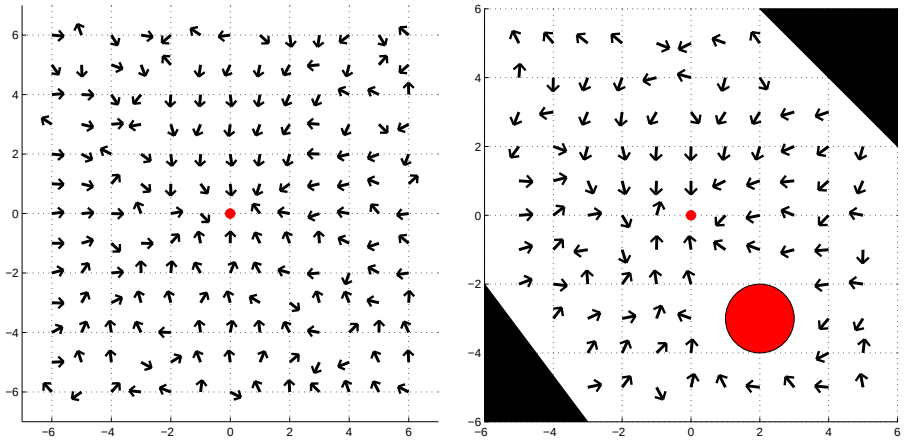
figure 4.7). Although it is evident that the performance is better than that of environment2, the performance is still not good enough. There is some mismatching using the SIFT algorithm. Therefore, we apply another feature of the matching algorithm to find the homing direction.

4.2.2 Weighted algorithm

To find a correct homing direction using the SIFT features, we need to remove or reduce the mismatching features causing a degradation of the overall performance. We need to decide whether each matching feature is correct or not, using the difference between matching features, and choosing the most possible matching features. We then also apply this measure to the ALV algorithm. The algorithm for the landmark vector uses a unit vector toward the observed landmarks. Instead of using a unit vector, we decide the magnitude by the difference between matching features. This modified algorithm is the following equation :

$$\overrightarrow{ALV}_{Home} = \sum_{n=1}^{N} Q_n LV_n$$

$$\overrightarrow{ALV}_{Curr} = \sum_{n=1}^{N} Q_n LV_n$$



(a) (b)

Figure 4.9: Navigation performance on a vector map: (a) in environment1; and in (b) environment2, using a scale compensated algorithm.

where Q_n is the difference between matching features. Using this algorithm it is possible to find the hom-ing direction (figure 4.8). There is a less average angular error than when the weighted algorithm is used, but the overall performance is not much changed. This implies that we cannot decide whether the matching features are correct or not using this measure.

4.2.3 Scale compensation

This algorithm is unable to consider the distance to landmarks and assumes that the distance between landmarks and a location are all equal. This assumption is not always satisfied, and therefore the overall performance is degraded. Finding the distance to landmarks is a challenging problem, but there is a way to compensate this difference using the ratio of distances. This ALV algorithm does not need the exact distance to landmarks. A ratio of the distance between the current location and a landmark, and the distance between the home location and the landmark is required to compensate this difference. We are able to estimate this ratio using the difference in the scale space between matching features. We compensate estimating direction to home with this ratio in figure 4.9. The results show that the overall performance is much better than the one that is not compensated, and its average angular error of estimating the homing direction is reduced. This thus shows that compensation using a scale difference can be used as a solution for the absence of distance information.

4.3 Navigation with SURF features

4.3.1 The SURF features & navigation

With confirm that the SURF algorithm is also able to adequately perform descriptor matching. Firstly, we select five matches (the result is shown in figure 4.10. where the performance is shown to have improved relative to the previous simulation). We expect that using a greater number of landmarks will deliver an enhanced performance. However, the result of this test is shown in figure 4.10, where it is confirmed that the

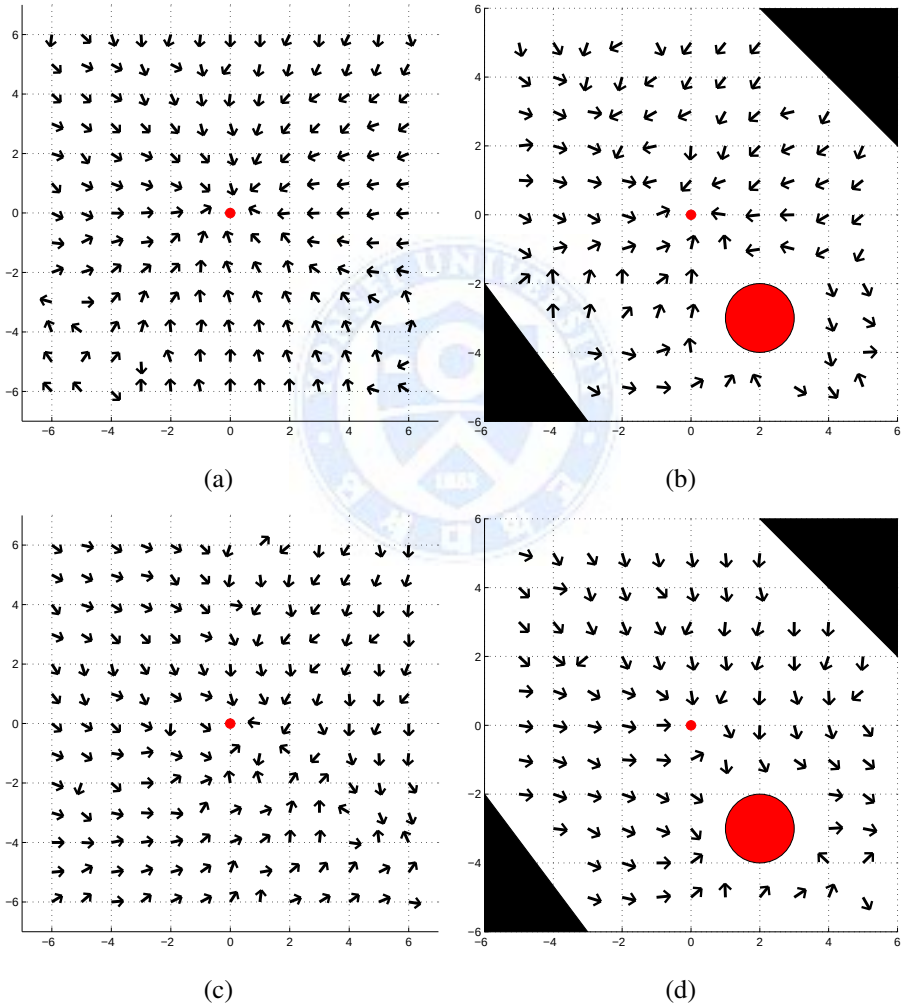


Figure 4.10: Navigation performance shown on a vector map: (a) in environment1; and (b) environment2 using five landmarks with the SURF algorithm. (c) in environment1; and (d) in environment2 using all matching pairs as landmarks with the SURF algorithm.

performance is degraded. We thus analyze this result to determine the cause, and find

that the degradation is caused by mismatching. With a greater number of landmarks, there are a greater number of mismatched landmarks, which degrades the performance. There are less mismatched landmarks using the SURF algorithm, and it is possible to use only three landmarks to determine the homing direction. As previously mentioned, with more landmarks there are a greater number of errors. This result is in contrast to the previous simulation using the SIFT algorithm. We consider that five landmarks are not adequate for use with the SIFT algorithm, and there is thus an optimal number of landmarks.

4.3.2 Weighted algorithm

In the previous section, we showed the navigation performance in relation to the number of landmarks used. The performance of this navigation algorithm was found to be degraded by mismatching. We therefore suggest an alternative method for determining the homing direction. In the previous section, we applied the average landmark vector algorithm to determine the homing direction, which drew the unit vector toward the observed landmarks. Instead of draw a unit vector, we differ magnitude of the vector. We need to therefore decide its magnitude. Fortunately, we are able to calculate the matching score for each matching descriptor, which can be used to consider whether this matching is correct. This score does not always work, but we can still use this factor to determine the similarity of matching. We assume that a matching with higher score have more possibility to correct. We can enhance the effect by matching with a higher score and reduce the disturbance of mismatching by setting the magnitude of a landmark vector to its matching score. We thus apply the SURF algorithm to match the descriptors. We also compare the effect of the number of landmarks used and expect that the performance would be improved using a larger number of landmarks. Figure 4.11 shows the performance when five landmarks are used. However, the performance of this algorithm is found to be the same as before, and we consider that it is difficult to improve the previous result gained using five landmarks. In some locations, the angular error was slightly increased when using the weighted ALV algorithm. We thus apply this weighted algorithm using 20 landmarks (this result is shown in figure 4.11). We then apply the weighted average landmark vector algorithm, and find that the performance is very much improved. In most locations, this algorithm was able to direct toward the homing direction. Although the performance is no better than when using five landmarks, we expect that this method is more robust in relation

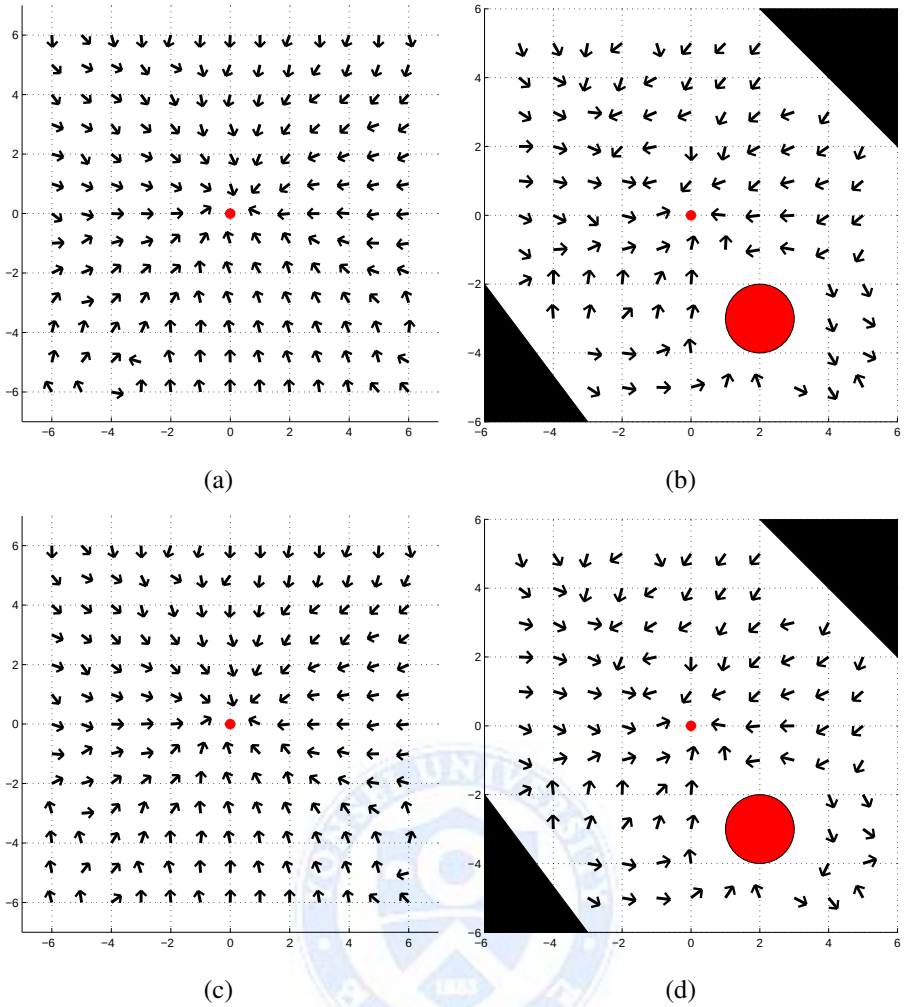


Figure 4.11: Navigation performance on a vector map: (a) in environment1; and (b) in environment2 using 5 landmarks with the SURF algorithm applied to a weighted ALV algorithm. (c) in environment1; and (d) in environment2 using 5 landmarks with the SURF algorithm applied to a scale compensated ALV algorithm.

external factors, such as mismatching. For example, if one mismatched landmark exists, another correctly matched landmark can revise the error; for example, even with the occlusion in environment2, the homing direction was correctly identified. Therefore, it is possible to reduce the effect of mismatching by applying the weighted ALV algorithm.

4.4 Localization using vertical feature distribution

In the previous section, we extract vertical lines from images. We suggest an algorithm to find its current location using its distribution. We have vertical lines in figure 4.12(a). We set a window around extracted vertical lines. Figure 4.12 (b) shows examples of it. We can find distinctive window templates to find a unique property for each

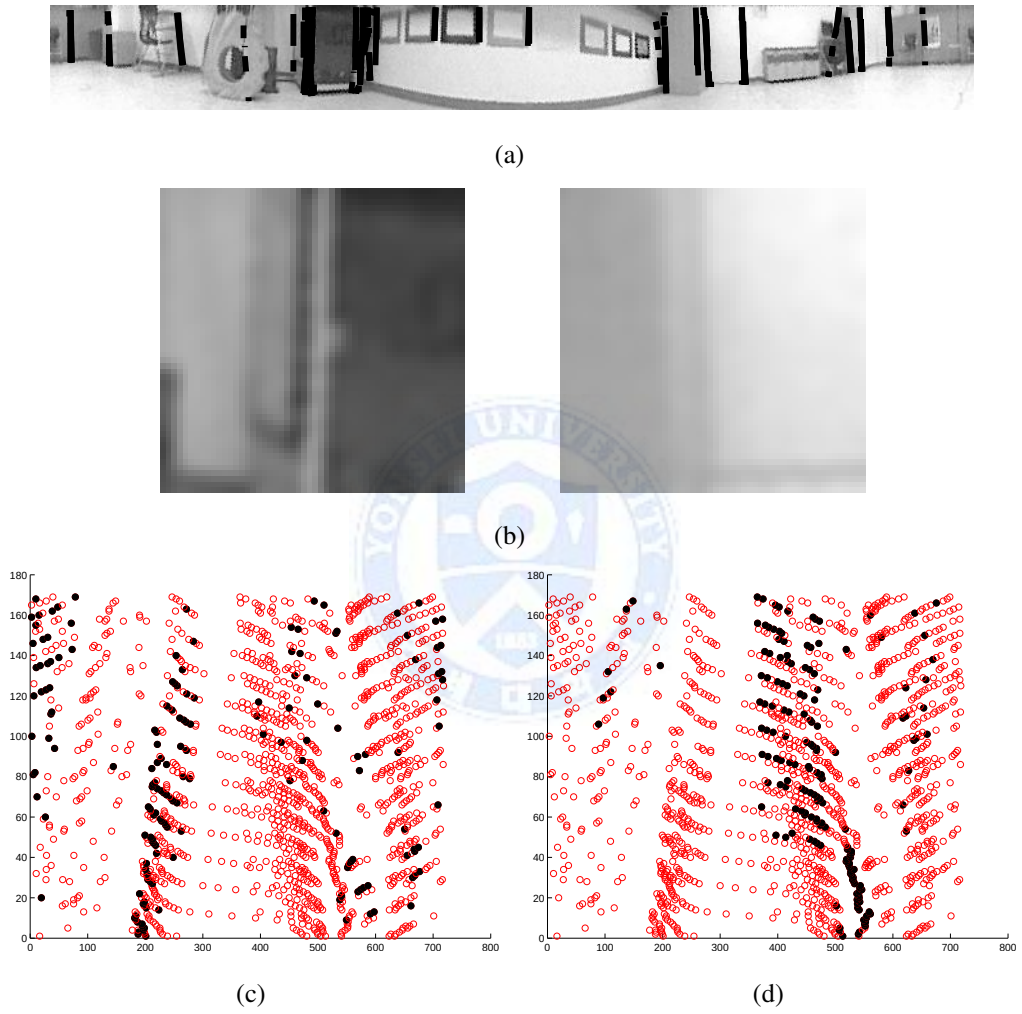


Figure 4.12: Extract vertical line information. (a) Extracted vertical lines, (b) Examples of template around vertical lines, (c), (d) Template responses.

location. Figure 4.12 (c), (d) shows its responses for each template. In this graph, x-axis implies observed angle of vertical lines, and y-axis implies index of locations in the environment. We have total 169 environments. For each location, we find a vertical line which has most similarity to the given template image. The red circle in the graph implies observed vertical lines, and the black dot implies that most similar

vertical line with the given template. We can see this response have characteristic. Only one response of one template cannot show unique property, but a combination of them can represent each location. We use 6 template images to describe response for each location.

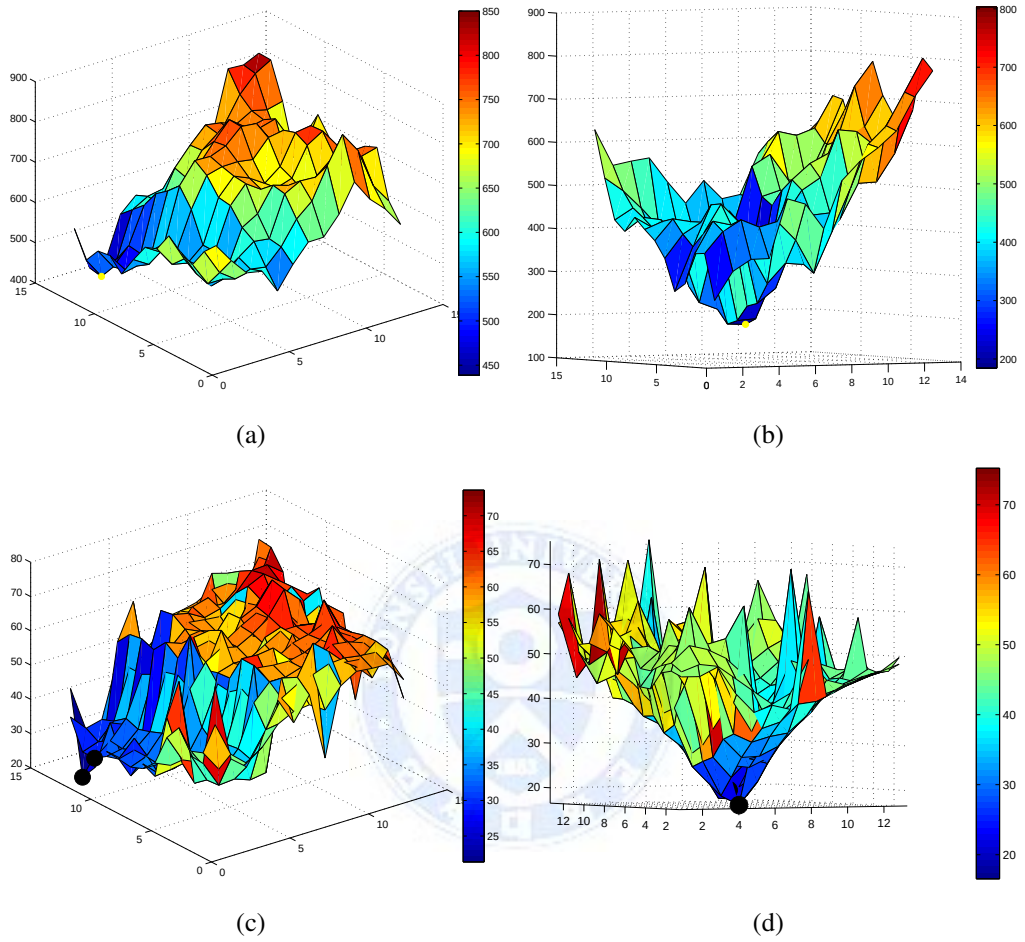
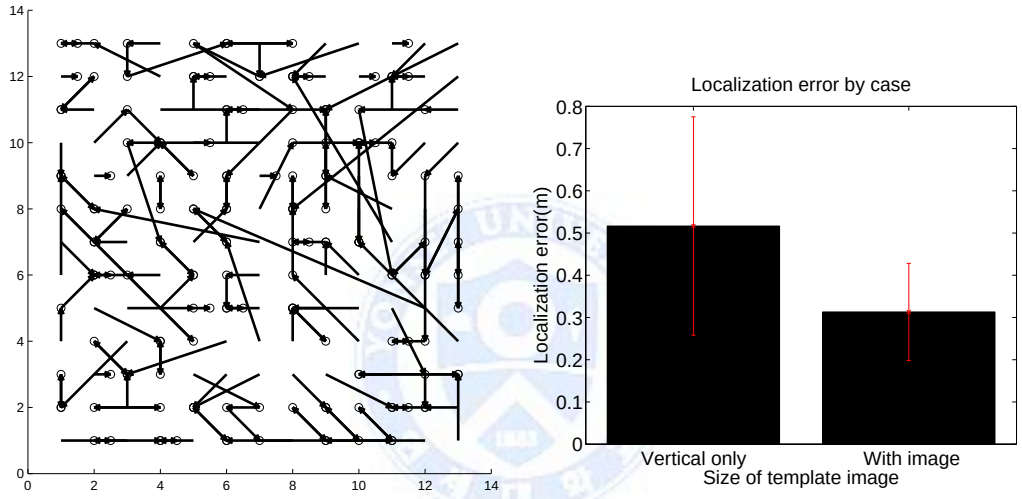


Figure 4.13: Distance between descriptors from current location, and other locations. (a) Current location is at a corner of environment, (b) Current location is at a center of environment, (c), (d) Image difference for each case.

Figure 4.13 (a), (b) show difference between descriptors of target location, and others. We can find there is the minimum area around the target location. We can find there is a convex like distance which guarantee the convergence. We simulate this localization with different location. To find current location, it does not compare with descriptors from current location. Even it does not have its current data, it can estimate its current location with a small error. We compare overall performance with different size of template images in figure 4.13(c). We can estimate its current location with a small error. With this comparison, it shows larger size of the template guarantee better per-

formance. In this experiment, we show its current location can be estimated using a distribution of vertical edge features of the images. This estimation has a relatively large localization error, and we use additional feature which also depends on visual information, and difference of images (Franz et al., 1998). The difference of two given images is increasing with distance between locations where they have taken. We also use this measure with distribution of vertical lines to compensate the error from vertical lines. With this information from an image, we can localize its current location in figure **??**(a). With this distribution of vertical lines, we can find its current loca-



(a) (b)

Figure 4.14: (a) Localization result in given environment. (b) Localization error with, and without image difference information.

tion by comparing its distribution with the dataset. It has some localization error from mismatching of giving template, and lack of information for distribution. For some locations with higher error, response of template image is moderate, and it is not distinctive. It causes an error from localization. In some case, it does not have enough vertical segments in an image. It also cause an error severely. We find there is few minimum points at far from some locations which has a similar pattern of vertical lines. We remove them by using the difference between images. By combining information from vertical lines, and from images, we can improve its overall performance.

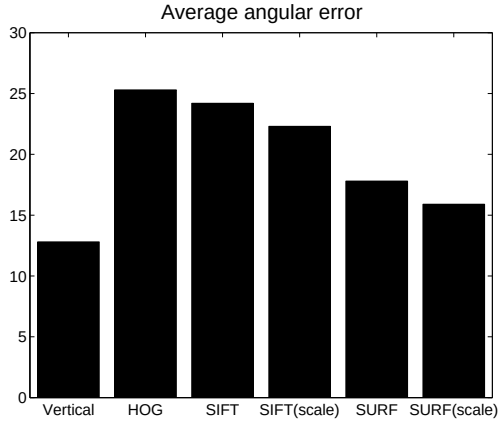


Figure 4.15: Average angular error for each algorithm.

4.5 Summary of Chapter 4

The challenging problem related to navigation is accurately determining the location, and this localization problem can be solved using visual information. Plenty of research has been conducted using a visual dependent localization system. However, there is another approach that can be used to solve this problem, and this is based on the activity of rodents brains. The hippocampus of the brain of rodents is used to determine their current location. We modelled our study on the brains of rodents and thus our model uses images in a dataset before it performs its navigation task. Using this information it can find the current location by comparing the current images with those in the dataset. This procedure of comparison needs to be reasonable, and we therefore suggested an algorithm to compare with the SIFT features and its matching procedure. This SIFT algorithm provides the corresponding points on images. These features are extremely important and are very robust to occlusion. We therefore used a feature-based algorithm to compare these images. With SIFT matching, we suggested a measuring distance for use in determining the similarity between images. With this similarity information, the current location can be determined with a high accuracy. We conducted many experiments to show that this algorithm works in occluded environments. We also suggest that this algorithm is able to find its current location and it can navigate along the locations we set. This deciding procedure have an important shortage to find current location with similarity measure. Which is it includes most of times. We analyzed this error, and show this error can be compensated by its moving. Thus, this error is not increasing by its movement. This error level is maintained by its moving. The error is decided by the density of place cells in a given environment.

We also simulated the random generating of these place cells, when we chose premediated random movement and gathered images during random movement. We decided whether to use the current image or not by similarity between a current image, and images in dataset. The results depended on a threshold value that decides whether a current image is similar or not. The effect of this threshold value was determined, and the overall density of place cells was decided. Using a different distribution of the place cells, we compared the overall localization performance in figure 4.15. Using vertical feature can be a accurate solution for the local navigation problem which consume more computation. Otherwise, using the HOG descriptor have inferior performance, but it does not requires complex calculation to find homing direction than other algorithms. Each algorithm have its own advantages, and shortage. We also suggested an algorithm that enabled determining the current location using visual information and a previously explored memory.

Chapter 5

Feature based navigation for the SLAM problem

We expand this concept for use in self-localization. The snapshot images are taken with an omnidirectional camera using the previously explored environment, and the feature-matching algorithm is applied to enable the robot to find the current location. To determine the current location, we find a similarity between the images, and using the current location, we are able to navigate the robot with a small localization error. This section is being prepared for submission as a scientific paper (Baek and Kim, 2015a).

5.1 Feature matching based localization

5.1.1 Image similarity

We gather a large number of images within environments at different locations and build SIFT descriptors for each image (as generated in figure 5.1). These generated descriptors depend on edge thresholding. The threshold value is decided in relation to whether points are interest points or not. Using this threshold value, the number of generated interest points is altered. The threshold value enables different candidates to be used for the matching process, but the matching results do not differ greatly using this threshold value. In the figures, the first, second, and third images are those generated by the SIFT descriptors with different threshold values, and the fourth and fifth im-

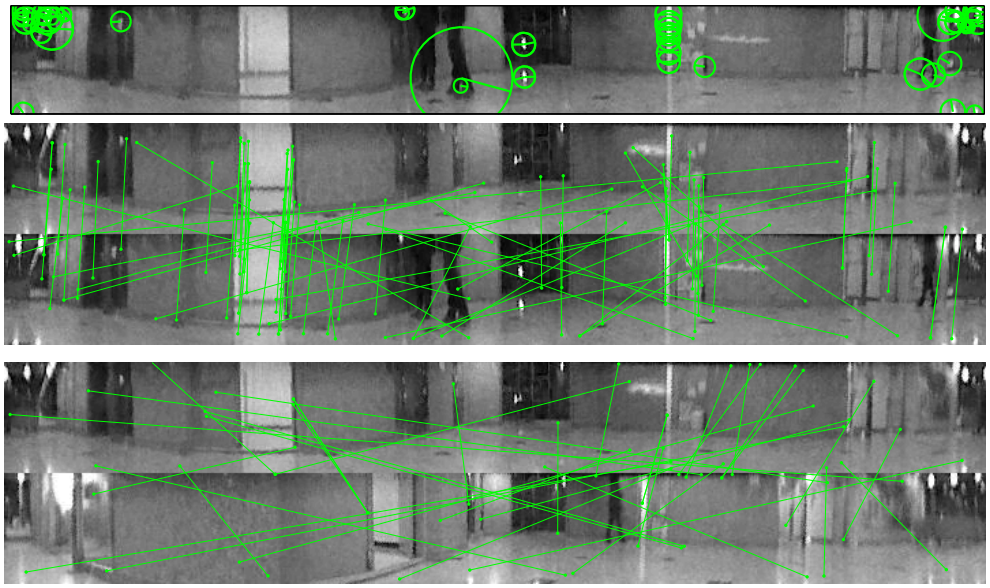


Figure 5.1: SIFT descriptors generated with different edge threshold values. SIFT matching result between two images from near locations, and from distant locations.

ages are the matching results. Different matching exists with different threshold values when generating the SIFT descriptors. The fourth image shows SIFT matching with two images from near locations, and with occlusions related to people. The fifth image gives the same result using images from distant locations. These results have reasonable matching, but still some of the matching interest point pairs are wrong. Many algorithms can be used to refine these matching pairs of points, and one well-known algorithm is the random sample consensus to detect outliers, otherwise known as the wrong matching pairs of points (Fischler and Bolles, 1981). However, this algorithm has a high calculation complexity when used to find proper matching.

Instead of using the RANSAC algorithm to find correct matching, we can compare the descriptors themselves. In figure 5.2, the graphs represent the distribution of difference between the descriptors of matching pairs of interest points. Many matching pairs have large differences, but there are also many matching pairs with small differences. Therefore, we determine these differences and choose those that have the smallest matching, as we consider we are more likely to be able to correct these. In the figure, matching shows the results. Most of them are correct on both cases. Although RANSAC can guarantee correct matching, it has a higher calculation complexity than when descriptors are compared. We therefore use the comparison of descriptors to determine correct matching pairs. This result also implies another important fact, that there are still mismatched pairs of interest points in both results, but that the mis-

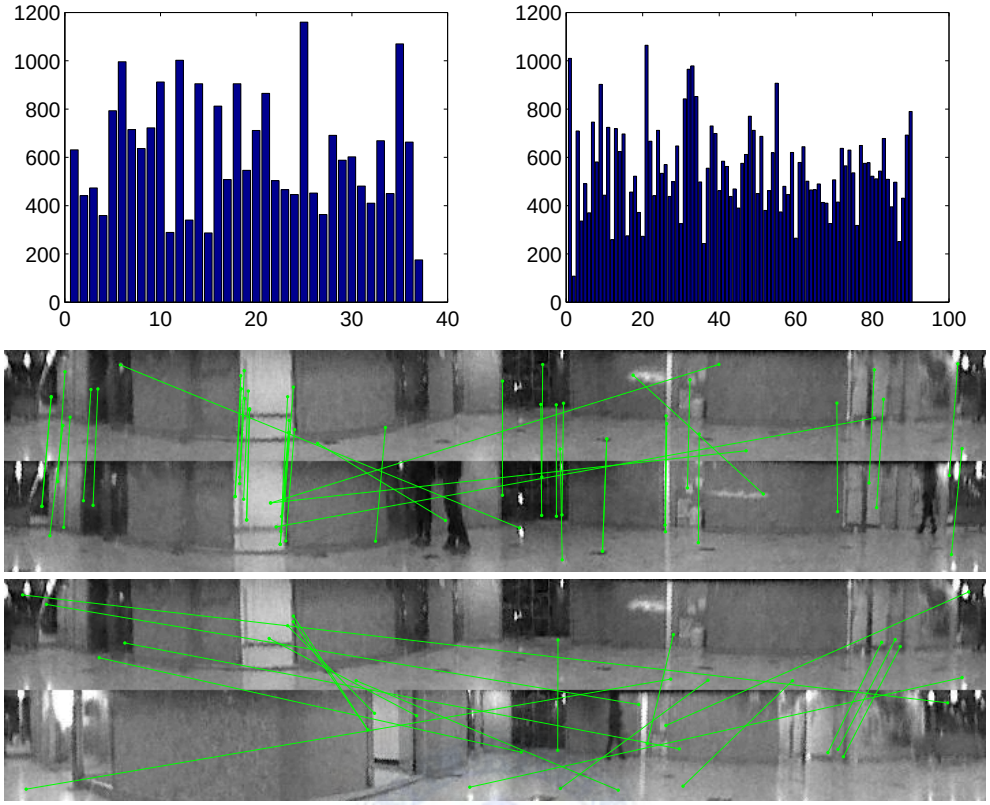


Figure 5.2: Distribution of difference of SIFT descriptors for each matching pairs of points. Refined results when choosing matching pairs of interest points with the smallest difference between descriptors.

matching is greater with distant images. In contrast, images from close locations have a small amount of mismatching. Using these matching pairs of interest points, we find similarity between the images, and these mismatching pairs can be compensated in the process of calculating similarity. This similarity measure uses similar close images and different distant images and follows the method outlined below. We find corresponding points between two images using the SIFT algorithm and consider that these points are detected landmarks on the image. If these corresponding points are correct, there is a relationship between the detected angles in figure 5.3. A target image is found at the bottom of the figure, and two locations with different distance to target location. The following equation is then used:

$$D_1 < D_2$$

For each location, we detect the same arbitrary landmark using the SIFT feature matching algorithm. With these detected matching pairs of interesting points, we can then find a relationship between the detected angle. For the i-th landmark, the following

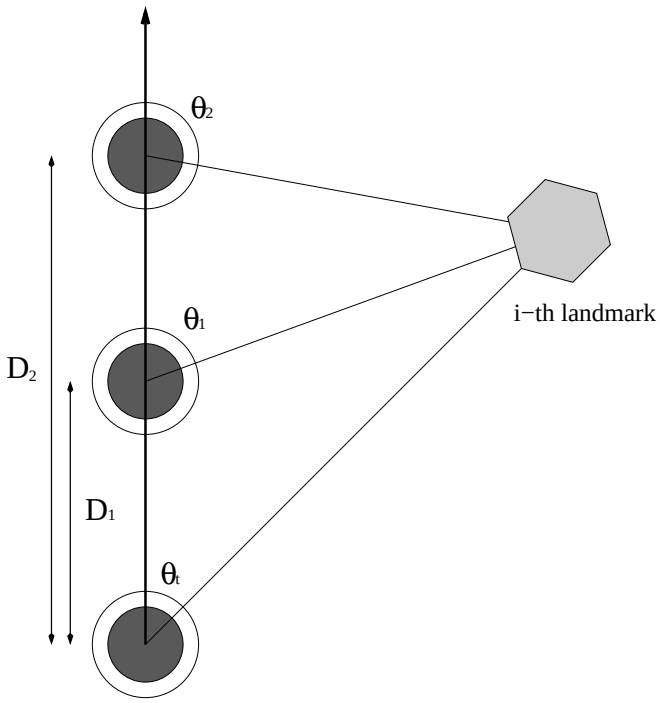


Figure 5.3: Landmark and detected angle for each location. Target location (at the bottom), close location (middle location), and distant location (top location).

relationships are determined:

$$\|\theta_1 - \theta_t\| < \|\theta_2 - \theta_t\|$$
$$f_i(D) = \|\theta_D - \theta_t\|$$
$$\sum_{i=1}^{N} f_i(D_1) < \sum_{i=1}^{N} f_i(D_2)$$

From the above relationships, the detected angle difference $(\theta_1 - \theta_t)$ depends on many parameters, such as the distances between location 1, t and landmark, D_1, and θ_t. This detected angle difference is increased by the increasing distance between locations, which is not linear but is monotonically increasing. When there are N detected landmarks, the sum of the detected angles satisfies the above relationship, which assumes the same detected landmarks. However, the same detected landmarks are not always apparent on the images using SIFT matching algorithm, and at times there are a different number of matching pairs of landmarks. We assume that there are N matched landmarks between location 1, and location t, and M matched landmarks between lo-

cation 2, and location t. With these landmarks, the following relationship exists,

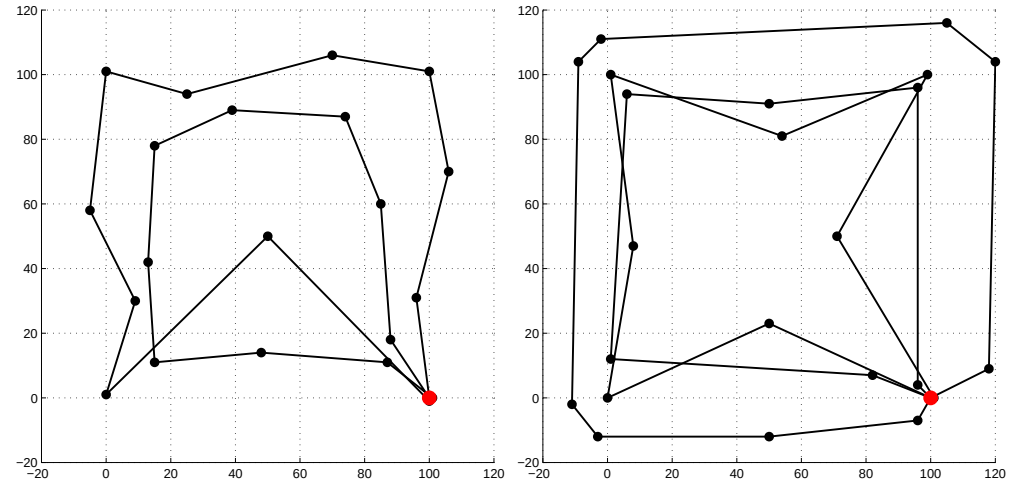
$$\frac{1}{N}\sum_{i=1}^{N} f_i(D_1) < \frac{1}{M}\sum_{i=1}^{M} f_i(D_2)$$

and this relationship is always satisfied when there are enough landmarks. The mean of the differences of the detected angle also monotonically increases with the increasing distance between locations. From this relationship, we define the distance between the image with the following equation,

$$Distance = \frac{1}{N}\sum_{i=1}^{N} \|\theta_i - \theta_t\|$$

and can find both the distance between images with this distance measure and the current location. Before the robot navigates, it gathers images from within its environment. These images are same manner of memory of the place cell with visual information. Using these dataset images, it can estimate its current location by comparing the current images with those in the dataset. It finds all possible distances between current image and dataset images, then finds the smallest one and considers this as its current location. This consideration can cause some error in finding its current location, but this is not increased with moving. This is discussed later in the paper.

We assume that there is a place cell near a current location. In the previous section, we showed that there were many mismatching pairs in the SIFT matching results that either disturb or degrade the overall performance. A larger number are mismatched when there is a distance between the pairs, and mismatching increases the distance measure of a distant object more than a close image. For this reason, the robot is not particularly disturbed by these mismatching pairs when using this distance measure to decide the current location. This works well within a specific range with this distance measure, but this is not reliable when the distance between images is too far. When it finds the current location using this distance measure, the smallest image is merged at a close range, but more are merged at a distant range. This phenomenon is caused by the mismatching pairs. To prevent this false localization, we apply a distance filter. When there is distant activated place cell, we do not consider this one. We estimate the current location using odometer information, and movement information. We are able to find the distance between an estimated current location, and locations where



(a)

(b)

Figure 5.4: (a) Two different exploration routes. Dataset images are gathered on black dots. The beginning and end points are the red dot on the figure. (b) One image gathered in the dataset with occlusions.

we acquire images in the dataset. By estimating the distance information, we can also find the distance score between images. We then normalize this distance score, and combine this information with the distance measure. This distance filter can deny activation from a distant place cell. This distance filter have a trade off to find location of itself. We will discuss about this in later section.

5.1.2 Navigation task

We apply our algorithm to the dynamic environment (Appendix A). Before we perform a navigation task, we explore and gather images from previously set locations in an environment. We set an area that we consider navigating within a square, and manually set the locations to explore (see figure 5.4). We use two different exploration routes. One has 24 images in the dataset over the entire environment. This route cycles an environment twice. It is set to reduce the odometer error during exploration. If we

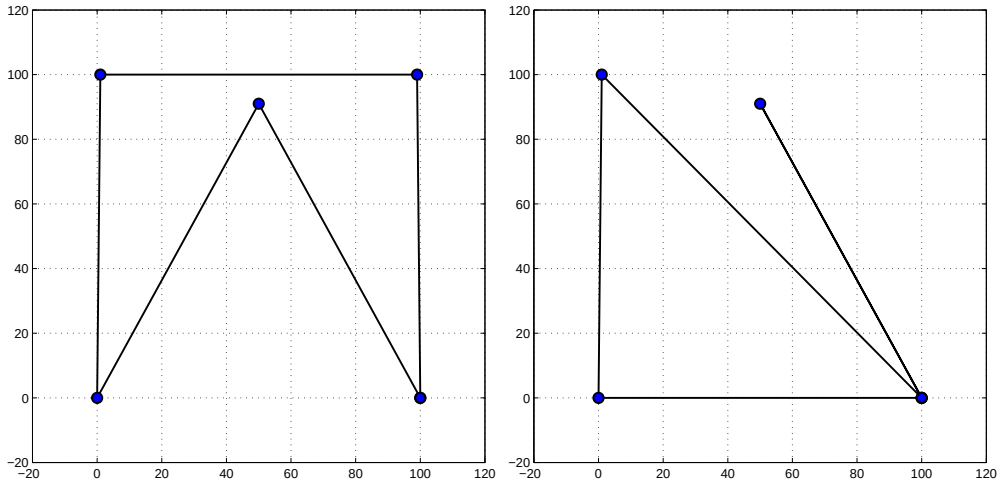


Figure 5.5: Navigation routes. (left) (100, 0)→(54, 81)→(0, 0)→(0, 100)→(100, 100)→(100, 0), (right) (100, 0)→(54, 81)→(100, 0)→(0, 0)→(0, 100)→(100, 0)

use a route that is too long to travel, a severe error in estimating the current location will occur. We separately run two cycles for this route. Another exploration route has more images, and we cover a wider area of the environment. These images are gathered during the exploration phase, and the SIFT descriptors for each image are calculated in the dataset. These exploration routes are set manually, but we also use this algorithm with a random exploration in a later section. In the exploration phase there are many passengers in the scene (figure 5.4 (b) shows a case that includes occlusion). With these images in the dataset, we also remember where these images were taken using given coordinates. With this information, it is able to find the location itself by the distance measure. In the above figure, the range from 0 to 100 is our area of interest for navigation, and this in real coordinates is 3.5m. We thus conduct a navigation task within this range. We obtained many dataset images during the exploration phase, and we use them to conduct a simple navigation task that follows previously set locations. It also starts at location (100, 0). From this starting point, a mobile robot moves to a location at (54, 81), which is one of the locations in the second exploration route. After moving to the location, it moves along the previously set locations in figure 5.5. We also have other types of navigation routes, in addition to these two routes. In this simulation, we move a robot along the previously explored locations to show that it can find its current location by comparing images. If it is not at a previously explored location, it decides its current location is the nearest explored location, which causes errors. This assumption does not always work. The robot can be anywhere in a given environment, and localization errors can often occur. These errors exist most

of time, but are at the same level and are not increased by moving. This implies that the odometer error can be compensated for by this localization algorithm; the error will be further discussed in a later section. After it moves to first location, the robot gathers a current omnidirectional image. It then executes the SIFT matching algorithm using both this current image and images in the dataset. With these matching results, it calculates the distance measure, as mentioned in a previous section. With these similarity measure, or distance measures, it decides its current location, and based on this it decides which way to move and the distance to the next location. It repeats this until it moves one cycle. We conducted this experiment in all the environments in the environment section. Environment 3 has a number of passengers in it, and there are many bystanders. We use this environment to show an example of matching in figure 5.6 where the result of SIFT matching can be seen. The upper matching result is the SIFT matching between the current image and the image from its nearest location. A number of occlusions can be seen on the image, and these are caused by the people within it. However, although occlusions exist, the robot is able to find matching pairs of interest points with high accuracy. In contrast, lower matching results are shown in SIFT matching between the current image and an image from a distant location, where a larger number of mismatches are found and a bigger detected angle difference, or distance measure. Figure 5.6(b) shows the distance measure using an image from (0, 100), which is from the images in the dataset. The smallest one at 11th one is from (1, 100). This result is reasonable. There are other smaller images at the second, fourth, and seventeenth which are from (-2, 111), (-9, 104), and (6, 94), which are points near the current image. This result implies that the algorithm can find its nearest point using this distance measure and that the robot can find its current location well. We simulate navigation route 1 in environment 3. As the robot moves, place cells in figure 5.6 (c) are activated in a sequence. This result shows that the robot can find its current location using this distance measure. We used this result for simulations in other environments, using other exploration and navigation routes. The results are sensitive to an illumination change, and contain a portion of occlusion, but the robot can still find its current location with high accuracy.

5.1.3 Localization error

With this algorithm, the robot estimates its location using its nearest previously explored location. This way of estimation causes some localization errors when it is not

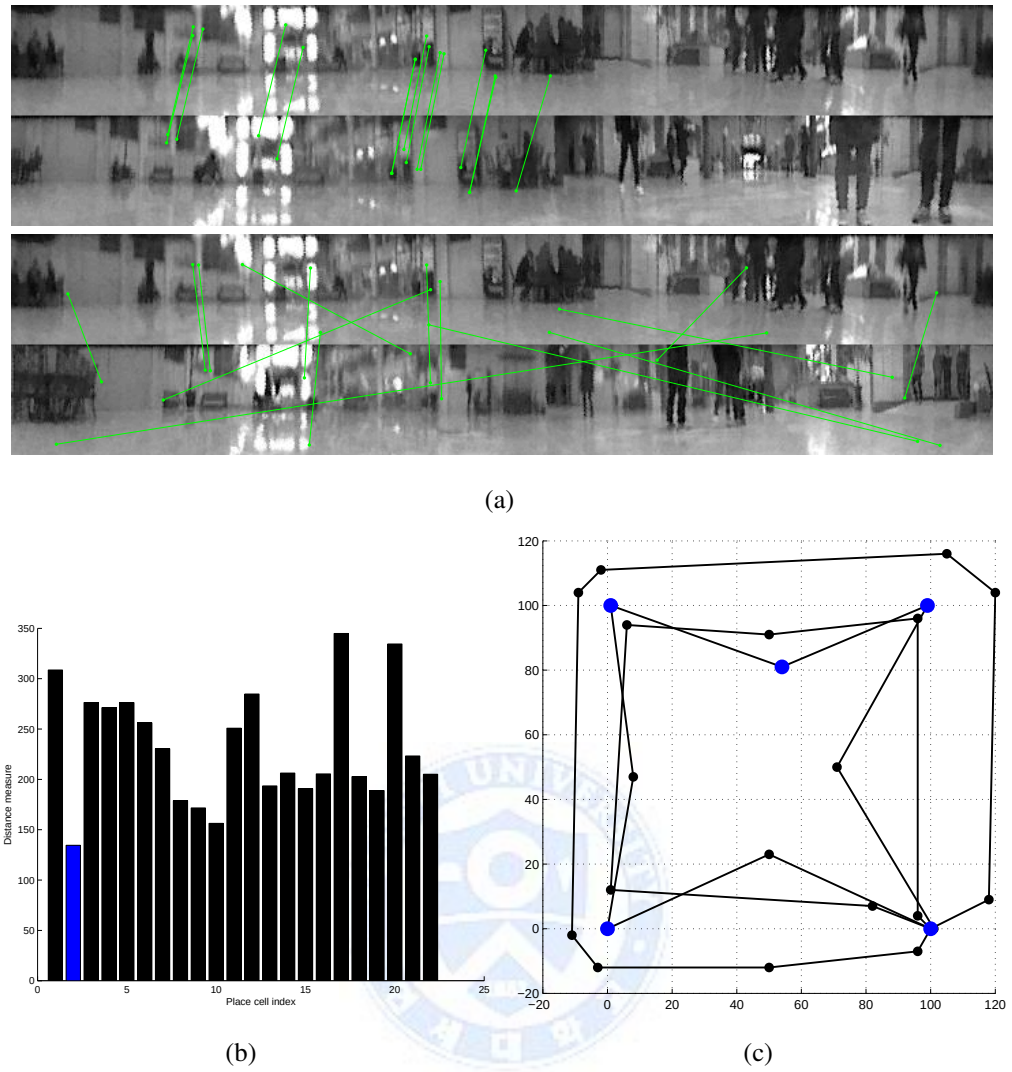


Figure 5.6: (a) (upper) SIFT matching result with close occluded images; (lower) SIFT matching result with distant occluded images. (b) Distance measure from gathered images in the dataset with occlusions. (c) Sequencial goals.

exactly at a previously explored location (the difference between its current location and its nearest previously explored location is this localization error). We simulated this situation and found a characteristic of this error in figure 5.7. We placed a robot on (62, 97), which was not a previously explored location (the nearest previously explored location is at (74, 87)). The robot considered this location as its current location. This error in this estimation is related to the distance between these locations. The robot tries to move to (100, 0), but it moves to (87, 11) because of the estimating error, or localization error. This implies that this error will then propagate to its next move. However, (87, 11) is a previously explored location on, and after it moves there it is

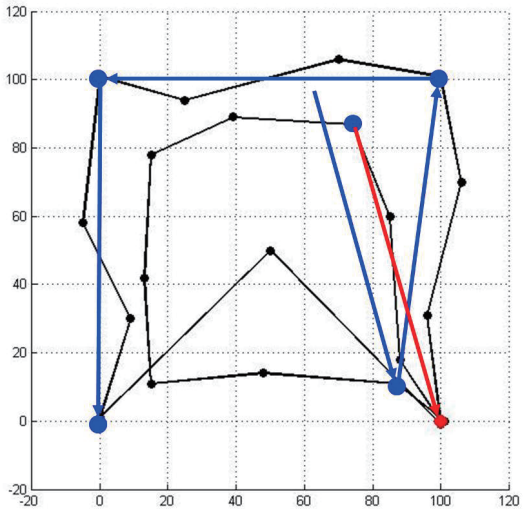


Figure 5.7: A successful case when a mobile robot is not on previously explored location.

able to find its current location correctly. After that, it follows set locations without error. We are certain that this error does not depend on the number of step the robot moves, and the error does not increase with the robots movement. Localization using the odometer creates a problem that increases with movement. This memory-based localization has the advantage of the robot being able to find its current location. The localization error occurs randomly, and we expect that it is highly related to the density of place cells within an environment. When there are many images in the dataset, or place cells in a given environment, the localization error is expected to be smaller than in other cases.

5.1.4 Randomly explored environment

In previous section, we manually set the place cells in the environment. However, this manual process may have limitations, and may cause an error when estimating the current location. We consider that we can gather these images from a place cell automatically, and adaptively. This gathering process depends on the environment, and the robot needs to decide whether to gather images from its current location during the exploration phase. This situation is simulated; to gather images in an environment it is necessary to find the current location by comparing images. It also cope overall environment with random movement. We firstly simulate random movement; we ran-

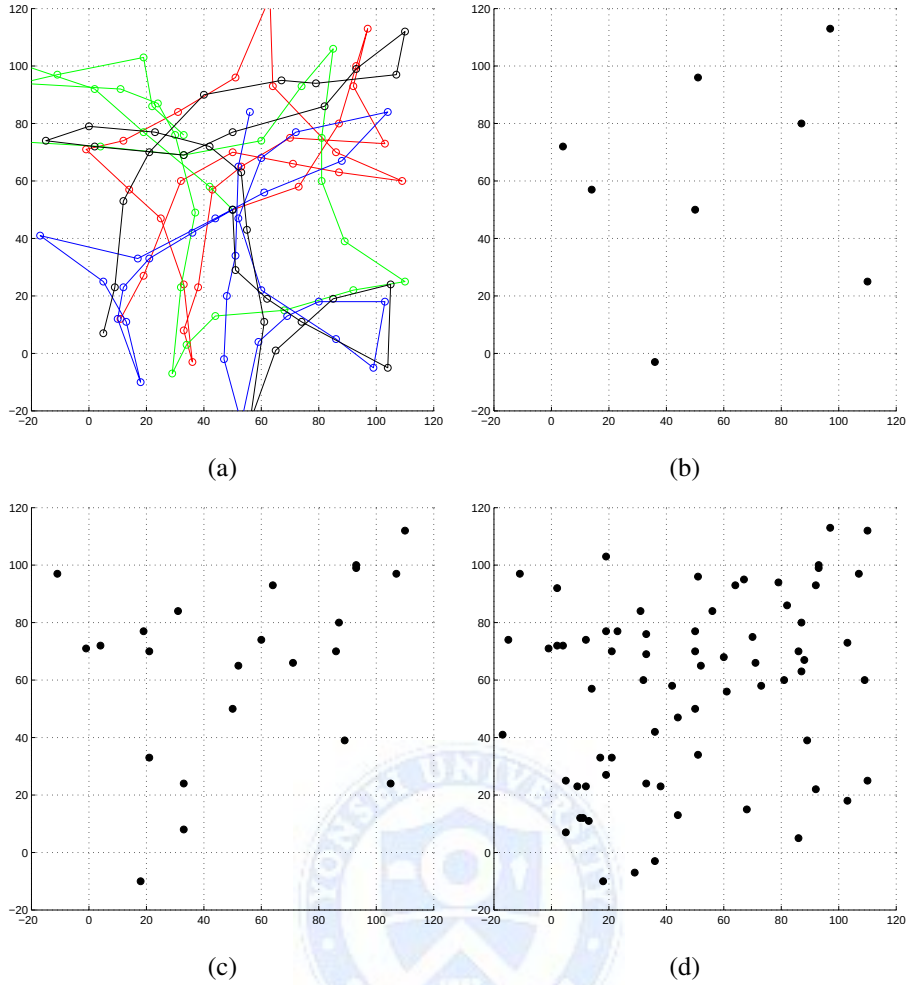
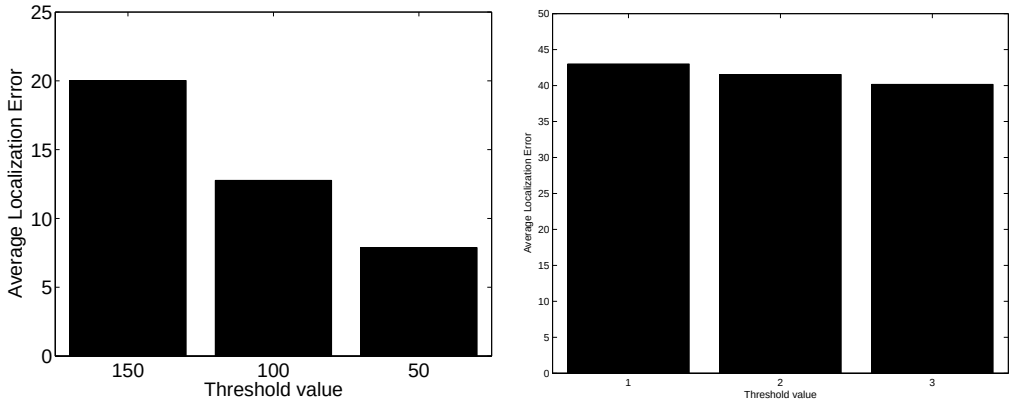


Figure 5.8: (a) Randomly moving trajectories. Generated place cells with different threshold values: (b) 150, (c) 100, (d) 50

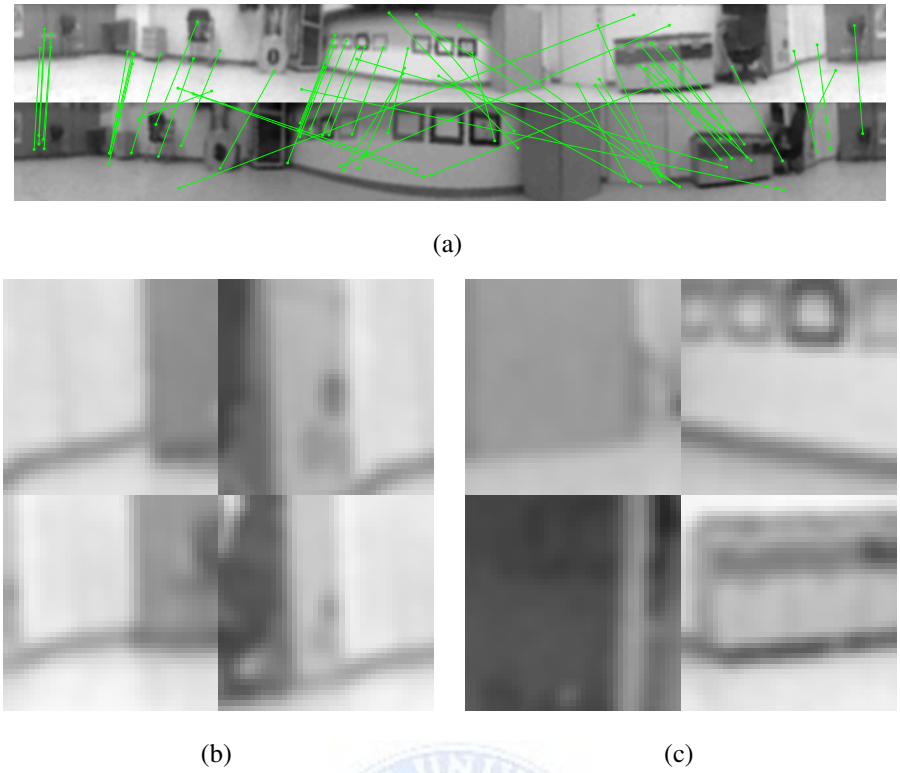
domly control the robot and record its trajectory. Upper random movement decides its moving direction completely randomly. The upper two random movements have different moving distance ranges, but these results do not cover the entire environment. We then reduce the random factor and the tendency to move forward, but if it is over the area of interest, it changes its moving direction drastically. These results with different moving range. We use this movement to gather images in a given environment. We repeat this random movement four times with different starting orientations (these trajectories are shown in figure 5.8 (a) and are sequential the order is red, green, blue, and black). The robot randomly moves and images are obtained at every step; an order is then obtained. Using this information, we simulate generating place cells with random movement, and with these random movements the robot decides whether to store a current image at each step by comparing the current image to those previously



(a) (b)

Figure 5.9: Performance comparison of different threshold values: (a) with distance filter; (b) without distance filter.

explored or stored in the dataset. If a current image is not different from the images in dataset, no current image is taken. When it obtains enough images that are different, it stores the images in the dataset. When using this procedure, deciding whether a current image has enough difference from the images in dataset requires a threshold value. We vary this threshold value from 150, 100, to 50, and gather images using these threshold values (results are shown in figure 5.8 (b), (c), (d)). If the threshold value is high, sparse place cells are generated, but if we use a lower threshold value dense place cells occur. We evaluate the distribution for each generated place cell with a different threshold value. New images in a given environment are gathered, and their location is shown in figure 5.9 (c). We gather 30 images for random locations in the environment, and evaluate these images. We thus have a dataset with different threshold values and we evaluate the distributions with how accurately the current location is found. The current location is found in each of 30 testing images. For localization errors, we calculate the average localization error for 30 testing images, and figure 5.9 (a), (b) shows these results, where 5.9 (a) shows results using a distance filter to find the distance measure, and (b) shows results without this distance filter. We thus conclude that it is necessary for the robot to use this distance filter to find the current location with a higher accuracy. We also conclude that when the place cells have a denser distribution the performance is better.



(a) (b) (c)

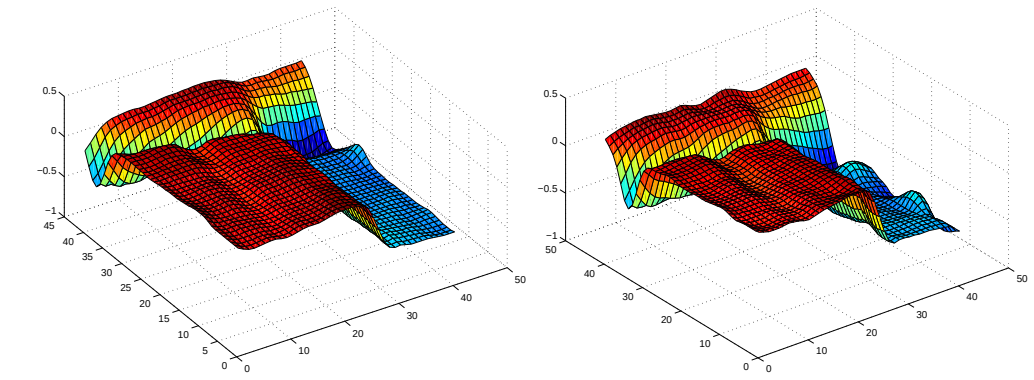
Figure 5.10: The SIFT matching result. (a) Matching features in panoramic image. (b) Window images around correct matching features. (c) Window images around incorrect matching features.

5.2 SIFT based localization

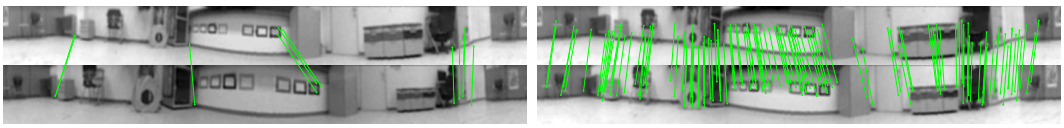
There are many researches to find its current location using the SIFT algorithm (Jeon et al., 2015). It depends on not only images from environment, but also external information to find its current location. Also, it finds the camera position from matching algorithm. This estimation consumes much calculation, and cause inaccurate results. We suggest another localization algorithm using the SIFT matching result with the simple measure with a different approach.

5.2.1 SIFT matching refinement

We apply the SIFT algorithm for panoramic images from omnidirectional camera in figure 5.10 (a). It includes many correct, and incorrect matching pairs. We have to find its correspondence between these points, and remove outliers. We set a window around matching interest points in figure 5.10(b), (c). For correct matching pairs, the images



(a)



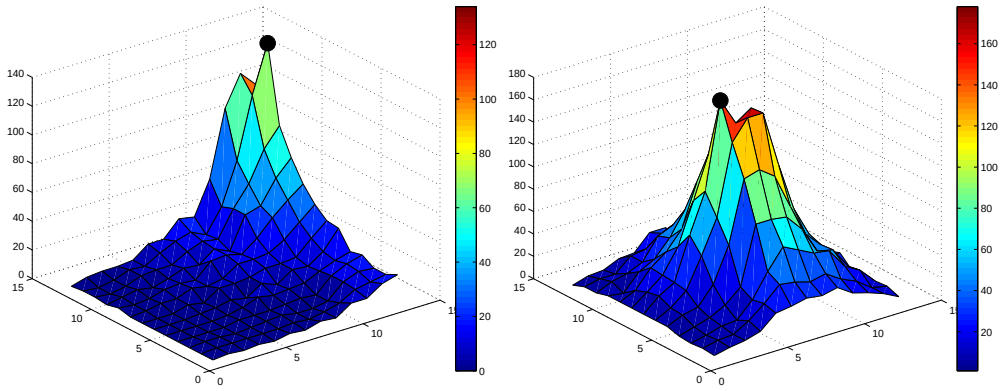
(b)

Figure 5.11: (a) Normalized window for corresponding features. (b) Refined SIFT matching result with different distance from target image.

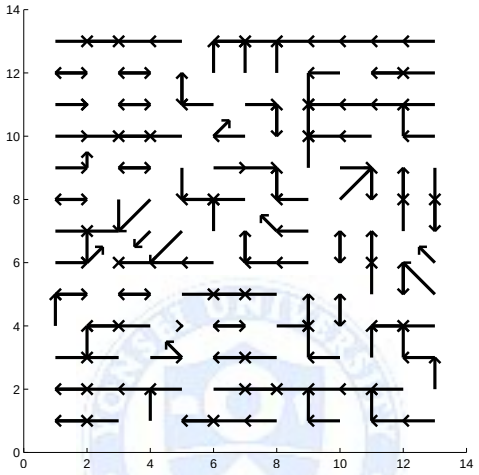
are similar, and its overall shape is similar. In some case, they have some changes by the difference of point of view. For incorrect matching features, they are distinctively different, and we use this characteristic to refine matching of these features. We normalize these this image into a specific range. We compare these normalized data, and decide its correspondence by difference of them. As a result, we can find refined matching results which does not include few outlier. There are just a few or none outlier in giving environment. We assume that result from the SIFT matching is reliable after the refining process.

5.2.2 Localization using the SIFT matching result

These matching results are reliable in a given environment, and we use matching result to localize itself. We assume that we have enough dataset in a given environment, and also have an image from current location. With these data, we compare the current image into the images in data set to find its current location. In this algorithm, we depend on the result from the SIFT matching algorithm. We can find there is difference of results by distance between the locations where the images are taken. When the distance is closer, there are more correct matching features. This then implies we can estimate distance between images using these measures. With one specific target



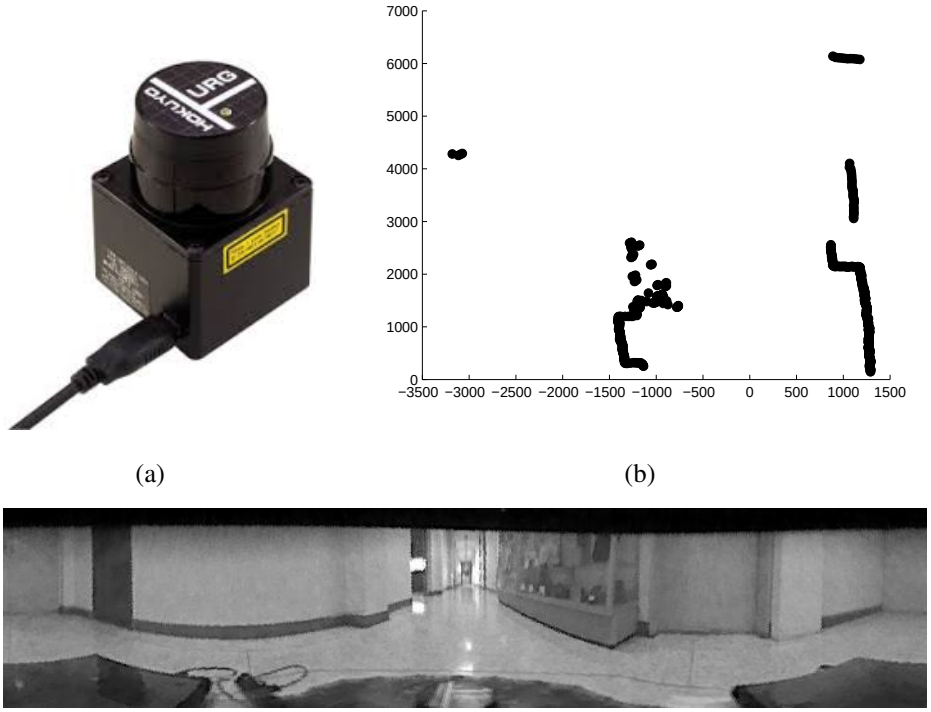
(a)



(b)

Figure 5.12: (a) Distance measure with the SIFT matching features. (b) Localization result.

image, we count the number of observed the SIFT matching features in figure 5.12(a). We set an image from at corner of the environment, and a center of the given environment. We can find it has peaks around the target location. To find the measure, we do not use target image itself, or images from exact the location. In this result, a peak is around the location, and the maximum one is adjacent one. We can find most similar one by comparing this measure from images in dataset. With this, we can find most closest location in dataset using result of the SIFT feature matching. We repeat this with different target locations in overall are of environment in figure 5.12(b). We can find all of estimated location is adjacent location. The average of localization error is 19.57Cm which is depending on distance between adjacent locations in environment. With this result of the SIFT algorithm, and refining algorithm, we can find distance relationship between two images from different locations.



(a) (b)

(c)

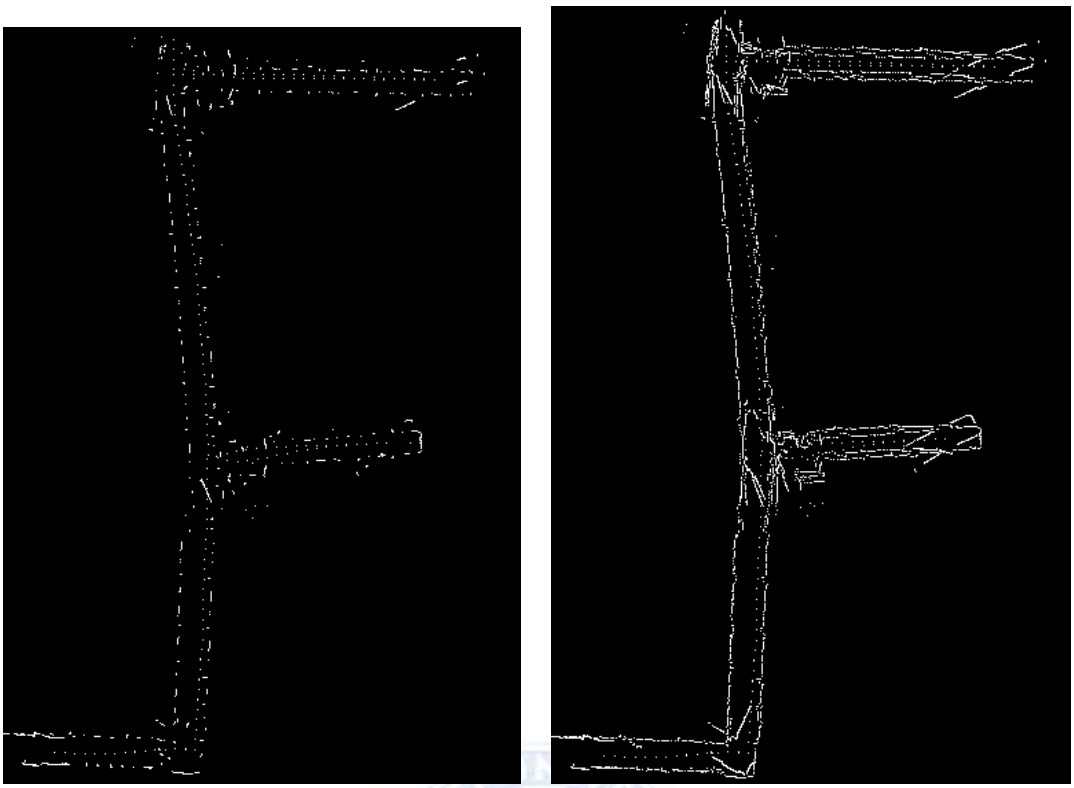
Figure 5.13: (a) Laser sensor(Hokuyo urg-04lx-ug01). (b) Sensor data plot. (c) Experiment environment.

5.3 The SLAM algorithm using a laser sensor

5.3.1 Laser sensor

We conduct another experiment using a mobile robot by applying a simple SLAM algorithm, and implementing this algorithm into the robot. We depend on using a laser sensor to observe the surrounding environment (Hokuyo urg-04lx-ug01) in figure 5.13(a). This sensor observes the distance within the environment in figure 5.10 5.13, and uses a limited detecting range of -120° to 120°. There are 681 data points with an equal angle resolution, and each point dissects the distance of the obstacle from the sensor with a specific angle. The robot is controlled using a mounted laptop, with a forward moving action, and is rotated for specific angle control action. However, this control causes control errors when it moves forward, as the velocity of the two wheels are not exactly the same and this slight difference causes a change in the heading direction. Thus, when it moves forward for a unit distance, the heading angle is changed by 0.3°. This change causes overall distortion of the data by the error in estimating the heading

direction. By controlling the robot manually, it can gather laser sensor data along a corridor on the 4th floor of the 1st engineering building of Yonsei university. Using these laser sensor data, we implement a simple SLAM algorithm. There are many steps involved in implementing a SLAM algorithm with the laser sensor data, and the first is initialization of the map. We observe laser data at the first time, and remember these laser sensor data by map. We set a origin at starting point of a mobile robot. From this point, the location of each observed obstacle is decided. We thus have information about the observed angle, distance from the robot, and the orientation of the robot, and we can thus estimate the locations of obstacles with coordinates. The robot remembers all data points and uses them as map information; after obtaining an initial map it moves to a designated location by the control action. The robot rotates itself using the controlling wheels to look at the designated location and then moves forward to reach the location. This is the method used in the control model. After reaching the location, it detects the range data again and uses the current data and map to find a relationship between them. There are many solutions for this problem, and we chose the most simple one. For each current data point, the robot finds the distance between that particular point and the points on the previous map and chooses the shortest one. If this shortest distance is less than the previously set threshold value, these points are matched. We repeat this matching process for all current data points and find matching points between the current data and map points. Using this information, we then apply the SLAM algorithm. We set a state variable of the Kalman filter as position information of the robot. In general case, it also contains position of map, or data points, but this data is too many to handle with this state variable. For this reason, we use a different map update technique to solve this problem. We set a state variable that includes information about the robots position. We use the control model, and we can find the robots location after moving by path integration. With this information, we have the prediction of a position, or belief in the position. Covariance is also varied using this control action. We then apply this mean and covariance of prediction to the Kalman filter. Innovation is another aspect used in the SLAM algorithm, and this part of the Kalman filter is used to interpret observed data. The robot uses the data information to estimate its current location and its covariance. We then have matching between the current location and map points, and by associating these data, we can find its current location and covariance. We consider these as the mean and covariance of innovation. We gain prediction from the motor command, estimation from the detected data, and apply the Kalman filter to update the robots position. With this, we can localize the



(a) (b)

Figure 5.14: (a) Data from laser sensor. (b) Map using a different method to find the corresponding map.

position of the robot, but we do not update its map because we have another map-updating process. We use the previously matched points and associate these to find the proper location. Many data points are not matched, and we add these new points to the map using the updated position information. After these processes, the position of the robot has been updated, in addition to the map. With this, the robot moves to a new location, and the process is repeated.

We implement a simple SLAM algorithm to a mobile robot and move this robot along a corridor that has two windows of different lengths at the side. We also travel this wings, and this traveling generated location of being repeatedly. Figure 5.14 shows the results of mapping. Figure 5.14(a) shows the gathered range data, and building the map of the environment. This map is reasonable and can express details of the environment. In this algorithm, there are many different parameters that need fitting; one is the threshold value used to find corresponding points and the other is its initial covariance of position. At the first time, it start on origin, but it also has initial covariance. This decide it follows control action, or estimation from distance data. Another important

one is that increment of covariance for every step for prediction. This also decide it depends on its control action, or measurement. In this algorithm, we assume that the covariance remains after movement and it can be shifted, rotated, or warped by the robot's movement and control action. In most cases, different models of motion exist for finding this change of covariance, but we assume that this covariance remains.

5.4 Summary of Chapter 5

We suggest an algorithm to compare with the SIFT features, and its matching procedure. This SIFT algorithm provides the robots corresponding points on images. These features enable it to be very robust to occlusion. This is the reason we use this feature based algorithm to compare these images. With SIFT matching, we suggest the use of distance measures to find similarity between images and with this similarity information the robot can find its current location with high accuracy. We have conducted many experiments to show this algorithm can works in occluded environments, and also suggest that this algorithm can enable the robot to find its current location and can navigate along set locations. This deciding procedure have an important shortage to find current location with similarity measure. Which is it includes most of times. We analysis about this error, and show this error can be compensated by its moving. Thus, this error is not increasing by its movement. This error level is maintained by its moving. This error is decided by the density of the place cells in a given environment. We also simulated random generation of these place cells where we chose random movement with a reason, and gathered images by random moving. We decided whether to use a current image depending on its similarity with a current image and images in the dataset. These results depend on a threshold value which decides whether a current image is similar or not. We determined the effect of this threshold value. It decide overall density of place cells. With different distribution.

Chapter 6

Conclusions

The navigation is one of complex task to find proper way. It have to find its current location which is the localization problem. To define its current location, it have to have its own map. This mapping problem is another issue of the navigation task. Commonly, it is solved by the SLAM algorithm which is based on probabilistic approach, but there is another approach inspired from behaviour of animals and insects. This approach have an advantage that is simple, and it does not requires much calculation. Despite of their simple structure, it guarantee navigation task can be achieved from this behaviour. We are inspired from this navigation algorithms of animals, and we suggest navigation algorithm with different types of landmarks. We suggest a navigation algorithm with the HOG descriptors. This algorithm depends on the HOG descriptors, and use them as landmarks. With these observed landmarks, we show it can find direction to target location with reasonable performance. We also show different types of feature can be a solution for this problem. We apply the SIFT features, the SURF features, and vertical line features to find proper direction to home with these landmarks.

6.1 HOG methods to estimate the homing vector

There is one important assumption for our algorithm which is observed object have its unique gradient characteristic. From this idea, we extract this gradient pattern using the HOG descriptors, and use this algorithm as landmarks to find proper direction to home. We show this algorithm can work with indoor environments with objects. We also show our algorithm can work in environment with many objects. These ob-

ject cause some occlusion, but this algorithm rely on rest of gradient to find proper direction. We also conduct experiments on different types of environment, but overall performance is not very well cause image has bad characteristic of gradient. We show that this algorithm will work with distinctive, and clear gradient patterns around environment to find proper direction. We show this algorithm has robustness against blurring, and interpolation. This robustness of changing of size cause reduction of calculation. We also show this algorithm have robustness against blurring effect on images. This blurring can not affect overall gradient of images, but it is only occur when we apply the blurring filter. When it is not equally blurred by some error from camera, it cause degrade of overall performance. This algorithm also has weak characteristic against addictional noise on images. It depends on gradient pattern, but noise severely defect this gradient characteristic. In this paper, we suggest an algorithm to find direction to home by comparing two snapshot images from the current location, and home location based on the HOG descriptors.

6.2 Feature based bio-inspired navigation

The ALV (Average Landmark Vector) algorithm estimates the direction toward the target location relative to the current location. This requires identifying landmarks, but this has difficulty in the image processing in terms of robustness and efficiency. Thus, a holistic approach like image distance method has been attractive for homing navigation. In this paper, we suggest a robust landmark detection based on edge features. The method shows a quite effective homing navigation in real environment. Many landmark-based methods experience much difficulty in landmark extraction. The suggested method uses vertical edges which are mostly invariant to warped images and the edge feature over neighbor pixels can be good candidates of landmarks. It is not known yet what kind of visual processing or landmark extraction process insects use. Our method may possibly provide a hint of understanding landmark-based navigation for insects.

We also apply computer vision techniques to solve navigation problem. Finding landmark correspondence is one of important problems. We show the SIFT algorithm can be a solution of this problem. With this SIFT algorithm, we can find home direction well. The performance is not quite reliable, but still this algorithm work. In this problem, we need to decide using how many landmarks is optimal. With the SIFT

algorithm, we can conclude using more landmarks is better.
We also apply the SURF algorithm to find out correspondence of landmarks. In contrast, performance with SURF algorithm is better with less landmarks. We calculate angular error by number of landmarks. The result say using just three landmark is optimal number, but there is large error in some locations. We conclude five or around number of landmarks are better.
There is a major factor to degrade its performance, the mismatching. We suggest a way to reduce this effect. The way is applying weighted landmark vector algorithm by its matching score. We assume a mismatching has lower matching score. Based on this assumption, we expect this can reduce error. Result with just five landmarks says there is no improvement, because its performance is enough good. We also apply this algorithm to using many landmarks. This performance is incredibly increased. This result implies we can reduce its disturbance of mismatching by applying weighted average landmark vector algorithm.
There is a important issue of navigation. That is the occlusion problem. Detected object can be disappeared by its moving in real situation. Most of navigation algorithm is weak to this change. We expect that, since this algorithm includes a procedure to classify existence of landmark first by calculating matching score, it will robust to occlusion. We set some environment, and simulate with it. The result is reliable, even we remove or add many objects. We show this algorithm has robustness to occlusion.
There is a concept to estimate distance between two location by comparing two images. The image distance is one of them. We find the sum of matching score by the SURF algorithm has similar characteristic. This sum of score increase by distance smoothly, and there is no exceptional location by similarity of scene. This factor can be used as image distance, and we expect this has more better performance.
We show the feature matching technique can be a solution of finding landmark corresponding problem. Applying this algorithm has many benefit to other local navigation algorithm. The conversion of navigation problem and feature matching technique bear solutions.

6.3 Future works

6.3.1 Landmark matching in complex environment

For most of algorithm, there is one common problem to find proper direction which is distance between locations and observed distance cannot be estimated. The ALV algorithm assume that this distances are almost equal, but it is not satisfied for common environment. The outdoor environment show one of this example. it include some landmarks with close range, and far range. This property cause severely degraded performance. We estimate distance difference by compensating scale difference, but this is not enough to improve overall performance. Estimating distance between landmarks and locations allow to estimate correctly the current location relative to the home location. There is another different assumption which is these environments are static. In real environments, there are many moving people around given environment. This difference also cause degrade of overall performance. In theoretically, using the SIFT, and the SURF feature does not affected by these moving people. These feature matching algorithm can discard these moving people while matching process, but most of case have degrade of overall performance by mismatching features. The HOG method also have degrade of overall performance. It does not have matching process, and depends on overall images to find proper direction. This occlusion, or moving people cause bad performance of this algorithm. There are many additional assumption for the environment. Illumination of environment have to be sustained. Objects should have similar characteristic with different point of view. Floor have to be flatten, and height of camera does not changed. These assumptions have to be solved by algorithm to have robustness characteristic.

6.3.2 Robust feature detection

The feature have to be robust against external changes. For navigation task based on landmarks, finding corresponding landmarks is most important process. If it fails to find corresponding landmarks, its overall performance will falls. For this problem, observing robust landmarks has many limitation, since we rely on visual information only. This visual information cannot provide additional information of observed landmarks. To avoid this problem, there are many different solutions. The random sample consensus (RANSAC) can be a solution for this problem. It assume conditions of ob-

served landmarks, and find proper landmarks with this assumptions. For example, it assume height of camera does not changed, and it find matching landmarks with same height. It generate criteria of observed landmarks, and find proper landmarks. Repeating this guarantee optimal matching results. There is another way to detour this problem which is using additional information. Using only camera has limited performance, but using additional information lower mismatching pairs of landmarks. These methods can improve matching of observed feature, or landmarks. This also cause enhancement of overall performance.

6.3.3 Network based SLAM

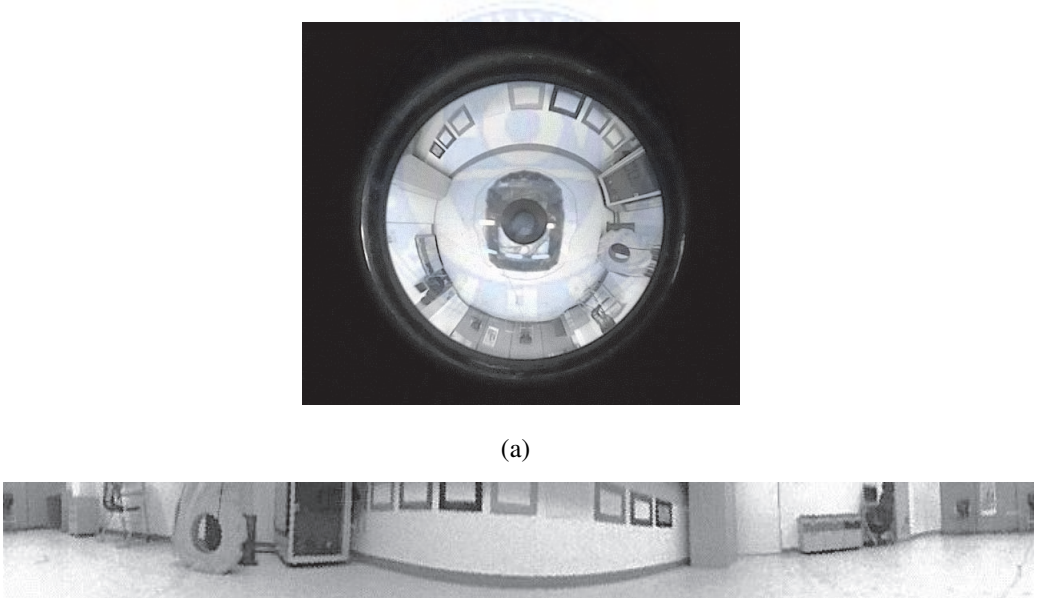
There are many researches on mammal brain, and there is explaination their brain activity (Hafting et al., 2005) in the dorsocaudal medial entorhinal cortex. This cortex is involving to their navigation ability, but there was any understanding of this mechanism. This work sense this cortex with proper mechanism, and explain how it works. It shows the grid cells are highly involves to navigation ability of rodents, and this cells are synchronized, or aligned with detected landmarks with distinctive features. This also explain that this grid cells can be a part of path-integration-based map of the spatial environment. In previous research, there is a heading direction cells to find its orientation. They interact to find its position, and these are one of important role of navigation ability of the rodents.

Form this architecture of navigation part of animals, or rodents, there is an approach to solve the SLAM algorithm (Milford and Wyeth, 2010). The ”RatSLAM” is most representative algorithm from animal behaviour. This algorithm is assume that these animals are depending on their visual information. This algorithm is also assume that we have our internal memory to remember where we have been, and images at the locations. With these simple, and reasonable assumption, the ”RatSLAM” suggest a robust, and persistent navigation, and mapping algorithm. This SLAM algorithm is based on training of networks to mimic activity of their brain. In most of study, it rely on entire images to activate the network. Otherwise, there are researches on animals, and insects. These researches claim that they are rely on distinctive feature to localize itself, or finding target direction. From this idea, we will train the network with extracted feature to find its current location, or finding proper direction to target location.

Appendix A

Environment & Image processing

A.1 Static environment



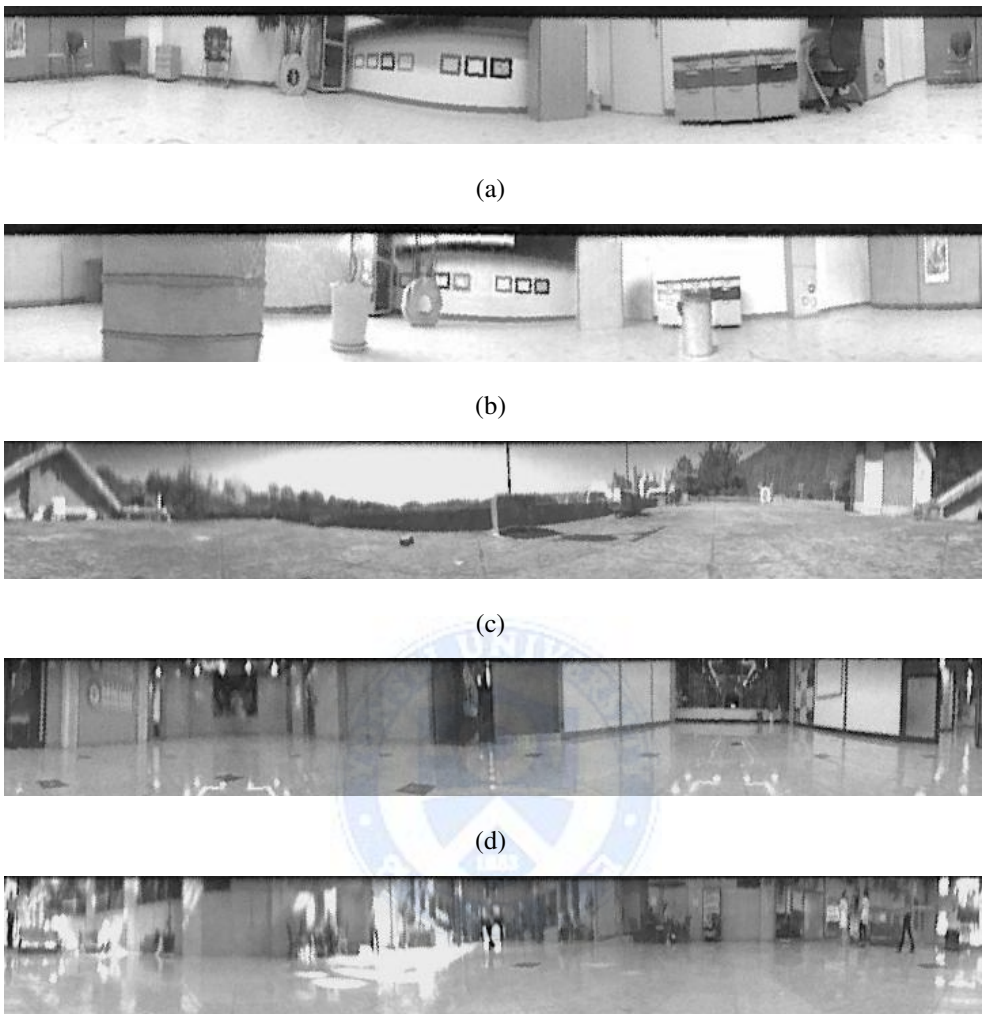
(a)

(b)

Figure A.1: (a) Omni directional image, (b) Converted panoramic image.

We use omni-directional camera, and take images on given environment. An example of snapshot image is in figure A.1(a). It have scene around environment, and describe overall objects. We warp this image into panoramic image in figure A.1(b). This panoramic image allow us to know bearing information of observed objects. We use this image as snapshot image from locations in environments.

We gather image data in various environments in figure A.2. Each environment has its own characteristic.



(a)

(b)

(c)

(d)

(e)

Figure A.2: (a) Indoor environment(13 by 13), (b) Indoor environment with objects(11 by 11), (c) Outdoor environment(11 by 11), (d) Featureless environment(11 by 11), (e) Occluding people(11 by 11).

Figure A.2(a) shows an indoor environment. There are many object around environment. There are some distinctive objects, and also similar objects. These characteristic can show validation of our algorithm. Figure A.2(b) shows an environment with many object in given environment. This cause occlusion of snapshot. These objects cause occlusion, and degrade of overall performance. We also conduct image with outdoor environment in figure A.2(c). This outdoor environment has features at far distance. These features are observed in same bearing of panoramic image, and cause some de-

grade of overall performance. There are another indoor environments which have its own characteristic. One has featureless scene around environment. It is composed of patternless walls, and similar colour. This environment has one advantage that has many vertical lines. Another environment has passengers in environment. This passenger also cause occlusion or defection of panoramic image. We show our algorithm can work in this environment.

In these environments, we set an area which is composed of grid points. We have an image from each grid point. Distance between adjacent points is about 20Cm. A size of grid is 13 by 13, or 11 by 11. We set these environments, and show our algorithm can works in various environments.

A.2 Dynamic environment

We conduct a navigation task in different types of environments. We have two static environments, and two dynamic environment. In environment 1, we place artificial objects which can be concrete, and distinctive landmarks. We place these object outside, and inside interest area. A scene of environment 2 is monotonous. Overall color it can detect is similar, and there are similar objects around this interest area. This environment is almost static. In this place, there are just few passengers during experiment. On the other hand, an environment 3 has a number of passengers around it. This also has monotonous scene, and color compositions. This environment also has one characteristic to consider which is illumination change. Its illumination is from natural sunlight. This illumination condition is varying by its time. Oppositely, environment 4 has artificial light, it maintain same illumination condition. It also has many passengers in this environment. In these environment, we set an interest area to consider. In environment 2, and 3, we conduct our experiments afternoon. When it is late, and it is too dark, the SIFT matching have worse performance than other cases.

(a) (b)

(c) (d)

Figure A.3: (a) Static indoor environment with artificial objects. (b) Static indoor environment. (c), (d) Dynamic environments.

Bibliography

Addis, D. R., Wong, A. T., and Schacter, D. L. (2007). Remembering the past and imagining the future: Common and distinct neural substrates during event construction and elaboration. *Neuropsychologia*, 45(7):1363–1377.

Andel, D. and Wehner, R. (2004). Path integration in desert ants, cataglyphis: how to make a homing ant run away from home. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 271(1547):1485–1489.

Andersen, P., Morris, R., Amaral, D., Bliss, T., and O'Keefe, J. (2006). *The hippocampus book*. Oxford University Press.

Baek, S. and Kim, D. (2014). Snapshot homing navigation based on edge features. In *From Animals to Animats 13*, pages 98–107. Springer.

Baek, S. and Kim, D. (2015a). Feature extraction and network learning based route learning and localization (being prepared).

Baek, S. and Kim, D. (2015b). Histogram of oriented gradient descriptors based navigation and analysis (being prepared).

Baek, S. and Kim, D. (2015c). Local navigation using feature matching technique (being prepared).

Barron, J. L., Fleet, D. J., and Beauchemin, S. (1994). Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77.

Barshan, B. and Durrant-Whyte, H. F. (1995). Inertial navigation systems for mobile robots. *Robotics and Automation, IEEE Transactions on*, 11(3):328–342.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359.

Betke, M. and Gurvits, L. (1997). Mobile robot localization using landmarks. *Robotics and Automation, IEEE Transactions on*, 13(2):251–263.

Bichsel, M. and Pentland, A. P. (1994). Human face recognition and the face image set's topology. *CVGIP Image Understanding*, 59:254–254.

Binding, D. M. and Labrosse, F. (2006). Visual local navigation using warped panoramic images.

Bird, C. M. and Burgess, N. (2008). The hippocampus and memory: insights from spatial processing. *Nature Reviews Neuroscience*, 9(3):182–194.

Booij, O., Zivkovic, Z., and Kröse, B. (2008). Sampling in image space for vision based slam.

Booij, O., Zivkovic, Z., and Kröse, B. (2009). Efficient data association for view based slam using connected dominating sets. *Robotics and Autonomous Systems*, 57(12):1225–1234.

Brady, M. and Wang, H. (1992). Vision for mobile robots. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 337(1281):341–350.

Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. pages 25–36.

Buckley, M. and Gaffan, D. (1997). Impairment of visual object-discrimination learning after perirhinal cortex ablation. *Behavioral neuroscience*, 111(3):467.

Burke, A. and Vardy, A. (2006). Visual compass methods for robot navigation.

Cartwright, B. and Collett, T. S. (1983). Landmark learning in bees. *Journal of Comparative Physiology*, 151(4):521–543.

Cha, Y. and Kim, D. (2012). Omni-directional image matching for homing navigation based on optical flow algorithm. pages 1446–1451.

Chahl, J. and Srinivasan, M. (1996). Visual computation of egomotion using an image interpolation technique. *Biological Cybernetics*, 74(5):405–411.

Chen, H.-m. (2004). Mutual information: a similarity measure for intensity based image registration. pages 89–108.

Collett, M., Collett, T. S., and Srinivasan, M. V. (2006). Insect navigation: measuring travel distance across ground and through air. *Current biology*, 16(20):R887.

Collett, T. (1992). Landmark learning and guidance in insects. *Philosophical Transactions: Biological Sciences*, pages 295–303.

Collett, T. (1996). Insect navigation en route to the goal: multiple strategies for the use of landmarks. *Journal of Experimental Biology*, 199(1):227–235.

Collett, T. and Land, M. (1975). Visual spatial memory in a hoverfly. *Journal of Comparative Physiology*, 100(1):59–84.

Collett, T. S. (2008). Insect navigation: visual panoramas and the sky compass. *Current Biology*, 18(22):R1058–R1061.

Collett, T. S. and Baron, J. (1994). Biological compasses and the coordinate frame of landmark memories in honeybees. *Nature*, 368(6467):137–140.

Cozman, F. and Krotkov, E. (1995). Robot localization using a computer vision sextant. 1:106–111.

Cozman, F., Krotkov, E., and Guestrin, C. (2000). Outdoor visual position estimation for planetary rovers. *Autonomous Robots*, 9(2):135–150.

Cressant, A., Muller, R. U., and Poucet, B. (1997). Failure of centrally placed objects to control the firing fields of hippocampal place cells. *The Journal of Neuroscience*, 17(7):2531–2542.

Epstein, R. and Kanwisher, N. (1998). A cortical representation of the local visual environment. *Nature*, 392(6676):598–601.

Etemad, K. and Chellappa, R. (1997). Discriminant analysis for recognition of human face images. *JOSA A*, 14(8):1724–1733.

Fennema, C. L. and Thompson, W. B. (1979). Velocity determination in scenes containing several moving objects. *Computer graphics and image processing*, 9(4):301–315.

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

Franz, M. O. and Krapp, H. G. (2000). Wide-field, motion-sensitive neurons and matched filters for optic flow fields. *Biological cybernetics*, 83(3):185–197.

Franz, M. O., Schölkopf, B., and Bülthoff, H. H. (1997). Homing by parameterized scene matching.

Franz, M. O., Schölkopf, B., Mallot, H. A., and Bülthoff, H. H. (1998). Where did i take that snapshot? scene-based homing by image matching. *Biological Cybernetics*, 79(3):191–202.

Frier, H., Edwards, E., Smith, C., Neale, S., and Collett, T. (1996). Magnetic compass cues and visual pattern learning in honeybees. *Journal of Experimental Biology*, 199(6):1353–1361.

Gaspar, J., Winters, N., and Santos-Victor, J. (2000). Vision-based navigation and environmental representations with an omnidirectional camera. *Robotics and Automation, IEEE Transactions on*, 16(6):890–898.

Giachetti, A. (2000). Matching techniques to compute image motion. *Image and Vision Computing*, 18(3):247–260.

Gluckman, J. and Nayar, S. K. (1998). Ego-motion and omnidirectional cameras. pages 999–1005.

Goedemé, T., Nuttin, M., Tuytelaars, T., and Van Gool, L. (2004). Vision based intelligent wheel chair control: the role of vision and inertial sensing in topological navigation. *Journal of Robotic Systems*, 21(2):85–94.

Goedemé, T., Tuytelaars, T., Van Gool, L., Vanhooydonck, D., Demeester, E., and Nuttin, M. (2005). Is structure needed for omnidirectional visual homing? pages 303–308.

Goldman, A. and Johansson, J. K. (1978). Determinants for search of lower prices: an empirical assessment of the economics of information theory. *Journal of Consumer Research*, pages 176–186.

Gonzalez-Barbosa, J.-J. and Lacroix, S. (2002). Rover localization in natural environments by indexing panoramic images. 2:1365–1370.

Gourichon, S., Meyer, J., and Pirim, P. (2002). Using coloured snapshots for short-range guidance in mobile robots. *International Journal of Robotics and Automation*, 17(4):154–162.

Gourichon, S., Meyer, J.-A., Ieng, S., Smadja, L., and Benosman, R. (2003). Estimat-

ing ego-motion using a panoramic sensor: Comparison between a bio-inspired and a camera-calibrated method. pages 91–101.

Graham, P. and Collett, T. S. (2006). Bi-directional route learning in wood ants. *Journal of experimental Biology*, 209(18):3677–3684.

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., and Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK.

Harris, R. A., Graham, P., and Collett, T. S. (2007). Visual cues for the retrieval of landmark memories by navigating wood ants. *Current biology*, 17(2):93.

Hatzitheodorou, M., Karabassi, E., Papaioannou, G., Boehm, A., and Theoharis, T. (2000). Stereo matching using optic flow. *Real-Time Imaging*, 6(4):251–266.

Hong, J., Tan, X., Pinette, B., Weiss, R., and Riseman, E. M. (1992). Image-based homing. *Control Systems, IEEE*, 12(1):38–45.

Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, 17(1):185–203.

Jain, J. and Jain, A. (1981). Displacement measurement and its application in interframe image coding. *Communications, IEEE Transactions on*, 29(12):1799–1808.

Jeon, D., Kim, S., Lee, S., and Jeon, J.-i. (2015). Architecture of image feature db storage for mobile visual localization. In *Advanced Communication Technology (ICACT), 2015 17th International Conference on*, pages 88–91. IEEE.

Jogan, M. and Leonardis, A. (1999). Panoramic eigenimages for spatial localisation. pages 558–567.

Kohler, M. and Wehner, R. (2005). Idiosyncratic route-based memories in desert ants, melophorus bagoti: how do they interact with path-integration vectors? *Neurobiology of learning and memory*, 83(1):1–12.

Krapp, H. G., Hengstenberg, R., et al. (1996). Estimation of self-motion by optic flow processing in single visual interneurons. *Nature*, 384(6608):463–466.

Kuipers, B. J. and Byun, Y.-T. (1988). A robust, qualitative method for robot spatial learning. 88:774–779.

Labrosse, F. (2004). Visual compass. *Proceedings of Towards Autonomous Robotic Systems, University of Essex, Colchester, UK*, pages 85–92.

Labrosse, F. (2007). Short and long-range visual navigation using warped panoramic images. *Robotics and Autonomous Systems*, 55(9):675–684.

Lambrinos, D. (1998). *Navigation in biorobotic agents*. PhD thesis.

Lambrinos, D., Kobayashi, H., Pfeifer, R., Maris, M., Labhart, T., and Wehner, R. (1997). An autonomous agent navigating with a polarized light compass. *Adaptive behavior*, 6(1):131–161.

Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., and Wehner, R. (2000). A mobile robot employing insect strategies for navigation. *Robotics and Autonomous systems*, 30(1):39–64.

Lehrer, M. and Bianco, G. (2000). The turn-back-and-look behaviour: bee versus robot. *Biological cybernetics*, 83(3):211–229.

Liu, M., Pradalier, C., Chen, Q., and Siegwart, R. (2010). A bearing-only 2d/3d-homing method under a visual servoing framework. pages 4062–4067.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision.

Makadia, A. and Daniilidis, K. (2006). Rotation recovery from spherical images without correspondences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1170–1175.

Mallot, H. A., Arndt, P. A., and Bülthoff, H. H. (1996). A psychophysical and computational analysis of intensity–based stereo. *Biological cybernetics*, 75(3):187–198.

Milford, M. and Wyeth, G. (2010). Persistent navigation and mapping using a biologically inspired slam system. *The International Journal of Robotics Research*, 29(9):1131–1153.

Möller, R. (2000). Insect visual homing strategies in a robot with analog processing. *Biological Cybernetics*, 83(3):231–243.

Möller, R., Krzykawski, M., and Gerstmayr, L. (2010). Three 2d-warping schemes for visual robot navigation. *Autonomous Robots*, 29(3-4):253–291.

Möller, R., Lambrinos, D., Pfeifer, R., Labhart, T., and Wehner, R. (1998). Modeling ant navigation with an autonomous agent. *From animals to animats*, 5:185–194.

Möller, R., Maris, M., and Lambrinos, D. (1999). A neural model of landmark navigation in insects. *Neurocomputing*, 26:801–808.

Müller, M. and Wehner, R. (1988). Path integration in desert ants, cataglyphis fortis. *Proceedings of the National Academy of Sciences*, 85(14):5287–5290.

Nagel, H.-H. (1982). On change detection and displacement vector estimation in image sequences. *Pattern Recognition Letters*, 1(1):55–59.

Nakazawa, K., McHugh, T. J., Wilson, M. A., and Tonegawa, S. (2004). Nmda receptors, place cells and hippocampal spatial memory. *Nature Reviews Neuroscience*, 5(5):361–372.

Nelson, R. C. and Aloimonos, J. (1988). Finding motion parameters from spherical motion fields (or the advantages of having eyes in the back of your head). *Biological Cybernetics*, 58(4):261–273.

Nistér, D., Naroditsky, O., and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20.

O'Keefe, J. and Dostrovsky, J. (1971). The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 34(1):171–175.

Papenberg, N., Bruhn, A., Brox, T., Didas, S., and Weickert, J. (2006). Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67(2):141–158.

Röfer, T. (1995). Controlling a robot with image-based homing. *Kognitive Robotik, ZKW-Bericht*, 3:95.

Se, S., Lowe, D., and Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international Journal of robotics Research*, 21(8):735–758.

Shi, J. and Tomasi, C. (1994). Good features to track. pages 593–600.

Srinivasan, M. V., Chahl, J. S., and Zhang, S.-W. (1997). Robot navigation by visual dead-reckoning: inspiration from insects. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(01):35–47.

Vardy, A. and Moller, R. (2005). Biologically plausible visual homing methods based on optical flow techniques. *Connection Science*, 17(1-2):47–89.

Vardy, A. and Oppacher, F. (2003). Low-level visual homing. In *Advances in Artificial Life*, pages 875–884. Springer.

Vardy, A. and Oppacher, F. (2004). Anatomy and physiology of an artificial vision matrix. In *Biologically Inspired Approaches to Advanced Information Technology*, pages 290–305. Springer.

Vasudevan, S., Gächter, S., Nguyen, V., and Siegwart, R. (2007). Cognitive maps for mobile robotsan object based approach. *Robotics and Autonomous Systems*, 55(5):359–371.

Weber, K., Venkatesh, S., and Srinivasan, M. (1999). Insect-inspired robotic homing. *Adaptive Behavior*, 7(1):65–97.

Wehner, R. (1987). Spatial organization of foraging behavior in individually searching desert ants, cataglyphis (sahara desert) and ocymyrmex (namib desert). *Experientia. Supplementum*, (54):15–42.

Wehner, R. (1997). The ants celestial compass system: spectral and polarization channels. In *Orientation and communication in arthropods*, pages 145–185. Springer.

Wehner, R., Michel, B., and Antonsen, P. (1996). Visual navigation in insects: coupling of egocentric and geocentric information. *Journal of Experimental Biology*, 199(1):129–140.

Wehner, R. and Müller, M. (1985). Does interocular transfer occur in visual navigation by ants?

Wehner, R. and Räber, F. (1979). Visual spatial memory in desert ants, cataglyphis bicolor (hymenoptera: Formicidae). *Experientia*, 35(12):1569–1571.

Wehner, R. and Wehner, S. (1986). Path integration in desert ants. approaching a long-standing puzzle in insect navigation. *Monitore zoologico italiano*, 20(3):309–331.

Wehner, R. and Wehner, S. (1990). Insect navigation: use of maps or ariadne's thread? *Ethology Ecology & Evolution*, 2(1):27–48.

Winters, N. and Santos-Victor, J. (1999). Omni-directional visual navigation. pages 109–118.

Wittmann, T. (1995). Modeling landmark navigation. *Report*, 3:95.

Yagi, Y., Nishizawa, Y., and Yachida, M. (1995). Map-based navigation for a mobile robot with omnidirectional image sensor copis. *Robotics and Automation, IEEE Transactions on*, 11(5):634–648.

Yagi, Y. and Yachida, M. (1991). Real-time generation of environmental map and obstacle avoidance using omnidirectional image sensor with conic mirror. pages 160–165.