iNavigation: An Image Based Indoor Navigation System

Linfeng (Eric) Wang

A thesis submitted to
Auckland University of Technology
in partial fulfilment of the requirements for the degree
of
Master of Computer and Information Sciences (MCIS)

2012

School of Computing and Mathematical Sciences

Contents

List of Figures

List of Tables

Attestation of Authorship

Acknowledgements

I would like to thank my parents for their love and support. The entire school of the AUT university has been extraordinary helpful in bringing this work to completion. I would like to thank in particular all teachers, supervisors and administration staff. Dr WeiQi Yan had been everything one could hope for in a supervisor and more. I would also like to thank Ms Shoba Tegginmath and Ms Ann Wu for their insight and dedicated support. Finally I would like to acknowledge Rob Hess and Dahua Lin for sharing their code.

Linfeng (Eric) Wang

Auckland

September 2012

Abstract

Image-based navigation techniques attract much attention as a powerful alternative to traditional map-based navigation. The Google Street View, which gets a mixed reception is one such technique featured in Google Map that allows users to navigate through large scale outdoor environment with 360 degree imagery. However, the Google Street View can not provide timely updates because it requires immense data, and this technique involving a panoramic camera has not been extended to indoor environment due to its data collection approach. This thesis proposes a convenient and efficient image-based indoor navigation web application designed for mobile phones. It utilizes ordinary photographs and navigation information, rather than the panoramic view used in Google Street View, to guide people through the building. Our system implements SIFT feature detection and ANN search to provide timely positioning service. It also can roughly locate a query image on the map by using IPM (Inverse Perspective Mapping), if the query image is taken following certain rules. Thus it enables interactive navigation and knowledge sharing among users.

Chapter 1

Introduction

In this thesis, we propose a novel image based indoor navigation system for mobile phone, named iNavigation. We begin by discussing the necessity for indoor navigation systems. After giving a brief overview of indoor and outdoor navigation, we formulate the research question which provides the overall focus of the research. The methodologies that we follow are also presented. Finally, we present the overall structure of this thesis.

1.1 Motivation

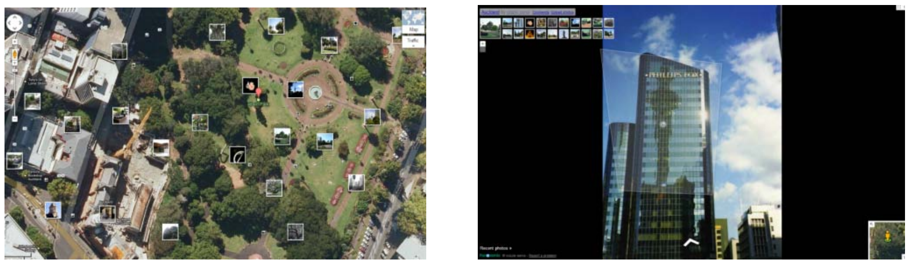
We sometimes get lost in a complex building when we are looking for a room in a hurry [1]. In such situations, we have to count on signs or a “You are here” map in the building. However, research conducted in [2, 3, 4] showed that such signs and maps cannot significantly shorten the time to find rooms. If however our mobile phone could tell us where we are, and even orient us through the building to the destination in real time, we surely could arrive at the destination on time. It just makes life easier and that is the value of indoor navigation.

Outdoor navigation service has been growing very fast and now has becomes an important part of people’s daily life. People use it to acquire directions for driving, travel, routing, etc. However, indoor navigation for pedestrians has only just begun to attract research interest over the last few years, mainly due to the unavailability of GPS signals in indoor environment [5]. In the outdoor environment, as long as the GPS navigation system is used where there is adequate satellite coverage and strong signal strength for GPS positioning, the user’s location can be determined accurately. However, once the user is in a building, i.e. an area without enough satellite coverage, the positioning becomes either inaccessible, or even though positioning can be achieved with the aid of cellular network, it is not sufficiently accurate for navigation purposes. Another issue is that research finds traditional 2D map may not be the best way for navigation because of lack of reality. Satalich’s research [2] shows that reading maps does not provide sufficient visual information to users. Additional semantic as well as visual information such as annotations, landmarks, routes could help users understand the map explicitly and quickly.

Several dedicated indoor navigation solutions addressing these issues have been developed in this industry. However, their drawbacks are obvious and include high cost, low accuracy and difficulties in deployment. Accurate and robust indoor navigation with low cost is still an open problem. Relevant research is ongoing, focusing on either developing new navigation techniques or finding a better combination of current solutions.

It is undeniable that mobile telecommunication technology is now a vital part of modern culture and civilization. The capabilities and performance of current mobile phones and smart phones have transformed them into portable communication, entertainment, and multimedia information and navigation devices with very strong computation ability. On the other hand, widely deployed wireless networks and channels make it possible to transmit multimedia content, i.e. audios, photos and videos to and from mobile devices in time. As a result, nowadays we can use our mobile phones to take photos, record videos, download/upload and view multimedia information via Wi-Fi, 3G/4G or GPRS network almost anywhere and anytime. Hence the enormous information available on the internet becomes a “local” context for these mobile phone users [6]. Because of this overwhelming mobility trend, location-based services start aiming at delivering the local relevant information to mobile and smart phones rather than specifically designed navigation devices. It is now a growing part of the so-called web-service industry.

It just comes very naturally that image based techniques [8] attract navigation industry's interest since the mobility trend and wireless network started booming. An image has rich visual information that can help us recognize location; it can also be transmitted via wireless networks easily. Besides, the majority of mobile phones can be used to take photos by ordinary users. This feature enables photo-based knowledge sharing for navigation purposes. Although image based navigation looks promising, few commercial products based on this technique can be seen on the market. It seems to be less popular than we expected. One of the successes, Google Maps with Street View is such a navigation technique that visualizes the outdoor environment with 360-degree street level imagery. It acquires GPS information from Google Maps and renders a virtual street environment by stitching ordinary photos together. This technique requires a panorama camera installed on top of a moving car that records images and geo-data simultaneously. Therefore, it is quite expensive (money is not a real problem for IT giants like Google) and not feasible to be implemented in an area where the location is inaccessible for cars and tricycles, such as mountains, parks as well as the insides of buildings. In order to solve this problem, Google provides an alternative service called Google Image Gallery shown in Figure 1.1. It retrieves images from a geo-location-oriented photo sharing website named "Panoramio" (purchased by Google several years ago) and pins them on Google Maps. If correspondences can be found between images, they will be linked together and the directions from one image to other linked images will be given as well.



(a) (b)

Figure. 1.1. An example of Google Image Gallery. Fig. 1(a) shows the image gallery for a park, Fig. 1 (b) indicates an image in the gallery was linked to other images.

However, this service doesn’t provide sufficient information for navigation. We do not know where we are and where we are heading to. Other than the data collection issue, Google Maps with Street View could not provide timely updates because of its immense data that is not easily obtained.

1.2 Background

It is easy to understand and justify why outdoor navigation systems are quite desirable. Indoor navigation systems have attracted researchers’ interest over last few years but demand for indoor navigation is not convincing, so further justification is required. Navigation tasks performed within buildings don not meet the kind of challenges presented by outdoor navigation. For instance, mounted GPS navigators used by drivers are required to provide route information ranging from a few kilometers to even hundreds of kilometers, while the indoor navigators perform within a working space confined to the physical extent of the structure of the building [7]. Usually, in an indoor navigation usage scenario, the maximum distance from origin to destination is no longer than one kilometer (across floors). Other outdoor navigation challenges consist of sluggish map update and unexpected incidents that could affect road condition. The first issue mainly is due to the high cost and complexity of the vast outdoor geo-data collection, e.g. a GPS map vendor may update the map 1 – 3 times a year or whenever it is necessary. The Google Maps with Street View may update every 2 – 3 years! For a developing city, the map update frequency is definitely slow. The second challenge is the unexpected incident such as bad weather, traffic jams or accidents on road that could lead to temporary road block. In such a condition, a user may not be able to arrive at the destination in time if he/she follows the navigator and gets to the congestion area. Nevertheless, these issues baffle indoor navigation as well. There may be incidents like interior renovation, structure modification or lift malfunction which causes alteration of layout or path unavailability. After considering the diversity between indoor and outdoor navigation systems, we can conclude that outdoor navigation assistance can be used for a wide spectrum of activities, while indoor navigation systems are valuable under some special circumstances. These circumstances where indoor navigation is more practical consists of visiting unfamiliar complex (university, shopping mall, hospital, etc.), path finding in emergency, and a building’s interior on show such as museums, retailers, houses listed for sale by real estate agencies.

From the functionality standpoint, indoor navigation systems carry out similar functions as those implemented in outdoor navigation systems. In a typical navigation task, they all perform positioning, mapping, direction guiding / routing, but, are different with respect to the system requirements of directions and routing, the physical space in which navigation takes place and some of the technologies utilized [7].

Some technologies which are compatible to both outdoor and indoor navigation systems have been designed and there are also technologies specifically developed for indoor navigation systems. Positioning sensors, such as Wi-Fi and RFID, are now widely installed in buildings. They are usually used to form a sensor network. Such a network can provide geo-position data to indoor navigation systems. In a relatively small outdoor area, if density of sensors is high enough, the network is also able to provide information to outdoor navigation systems. However, these devices are not specifically devised for navigational purpose. Drawbacks, such as low accuracy and signal strength arise when they are implemented in navigation system, and these drawbacks lower the overall system performance. So, considering the level of efficiency and accuracy of these sensors, a different infrastructure which is specially designed might be more suitable for the indoor environment.

The difference in the physical space in which navigation is carried out is the major reason that technologies as well as routing and directions approaches required for indoor navigation differ from those used in outdoor navigation. The physical space for outdoor navigation usually comprises road and sidewalk networks, which are structured on the basis of standard features. However, it’s not easy to find such consistent characteristics in indoor area [7]. Generally, similar structures and layout are applied to the design of roads, sidewalks, as well as city blocks. For instance, structures and layouts of roads in different cities and countries are alike. In contrast with outdoor navigation, indoor navigation usually is performed in buildings where common structures are hard to find among different buildings, e.g. buildings might have distinct layout of floors (layout may vary from floor to floor even in the same building), different number of floors, different heights, variant exteriors and interiors.

1.3 Problem Statement

The justification of the need of navigation systems for indoor environments was discussed in the previous section. We now consider whether current outdoor / indoor navigation techniques fulfill the demand.

It has already been discussed that the dominant outdoor navigation technique, GPS, does not fit well into indoor navigation systems because of its sheltered signals in indoor environments. Therefor a number of indoor navigation systems, based on passive / active sensor network or reckoning technique that overcomes the signal availability issue, have been discussed in the literature. The main limitations of these are the high installation cost and the complexity of the system design. Additionally, most of the existing positioning systems do not provide an accurate position in large buildings [5, 12]. The accuracy usually depends on the density of the sensors in a network, which means high accuracy cannot be achieved unless there is a big investment in sensors. Thus, there is a strong requirement for an efficient and low cost indoor navigation system inside large buildings consisting of many rooms, floors and large halls.

Image based navigation [9, 10, 11, 13, 14, 15] satisfies the indoor navigation requirements. Because of the characteristics of the images, this technique supports users with rich visual information and enables a useful and novel user experience. It also provides relatively low cost indoor navigation service with relative accuracy. However, the existing image based navigation systems as well as systems proposed in the literature do have some limitations. Firstly, they usually implement sampling approaches which require expensive equipment and specific image data, such as panorama cameras and panorama images [9, 15], which are inaccessible to ordinary users. Secondly, a majority of the existing systems only provide localization but cannot come up with direction instructions, so they are not truly a navigation system. Lastly, current image based navigation systems do not efficiently use the photos uploaded by users. Basically, the query image is discarded after user's position is confirmed. The possible new knowledge learned from the uploaded images thus fails to be reused, which would have been very helpful to enrich and expand the current system. The gap in current literature can be articulated as: *The lack of interactivity between users and the* *navigation systems as well as improper image collection techniques hinder the user experience and the popularity of current indoor navigation systems.*

Thus the research questions formulated are: *what approaches can be used to guide users to destination by using ordinary images taken by mobile phones? And how is the new information from query images to be utilized to perform more interactive search?*

1.4 Contributions

In this thesis, we propose a prototype of image based indoor navigation system named iNavigation for mobile phones (particularly smart phones) which complements Google Street View and extends image based navigation to indoor environment. The proposed application aims at delivering a robust, interactive and visual service.

We first constructed an image database with absolute geo-data encoded (Our own location data rather than GPS data). We have taken photos with a smart phone built-in camera of the first floor of a building. The photos were collected and labeled in groups so as to show direction. In addition, we provided annotations such as location and orientation of the image. Users can virtually "go" to different locations inside the building using sliding images.

Furthermore, we built an image retrieval system consisting of image feature extraction based on SIFT and high dimensional data search module. A comparative experiment was conducted to evaluate the two search algorithms: ANN (Approximate Nearest Neighbor) and LSH (Localize Sensitive Hashing). It was found that both enable efficient image matching on a large dataset. Meanwhile, an image similarity ranking approach was also examined in the same experiment. We combined our dataset and system to form a basic image based navigation system that enables users to perform navigation and localization. We then constructed a path finding function based on topological search and annotations stored in database.

In order to perform interactive navigation and enable users to share their information and knowledge of a certain location, we implemented Inverse Perspective Mapping (IPM) technique. It can roughly figure out the absolute distance between a region of interest and the lens by using an ordinary image. With the aid of this technique, the system is now able to “understand” where to insert the query image in the database by comparing the distances between images. The system also enables annotations as well as tags submission along with a query image. The new information can be self-added to the system with the intention of benefiting other subsequent.

To our knowledge, such an indoor navigation system built from ordinary images photographed by a mobile camera, with the capability of user knowledge sharing that enables interactive navigation has not been found in market or in the literature.

1.5 Overall Methodology

The main goal of this research is to develop a prototype that shows the practicability of our ideas. This fulfils the meaning of constructive research [16] which is concerned with designing frameworks or technical developments. So we follow the steps of constructive method. Our research steps can be summarized as follows:

- Formulate a problem that has research potential.
- Obtain a general and comprehensive understanding on relevant topics.
- Innovate a solution idea.
- Show that the solution works.
- Analyse the applicability of the solution.

We have formulated research problems with potential. We then went down to conduct a literature review, developing a prototype and evaluated the system to prove the practicability of our solutions. We reviewed literature relevant to our problem domain. For prototype development, we referred to software engineering methodologies and conduct qualitative research, quantitative research, and experiments in system implementation and evaluation chapters to show how well our system performs and reviewed methodologies that we could implement in this research.

It is important to note that we only take the performance of our algorithms and the system into account at this stage, such as accuracy and speed of algorithm / system. We are not going to dive into the performance as well as issues of the user interface design. Basically, all design is trying to fulfil the functionality proposed. However, some basic usability issues, such as system response time, would be taken seriously.

1.6 Organization of This Thesis

In Chapter 2, our literature review surveys the research work relevant to indoor navigation, content based image retrieval and reviews the feature detection, description, matching as well as some comparisons of the state of the art feature detector and descriptors proposed in the literature.

We discuss our development and research methodologies in Chapter 3. Chapter 4 shows the system specifications elicited by two approaches. System design and techniques implementation are detailed in Chapter 5 and 6 respectively. Our system is tested and evaluated in Chapter 7. Finally, Chapter 8 concludes this thesis.

Chapter 2
Literature Review

This chapter reviews the literature closely related to image based indoor navigation. We first consider some current indoor navigation techniques, and then we step into the topic of image feature detection and matching. Some recent findings and algorithms are discussed. Following this, an overview of the state of the art in content based image retrieval is presented. These three topics basically form the foundation of image based indoor navigation. Finally, some papers about image based navigation are also critically reviewed.

2.1 Indoor Navigation Technologies

2.1.1 Overview

This section presents the key concepts of some major navigation techniques used in indoor navigation and highlights the pros and cons of particular implementations. Generally, in indoor navigation working environment there is the absence of satellite signals. However, alternative techniques based on GPS have been developed to perform indoor navigation, these are discussed in section of navigation techniques. Also, satellite-based navigation techniques are discussed in order to provide a complete picture.

Current approaches for indoor navigation systems are usually categorized into independent navigation systems and network based navigation systems [17]. Network based navigation systems generally utilize networking technologies such as sensor networks so as to capture users' position and provide routing, while independent navigation systems focus on utilizing autonomous user positions.

Network based navigation systems use technologies such as GPS/A-GPS, Bluetooth [19], Ultra Wide Band (UWB) [17, 18], Wi-Fi [20, 22, 23, 33, 34], Radio Frequency Identification (RFID) [21, 24], infrared [25] or NFC [5]. Positioning accuracy varies according to the technology implemented. Usually, UWB and Wi-Fi techniques can deliver more accuracy positioning than Bluetooth and RFID [5]. Independent navigation systems use Dead Reckoning (DR) methods [26]. Two types of positioning data are required when a person’s position needs to be marked on a map. Fixed position determines the location with the aid of enough number of assisting devices, such as GPS satellites. On the other hand, estimation of current position is figured out according to the last fixed position, velocity of the object, its route, and time elapsed from last to current position. These approaches of retrieving position data are implemented in navigation systems based on DR methods.

2.1.2 Technologies

Satellite Based Techniques Satellite navigation systems provide global positioning coverage. GPS is one of several satellite systems that work on delivering global positioning service. A major advantage of GPS is that receivers can get accurate position of longitude, latitude and altitude anywhere on earth, as long as they can receive satellite signals. The accuracy is approximately 5-6 meters in open space. GPS personal navigation devices are now inexpensive and easy to use. They deliver positioning service from pedestrian street navigation to car driving navigation.

However, GPS requires line of sight (LOS) to be fully working. This disables GPS to work indoors where LOS is blocked by the building body. Alternatives, such as pseudolite (contraction of pseudo-satellite) and A-GPS have been developed to overcome the LOS limitation.

The pseudolite system imitates GPS satellites by sending signals which are similar to those of GPS to receivers inside the building. The system can achieve sub centimetre accuracy [27]. It also has a very short convergence time which shortens system response time for mobile users. The A-GPS [29, 30] technique works by directing mobile devices to a particular available satellite. This process is primarily done by a third party, such as a cell phone network provider. This method is useful when satellite signals are weak or unavailable. But it is not quite precise, typically around 10-meters accuracy [28].

Bluetooth is a wireless communication approach that works over a short distance. A connection between two devices can be established by this technique. It enables information transmission, which has a maximum of 3Mbits/second. The standard of this technique is similar to Wi-Fi, which is IEEE 802.15 standard. The maximum communication distance can be up to 100 meters, for class 1 Bluetooth set. Ordinary Bluetooth set only works within 10 meters. Work shown in [19] describes a positioning network based on Bluetooth. The major limitation of such an implementation is the high cost.

Ultra-wide band (UWB) has been used in positioning and now is receiving increasing attention because of its accuracy [17, 18]. Its primary advantages are low-power density and wide bandwidth. Both advantages result in high reliability for indoor usage. A signal is less likely to be sheltered if signals in a broad range of frequency can be broadcast. It is also robust to interference of other signals in the same area. However, like other network based navigation systems, the implementation of this technique requires installation of extra devices in the building.

Wireless Fidelity (Wi-Fi) is the conventional name for IEEE 802.11 standard. Today, the availability of wireless service is more widespread than ever before. The wireless local area network (WLAN) has also been widely deployed in buildings. Such a network consists of a number of wireless routers or access points which broadcast signals to all users in a certain area. The wide use of WLAN draws the attention of developers to work on indoor navigation using wireless network [20, 22, 23]. There are a number of positioning systems that make use of WLAN. Some of them are based on Received Signal Strength (RSS) [33, 34], a metric that is used for wireless devices to measure the signal strength. Usually, RSS is converted into a number, which is the indicator of RSS (RSSI). Wireless devices such as smart phones and laptops can detect multiple RSSI and MAC addressed sending from multiple Access Points at the same time. The metric can be used by navigation systems to determine a user's location. Other applications based on WLAN normally implement techniques such as, time of arrival (TOA) [28, 32] or time difference of arrival (TDOA) [28, 31] to carry out localization. But they require configuration beforehand. One of the major drawbacks of WLAN based indoor Navigation system is that the accuracy is not high when compared to UWB and pseudolite. However, it is relatively inexpensive for implementation.

Infrared (IR) wireless network is a relatively new technology used in indoor navigation [25]. It shows surprisingly good accuracy with respect to indoor navigation. However, this application has some fundamental issues which are mainly due to the intrinsic shortcomings of infrared wireless network The first issue is that the working range of an IR network is quite limited. Another issue is that this network cannot provide data transmission. One implementation based on IR network is the Active Badge System. The location of a person is determined by emitting IR signal every ten seconds from a badge they are wearing. Then the signals are detected by sensors installed at various locations inside a building and return produced signal to a location management system. The accuracy of the system is good, but it suffers from problems like high sensor installation cost and short working range.

Optical methods are techniques which require image analysis by the navigation system. In optical method, either still images or continuous images provided by cameras can be used as visual information.

There are quite a few approach based on this technique. Such as methods using optical marker detection, e.g. quick response codes (QR codes) or encoded markers; complex scene analysis, e.g. line detection in corridors or even object recognition in images.

2.1.3 Mapping Techniques

Mapping the inside of a building requires two processes. First, information of building's layout must be gathered. Then such information has to be converted to a form which is interpretable to human. The information may consist of following data: (1) The location of building interior structure, such as walls, staircases, lifts, corridors, floors, etc. (2) Relative position of current map to other locations. There are two forms that can be used to present building interior information: 2D map images and 3D building models.

2D maps are in wide use for aiding navigation, such as a map showing fire escape routes. For map readers, the information interpreted from a 2D map representation is mainly the floor layout. Usually, coordination and scale are not indicated on 2D floor layout images, but they can be seen on more professional drawing maps, such as building blueprint. The creation process of a 2D map varies depending on the raw data. Some work [36, 37] has shown that CAD files can be used to demonstrate indoor mapping in navigation systems. Because CAD files already decompose the building structure into small elements, the map creation is simplified. If CAD files are not available, image processing techniques might have to be used to map the raw data to points, arcs and lines to image files [35]. This could involve object recognition or image filtering, which are quite complex.

Building interior presented via 3D model contains more information than 2D map images. Height, direction, scale, etc. are all available to users. This is the intrinsic advantage of 3D modelling. Additional location and orientation information can be added afterwards. Some navigation applications [38, 39] implementing 3D modelling have been developed and demonstrated. The major drawback is that the model creation is very troublesome and requires some extra equipment to map the actual scene into 3D models. The state of the art in this field is to tackle real time indoor 3D scene modelling [40, 41]. The first approach [40] based on colour dense depth information, is only able to generate low quality 3D maps. The latter one [41] can produce 3D scenes of good quality but require additional panorama cameras.

2.2 Image Feature Detection and Matching

2.2.1 Overview

Robust image feature detection, image matching and 3D models reconstruction from 2D images are topics in computer vision that have been discussed and researched. Initial exertions towards digital image feature recognition were limited to the detection of edges and corners. In many cases, detection of these features is not sufficient for elaboration of deformed image matching as well as 3D models reconstruction. These tasks require detection algorithms that achieve true scale or Affine invariant, which is one of the most important challenges in computer vision. Therefore different algorithms were coined to focus on matching textures, colours, shape and segmentation. But no prominent progress was made until the end of last decade. In 1999, Lowe [42] made a breakthrough in the design of feature detection algorithms. He describes the detection of scale invariant features in images in a timely fashion. Inspired by Lowe's work, a number of its variants and other competitive advanced algorithms were proposed over the last few years.

Such algorithms consist of three different steps [43]. First the "detector" distinguishes locations in the image which store distinctive and well defined visual information. Secondly, a vector featuring local visual information is computed to describe the feature detected in the first step. Finally, a given vector is associated with another vector computed from another image by a matching algorithm.

An ideal image feature matching algorithms should be able to identify a large number of significant features in an image and match them robustly in different conditions, such as changes in rotation, blur, viewpoint, illumination and scale [47].

2.2.2 Local and Global Features

In computer vision, a feature is interpreted as a description of a certain amount of assessments where each specifies some measurable characteristics of an object in the image, and is calculated so as to quantify the predominant characteristics of the object. A set of features can describe many characteristics such as corners, colour, texture, shape or structures [47]. Such kinds of image data are extremely useful for evaluating and comparing images. Essentially, they are a compact representation which is able to infer an image's properties. The major image features commonly include intensity, colour, and texture. Usually, features are classified into two types: general features and features in specific domains. The first comprises some independent features such as colour, shape, texture, intensity. These kinds of features are irrelevant to the problem domain, while the latter one comprises problem domain dependent features such as cars, human faces, hands, fingerprints or special patterns. These features can only be extracted and implemented in specific data. If we take a step further, taking feature extraction level into account, the first kind of feature i.e. general features, can be divided into two categories: local features and global features.

Local features A local feature is a kind of visual information which is distinctive from its neighbour information [47]. Normally it is related to the change of image properties, such as colour, intensity, spectra, etc. It may be located exactly where the change is, but not necessarily. Basic local feature types include interest points (also refer to corners), regions (also known as blobs). Sometimes, edges of objects can also be regarded as local features.

Image patches are usually used to evaluate local features in an image. Such patches contain meaningful information in the context of the image, which can then be computed as "descriptors" in high dimensional space. Research shows that such features may not be robust when geometric deformations are applied on the image.

Global features Global features are evaluated over the whole image or a sub-area of the image [47]. Generally, they present statistical facts of the image. Such features include mean pixel value, spectra, shape and colour histogram, and so on. Global features have been used to evaluate images in a variety of research fields. In image retrieval, image content is described by colour histogram [44], although it mixes foreground and background together. In object recognition, a good example of the global feature implementation is the work done by Murase and Nayar [45]. They used the PCA (principal component analysis) algorithm [46] to evaluate a set of training images and model the principal components as descriptors.

The results of their work were pretty good. However, global features have limitations in dealing with occlusion and clutter of images.

2.3 Local Feature Detector

2.3.1 Overview

A Feature detector can localize features in images by analysing the local neighbourhood of pixels [47]. In order to analyse the robustness of local feature detectors under different geometrical deformations, the detectors are usually categorized as scale invariant detectors and Affine invariant detectors; The first detector can perform in a stable manner under different scale and rotation; the latter detector is capable of detecting features robustly on Affine transformed images.

Harris [48] coined a detector based on derivative for detecting edges and corners. It works by identifying and measuring the gradient distribution matrix around interest points. Mikolajczyk and Schmid [49, 50, 51] applied Hessian Matrix as well as Harris function to locate and obtain maxima as interest points based on multi-scale space Laplacian operator selection. These methods were named Hessian-Laplace and Harris-Laplace respectively.

Brown, Szeliski, and Winder [52] proposed the multi-scale oriented patch detector. It carries out the same operations in a pyramid at different resolutions to extract features at multiple scales. Then features detected at the same level are processed by the match algorithm. Their approach is quite effective when query images that are successively taken.

Lowe [42] presented a novel algorithm for object recognition. It works by determining the local extrema in the scale-space constructed with different Gaussian (DoG) filters. The query image and the reference image are continuously sampled and smoothed by a Gaussian kernel to produce the DoG representation. The location and scale of the interest points in the image are determined by the local extrema in the DoG representation [42].

A different approach based on entropy extremum was proposed by Kadir and Brady [53]. The idea uses local complexity to evaluate saliency. The scale of saliency is determined by the entropy extremum in the descriptors rather than the descriptors themselves. Then features with high entropy are selected by the approach.

In practice, most applications require algorithms to deal with scale changes, which is most likely to be in-plane rotation. A better resolution to this problem is to figure out the dominant orientation for detected key points. A feature descriptor can be computed based on the scaled and oriented patch formed using local orientation and scale information of a key point [54]. This approach evaluates the histogram of orientations computed.

Affine invariant detectors not only deal with orientation and scale changes, but also perform robustly across Affine transformations, such as viewpoint change. Full Affine invariance is important for the performance of applications like wide based line stereo matching.

Discussions about the possibility of applying an ellipse to Hessian matrix or autocorrelation for figuring out the principle axes and ratios which can be used as Affine coordinates have been illustrated in [47, 50, 55, 56, 57]. Matas et al. [58] used the watershed algorithm to detect intensity regions, and then applied an ellipse to the evaluated boundaries. MSERs (Maximally stable extremal regions) are detected in binary regions which are computed via thresholding the image successively with all possible grey levels. While the threshold is being changed, the region of each component would be monitored; regions whose rate of change with respect to the threshold is minimal are defined as the maximally stable regions.

Recently, a novel Affine invariant detector based on DoG detector named Affine-SIFT [59] was proposed. This algorithm simulates sample views on the query image by changing the orientation parameters of two camera axis. The parameters include longitude and latitude angles, which are not considered by other detectors. The DoG detector is applied on all the sample views generated to produce a large number of features. This algorithm is proofed to be fully Affine invariant. However, its efficiency is not remarkable.

2.3.2 Detectors Based on Gaussian Derivative

A Gaussian Kernel is used to generate the scale-space representation of an image. It is represented by $G(x, y, \sigma)$. Lindeberg [60] stated that the only possible kernel that can be used to generate scale-space representation is the Gaussian Kernel. Various levels of resolutions are generated by doing convolution with Gaussian Kernel, which is shown in equation (2.1). In the equation, $I\,(x, y)$ is the input image intensity and $*$ is the convolution operator,

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{2.2}$$

Differential Expressions Construction The first phase of feature detection for detector implemented Gaussian Kernel is to identify and trace points in various scales which are invariant to scale change, such as rotation, scaling. It can be done by Harris function which was developed by Harris [48], furthermore it was used in Hessian-Laplace and Harris-Laplace [49, 50, 51]. Linderberg [60] applied Laplacian operator in his region detector and Difference-of-Gaussian was used in SIFT by Lowe [42, 54].

Scale-space Extrema Determination The process of local extrema detection usually involves comparing the points with their eight neighbours in current query images and nine neighbours in adjacent scale. This approach has been seen in Linderberg and Lowe’s work [42, 54, 60]. A similar approach based on trace and determinant of Hessian matrix as well as Harris-Laplace is used to find interest points, then Laplacian differential expressions were implemented to identify the local extrema among those points [49].

Affine Adaptation This process based on the second moment matrix was firstly developed by Lindeberg and Garding [55]. It enables a detector to work against Affine transformation. Their idea was to apply an Affine transformation from Affine scale-space to scale-space over the second moment matrix which is able to figure out the anisotropic form of an image. Acquired interest points and regions are only differentiable by an arbitrary 2D rotation. Mikolajczyk and Schmid [50, 51] created Harris-Affine and Hessian-Affine detectors by applying the approach to their Harris-Laplace and Hessian-Laplace detectors. Baumberg [56], Schaffalitzky and Zisserman [61] presented similar detectors.

2.3.3 MSER Detector

This approach is based on a segmentation algorithm which is similar to watershed segmentation [58]. It is an Affine-invariant blob detector. Homogeneous intensity regions are extracted in this algorithm. These regions are stable over a wide range of thresholds. Then ellipses like shapes with moments up to second-order are fitted into these regions. A Maximally Stable Extremal Region is a connected element of a properly threshed image. Maximally stable could be defined as the local minimum of the area change as a relationship of relative change of the threshold. However, no optimal threshold is sought after. Only thresholds and the stability of MSE region is taken into account. MSER detector outputs nested subset of MSE regions if multiple stable thresholds exist.

2.4 Local Feature Descriptors

Features need to be described and quantified before they are able to be identified and matched. This process also needs a local neighbourhood of pixels which is similar to the process used in feature detection. When an interest point is successfully described by its neighbourhood of pixels, the feature extraction algorithm will be able to figure out not only the location of the interest point, but also the shape and size of this neighbourhood of pixels.

2.4.1 SIFT Descriptor

SIFT algorithm implements DoG as the interest point detector but the feature representation is based on gradient distribution. SIFT descriptors are invariant to scale, rotation, illumination, blurring and partially invariant to viewpoint change. In order to achieve invariance to rotation, orientation and scale change, a primary orientation is evaluated for each feature and the local image is rotated to that orientation so as to compute descriptors [54]. The size of SIFT descriptor is controlled by the size of orientation histogram array ($n \times n$) and the number of orientations in each histogram (r) [54]. So the size is $r \times n^2$. The influence of various descriptor sizes on performance was studied in [54]. 128D SIFT, such as $n = 4$ and $r = 8$, stranded out the other sizes with respect to the matching accuracy. Descriptors in smaller size lead to less memory usage and data with smaller dimension [62]. But accuracy would be negatively affected. The procedures in feature representation of the original SIFT implementation is demonstrated as follows:

1) First, the orientation and magnitude of the points' image gradient in a 16×16 region around the key point is detected and quantified. The Gaussian blur level of the region is most similar to the scale of the key point [54].

2) Elements are then grouped into 4×4 sub-areas to present the key point descriptor. Each sub-area has 8 arrows pointing to different directions. The arrow’s length represents the sum of the magnitudes of points in the sub-area near the arrow [54].
3) After the interest point is localized, the coordinates of the descriptor is rotated to the orientation of the key point. This process enables the descriptor to be rotation invariant; a feature vector is also normalized to a unit vector. The descriptor will become invariant to illumination change after this process [54].
4) Finally, threshold all values in the unit vector lager than 0.2. The new vector is then normalized again.

2.4.2 SURF Descriptor

SURF algorithm implements Hessian matrix for fast feature detection. In terms of feature description, it is based on Haar wavelet calculation. SURF descriptor behaves similarly to SIFT descriptors. The descriptor first computes Haar wavelet around a given interest point, then the distribution of pixels’ intensities within a scale dependent neighbourhood is presented [63]. This process is applied to all interest points. Detailed procedure is shown below:

1) The reproducible orientation of Haar wavelet response is calculated along the vertical and horizontal direction. This process is applied on a circular region around the interest point.

2) The predominant orientation is then determined by the sum of horizontal and vertical wavelet responses within the region. The vector with the largest sum presents the orientation of the wavelet.

3) Afterwards, a rectangular region is applied on the neighbourhood of the interest point. The size of the region is 20*s* [63], where *s* represents the scale factor of the interest point which is decided by the feature detector. The region is then orientated to the predominant orientation of the interest point and split up to 4×4 small square sub-areas. Both vertical and horizontal Haar wavelet responses are computed in each square.

4) Finally, these responses are summed up in each region. The vector v = (Σd_x, Σd_y, $\Sigma|d_x|$, $\Sigma|d_y|$) [63] of each square region forms the descriptor.

2.5 Image Matching Algorithms

Once the features from the reference image and the query image are all extracted and presented, we need to establish the initial feature matches between images. The similarity is usually measured by a distance metric [64], such as Hamming distance, Euclidean distance, Manhattan distance, absolute distance and cosine similarity, etc. The matching step normally begins with the selection of the distance metric. Afterwards, an efficient data structure and algorithm should be used / devised to perform the matching in a timely manner.

Finding the Nearest Neighbours with Tree Structures Though matching a correspondent point is important it is usually computationally intensive. If an image contains *N* patches, an exhaustive linear search which goes through all *N* patches would take $O(N^2)$ time. That can take up to a few hours. However, we can use a Nearest Neighbours (NN) algorithm to get the same result in just a fraction of the time.

***k*-d tree** is a very common tree structure used in nearest neighbour search. It is usually constructed by successively dividing the search space into two sub-spaces. Each of them contains half of the parent dataset. This process continues until it meets a predefined stop condition. A query then is performed by traversing the tree from the root to a leaf node by evaluating the query at each branch. The dimension with maximum variance is selected to break down into smaller trees in each loop. Some intrinsic defects of *k*-d tree are:

1) The number of neighbours increases exponentially with dimension in each leaf node. Then the search becomes a linear exhaustive scan.
2) Ideally the dataset is supposed to be divided uniformly. However, it is quite rare in a high-dimensional dataset.

Best Bin First (BBF) search was then developed to resolve the first issue [65]. In this case, while a query traverses the *k*-d tree, its distances to every branching point are recorded in a priority queue. When backtracking, they are compared with the split distance of the points recorded in the priority queue. The backtracking stops either when the queue is empty or it gets the maximum number during the backtracking. So the nearest neighbour found is the one with minimum distance in this traversal. Although it can correctly find the nearest neighbour at very little computational cost [54], there is no guarantee it can find the nearest neighbour in every case.

***k*-Means** assigns points to the closest of *k* centres by iteratively switching between centre selection and points assignment to the centres until the partition of the centres and points are both stable. It is usually used as a simple non-hierarchical clustering approach which requires users to select the *k* and initial centres carefully. Poor selection could result in bad partitions and occurrence of the local minimal.

Other methods Some new approaches have been developed to improve nearest neighbour search. Nene and Nayar [66] coined the slicing technique that applies a set of 1D binary search to the list of points which are sorted in different dimensions so as to select candidate points lying within a hypercube around the interest point. Grauman and Darrell [67] improved the indexing tree by reweighting the matches at different levels. This improvement enables the search to be more robust to discretization faults in the tree construction. Nister and Stewenius [68] construct the metric tree which matches features to a small set of ‘prototypes’ at each level of the tree. Their technique outputs visual words that can be used to classify query images from a database of millions of images very efficiently. Muja and Lowe [69] compared a number of nearest neighbour search approaches and proposed a new technique based on performing the priority search on hierarchical k-means trees. They also concluded that multiple randomized *k*-d trees give the best performance.

Outlier removal is a process that is used to verify and remove false matches after all initial matches are detected. In practice, once initial matches are established, applications search for additional matches, such as additional matches along epipolar lines or the area of estimated locations on the basis of global transformation. These require correct matches to be filtered out. RANSAC [76] is s very popular outlier removal approach. It generates a model by successively analysing a small set of matches, and applies that model on the overall data to verify inliers. Once an initial set of correspondences has been established, some systems look for additional matches, e.g., by looking for additional correspondences along epipolar lines or in the vicinity of estimated locations based on the global transform.

2.6 Previous Studies on Detector and Descriptor Evaluation

Schmid et al. [70] performed an extensive study on feature stability by performing evaluations on feature detectors. They implemented some painting images photographed from different viewpoints as the test dataset. They generate the ground truth data by finding the homography between pairs of views. It is computed from a set of points projected on the paintings beforehand. Interest points are extracted and matched across the view pairs. The results were analysed by drawing repeatability curve. Mikolajczyk et al. [57] carried out a similar research on Affine invariant detectors. The scene in the image dataset was either photographed from distance or planar. So the scene looks flat in most images. Mikolajczyk et al. [57] calculated the homography (ground truth) between pairs of views using the same method proposed in [70]. But this time, they used manually selected correspondent points instead of projecting points to compute the homography. The performance results concluded from both studies show that only very few features are detected in each image. The performance of stability is good, the detector reached high stability; but this stability does not apply to the entire image matching process. It can only keep high stability if the descriptor can reach high stability as well. So this shows the necessity of descriptors' evaluation [86].

Mikolajczyk & Schmid [71] performed a complementary study on performance of descriptors. In their research, two points are regarded as a match only if their computed descriptors are closer than a certain threshold *t*. Initial matches will be compared to the ground truth to decide if they are true matches. The ground truth was computed in the same way shown in their previous research. The final results were analysed by using recall-precision curves. If *t* is small, then number of accepted initial matches drops, which causes a good precision rate but poor recall. If *t* is high, more false matches will occur, which lowers the precision. Ke and Sukthankar [72] implemented a similar approach by using threshold to decide matches to evaluate their PCA-SIFT descriptor against SIFT. Extracted features were indexed into the database. The matches were then examined; they would be accepted if they meet the threshold. Recall-precision curves were generated by varying the threshold. Synthetic data was created in this research for the ground truth data [86].

Mikolajczyk et al. [73] evaluated combinations of detectors and descriptors by integrating them in a recognition system. The advantage of this integration is that the bottom line performance of the system can be directly computed. Overall system with different detectors and descriptors has been evaluated and scored. It is clearly noticeable that if we move the algorithms to a recognition system with different architecture, the results may be changed.

Babbar [74] discussed the area based matching as well as the feature based matching, which extends the scope of texture based and feature based algorithms. This study compared both types of algorithms in several criteria, such as speed, sensitivity, occlusion and others. Babber's study showed that feature based algorithms performed slightly better than area based algorithms. One major issue of this study is that no detail of the dataset is given, which makes the study a little unconvincing.

Luo and Oubong [75] carried out an experiment to evaluate the performance of SURF, PCA-SIFT and SIFT on scaling, rotation, blurring, illumination changes and viewpoint change. This is a comprehensive study. It concludes that SIFT is the most robust algorithm among the three candidates. SURF is a great deal faster than the other two and managed to keep a decent performance. While PCA-SIFT showed its advantage in rotation and illumination change, its performance is worse than the others on other test dataset. Luo and Oubong also gave an analysis of the images fitted to a given algorithm along with recommendations.

Majority of research showed that SIFT is quite robust on various datasets, although it is not quite efficient. In terms of detector and descriptor combinations, it seems better to implement an Affine invariant detector plus the SIFT descriptor.

After going through the literature, we summarized pros and cons of major local feature detector and descriptor as shown in Tables 2.1 and 2.2:

Table 2.1 Image matching algorithm performance

Name of detectors	Type	Advantage	Disadvantage
Harris	corner	It runs fast and can detect edges and corners. Partially rotation invariance	Scale, Affine variance.
Hessian	region	Rotation invariance	Scale, Affine variance.
Harris-Affine	corner	Multi-scale approach	Rotation, illumination invariance. Problems with blurring.
Hessian-Affine	region	Good on detecting Affine regions	Relatively slow
Harris-Laplace	corner	Scale and invariance.	Problem with Affine regions.
Hessian-Laplace	region	Scale invariance	Rotation, Affine variance.
LoG	region	Scale invariance	Problems in single scale operating. Rotation, Affine variance.
DoG	region	Scale and rotation invariance.	Partial Affine invariance. Relatively slow.
MSER	region	Good on finding wide base line image correspondences (strong Affine invariance)	Partial rotation and scale invariance. Poor performance on illumination change and blurred image.
FAST	corner	Low computational complexity, very fast.	Scale, rotation, illumination variance.
ASIFT	region	Affine, scale and rotation invariance.	Very slow.

Table 2.2 Image matching algorithms invariance performance

Feature Descriptor	Rotation invariance	Scale invariance	Affine invariance
SIFT	Yes	Yes	Partial
SURF	Partially	Yes	Partial
GLOH	Yes	Yes	Partial
DAISY	No	Yes	Partial
BRIEF	No	Yes	No
ORB	Yes	Yes	No
SPIN	Yes	No	No
RIFT	Yes	No	No

2.7 Content Based Image Retrieval

Multimedia Information Retrieval (MIR) is a research area that deals with the search and extraction of knowledge in all kinds of information, such as audio, image and video, from a database. Such a database is not a conventional database which only allows textual search, it stores multimedia files. Content based method has been a fundamental technique to retrieve the desired information in a timely and precise manner from designated media databases. Research on improving this approach is still ongoing. Early work towards digital information extraction began with the commencement of digitization trend that overwhelmed physical media such as newspaper, books, and vinyl records. From a theoretical point of view, the development of MIR benefits from contributions of research fields such as computer vision, pattern recognition, artificial intelligence and optimization theory.

Content based image retrieval (CBIR) is a subfield of MIR. It mainly helps to organize digital image dataset using visual content. For a given database, it searches for images which are similar to a query image given by a user. There are two main intrinsic problems for CBIR technology:

- How to depict an image mathematically.
- How to evaluate similarity between two images on the basis of their abstract representation.

A typical CBIR system consists of three subsystems:

- Image acquisition system. It is mainly a user interface that allows users to input query images.
- Feature detection system. Feature extraction is then applied to every image to produce feature vector.
- Similarity scoring / matching system. The feature vector of query image is compared with other feature vectors. Normally threshold / distance metric approach, brute force linear search, K-NN search or high dimensional searches and other matching strategies are used to perform comparisons.

Early applications of image search usually worked within a specific problem domain. For example, face, fingerprint or shoeprint image retrieval in particular databases for biometric or forensic application; medical image retrieval for diagnosis; characters / script recognition in textual data and image document retrieval; location image retrieval which is used to aids robots to automatically travel a terrain without hitting any obstacles have all been researched. Nowadays, with the rapid growth of the Internet, the state of the art of CBIR is tackling the image search in mass media over Internet. In the following sections, we go through some CBIR approaches and applications of the state of the art.

Text search based method Sivic and Zissermann [77] developed a novel image search technique by assigning feature descriptor vector to a vocabulary of *K* words. Then they could use *TF-IDF* weighted vectors $V = (t_1,\ldots,t_m,\ldots,t_n)^T$ to describe the image. Afterwards, searching over the database is equivalent to comparing those *TF-IDF* weighted vectors.

Chun et al. [78] proposed the query expansion approach to improve image retrieval based on query texts. Other than the query image given by the user, the method also implements the dataset with the same objects as the query image. So the query image is expanded by these images already in the database. It is reported that this method improves the retrieval accuracy significantly [78, 80]. David Nisteur et al. discussed the possibility of improving image search based on tree structure. According to their method, first, the descriptors were hierarchically valued and populated in a tree structure; then the image similarity is also measured and stored the same way as descriptors in the tree. The tree structure is named vocabulary tree. Its complexity is the same as image retrieval using approximate nearest neighbourhood search.

Hashing based method Ke et al. [81] implements local sensitive hashing (LSH) in their application. It efficiently retrieves images from a database containing a million key points. Chum et al. [79] investigated a different way by presenting the image as a collection of visual words. The collection did not contain the number of occurrences of a word. Instead, it only records if a word occurred or not. They implemented min-Hashing functions in order to match the images.

2.8 Comparison of Our Work with the State of the Art

In this section, we review some recently developed image based indoor navigation systems. Differences between our work and the state of the art will be described.

Kawaji et al. [9] built an indoor localization system based on panorama images. They implemented omnidirectional camera to photograph images in a museum. All image features are extracted by using PCA-SIFT, which produces 36D SIFT descriptors. The image search is based on local sensitive hashing and RANSAC outliers filtering. 4000 images were taken and populated into an image database. The application gave very good precision rate (86% - 91%) in terms of finding locations in a timely manner. The average response time for image search is 3.11 seconds, which is quite fast considering the amount of images in the database. In general, the structure of their application is similar to ours, but it only provides positioning service and cannot reuse the query image. Therefore it cannot be used by users to navigate inside the building.

Chen et al. [92] proposed a visual-aided indoor navigation system. The main idea of their system is to estimate the motion change of users given a set of consecutive images. The motion is detected by a sensor, and the motion changes are then figured out by the given images. The proposed algorithm first detects lines from images, then it calculates the vanishing point where all lines are supposed to converge. This allows the application to guess where the user is heading to, in a 2D environment. One of the challenges is to determine the heading change. In their work, this is achieved by estimating vanishing point change. Finally, the algorithm is integrated with sensors by a Klaman filter. One limitation of their application is that if there is no structure with apparent lines that can be detected, the performance drops drastically. Another issue is that it is not purely based on images, it also requires extra sensors.

Werner et al. [14] developed an image based indoor navigation application without using extra markers (sensors). The main idea and system structure are similar to the work described in [9]. However, they introduced a distance estimation method to their work. It roughly estimates the distance by comparing the aspect ratio of the same object in different images. But one big issue is that if the viewpoints of the two images are not the same, the distance calculation is not reliable. Their application is quite efficient due to the implementation of SURF algorithm, which can detect features very fast. Our work also implements distance estimation, but it is independent on viewpoint changes.

This chapter reviewed related work on independent navigation systems and network based navigation systems. Research work on feature detection and matching and recent algorithm development were also discussed. In the next chapter, we discuss the methodology followed in the creation of a prototype.

Chapter 3

Methodology

Software development begins with a certain human need which can be articulated as a problem. We stated the research problems in the first chapter. Then we developed a comprehensive understanding on the topic after conducting a literature review. Next we had some ideas to solve the problems, and we formed them into reality by producing a prototype. This was achieved by software development, which involves eliciting system requirements specification, system design, system implementation and finally system evaluation. In other words, we also had to follow methodologies used in software engineering. Additionally, across the development, we used a combination of qualitative and quantitative research methods in particular parts.

3.1 Methodology for Requirements Specification

Identifying and understanding key requirements of the software is one of the main difficulties in producing high-quality software. In this research, we elicit system requirements using natural language as well as use case analysis.

Natural language is frequently used to articulate system requirements. It is considered the most understandable because it is the language in which we speak and write [88, 93]. This means anyone is able to understand what the requirements state. Because of this, requirements specified in natural language are understandable to stakeholders without much software development technical background. They are usually formatted in a contract style which lists all the requirements [88, 93]. A general form is shown below:

<entity> shall <description of a requirement specification>

The <entity> could be the system itself, the subsystem, certain conditions, development process, the contractor, the actors or a part of one of these. Secondly, sometimes, the “shall” can be replaced with “will”. “shall” is for emphasizing what has to be developed for the system, while the usage of “will” indicates what is expected of the system. Also the description of a requirement in natural language should be a complete sentence that has correct spelling and proper grammar. It would be better if the description is also concise and straightforward.

In this research, we follow the DoD (U.S. Department of Defence) standards [88, 93]. According to such standards, system requirements should be formatted in a fixed content. It is usually called computer software configuration item. The list below shows common requirements which are elicited in natural language for systems:

- Engineering requirements
    - User interface requirements
    - Capability requirements
    - Internal interface requirements
    - Data requirements
    - Security requirements
    - Design constraints
    - Human performance requirements
    - Quality factors
- Qualification requirements
- Requirements for delivery

Note that we are only expected to deliver a prototype for research purposes, not software ready for commercial market. So in the practice of eliciting system requirements in this research, we only focus with the part of engineering requirements. Some of the requirements closely relevant to commercial software development such as user interviews will be left out.

System requirements addressed in natural language have some defects. Firstly, although they are easy to understand by stakeholders, it is quite challenging to state the requirements precisely. Secondly, because the expression of the requirements can be freely formed, the requirements may lack check for completeness and consistency. Also, requirements addressed in natural language can possibly be structured poorly, which could cause confusion to the stakeholders. Finally, requirements are not detailed enough to be testable if they are stated using natural language. Due to these shortcomings, in this research we only implement natural language to address some general or high level requirements which are used for guidance. Detailed requirements elicited by natural language are described in the first section of Chapter 4.

We then employed use case methodology to derive detailed user and functional requirements.

Use case analysis is employed to complement the natural language approach. Since it primarily analyses how external users interact with the system, it is a good alternative to natural language as an approach to describe user and functional requirements [88, 93]. The goal of using this methodology is to output a model that includes complete UML diagrams, description of actors, use case and use case scenarios. A simple walkthrough [98] of how we implement use case analysis is given as follows:

- First we have to define the system boundary. It is used in use case diagrams to separate the internal elements of a system from the external entities.

- Then we come to the definition of the use case actor. It is an external entity that interacts with the system for certain goals. There might be different types of actors. For example, in our case, we have the actor doing localization, the actor viewing images, and so on. Each actor wants to achieve different goals.
- We are then able to collect the descriptions of actions between an actor and the system, which is the use case. Note that a use case only describes general behaviour rather than what happens when a specific action is performed by a specific actor. Those detailed executions are often captured in a use case scenario. We usually focus on the use cases that are more likely to take place when everything goes well, called "sunny day" use cases. Also sometimes it is necessary to derive the "rainy days" scenarios that are likely to happen, albeit rarely.
- Finally, we create a use case index which usually includes scope, status and complexity. The collection of use cases are also presented by a Unified Modelling Language (UML) top-levelled use case diagram. Possible relationship such as "include", "extend" can be presented in the diagram as well, if the system is object-oriented.

Unlike natural language, the use case analysis is a more professional approach for eliciting system requirements. It is better to be performed by a requirements engineer or people with such experience. So the use case diagram as well as use case index may not be written well by people without such specific background. Additionally, because use case is a scenario based approach, when the number of scenario increases the checking for requirements consistency becomes quite challenging [88, 93].

As shown in Figure 3.1 [88], the major advantage of use case is that it generates functional requirements and can be used for functional testing. Compared with natural language, it describes requirements more precisely. We present the top-level UML diagrams and the functional requirements elicited by use case in Chapter 4.

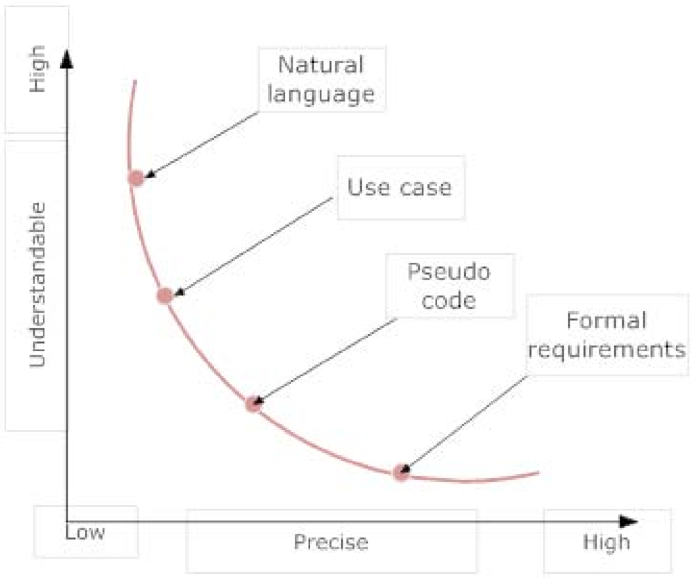


Figure 3.1 Relationship between different requirements.

3.2 Methodology for System Design

At this point, our proposed system runs as a web application. The client side is the interface which is supposed to be rendered in the browsers (not an application that directly runs on a mobile phone). It can be accessed by mobile phones via a wireless network. All computations are done at the server. There is no interface for support, such as administration tools, for the server side at this stage.

Regarding the development of the application, we adopted MVC (Model, View and Controller) pattern, more specifically the ASP.NET MVC framework. This methodology [87] is appropriate for developing dynamic and interactive applications. It separates the logic, data and presentation of the application into independent components. There are other frameworks derived from the MVC pattern, such as Structs based on JAVA and Django based on Python. They have their own features but at the high level there are always three major components that form a web application which are the model, the view and the controller. Object-Oriented Design can be implemented in MVC pattern. But in our development, we have adopted MVC without implementing Object-Oriented Design, as Object-Oriented Design is not appropriate for our application.

The View is what the browser renders and displays to the user. It also includes the interactions between the user and the browser, like click a button, or type in an input.

The Model component presents the dataset layer in the application level. This might not be a physical database. A web application usually stores and retrieves objects and logic of the application in this kind of storage.

The Controller At high level, it basically handles the user input and updates the model to wake the change caused by user action. The controller analyses the HTTP request coming from the browser, then dispatches and forwards the request to the handler of the model. After the request is processed, the controller sent the request back to the browser.

Note that the MVC pattern implemented in different frameworks basically stays the same. The main reason why ASP.NET framework was picked up is because I feel more comfortable with C#, which is the main language in ASP.NET.

In terms of system implementation methodologies, we didn’t apply any formal approaches in our research. Our application is a relatively small scale project and the development is quite flexible. Basically we follow MVC framework and achieve every system requirement.

3.3 Methodology for Data Acquisition

There are two types of data collected in this research. The photos showing the building interior and the actual distance from the location where the image is captured to the location where the camera faces.

Our proposed system requires a predefined image database that encoded with geo-data to execute its functionality. So we need to compile a collection of images which shows the inside of the building. This collection of images is used as the reference images for query images. Also, the corresponding geo-data has to be available to every image. The distance data we compiled is for the training of the Inverse Perspective Mapping (IPM) function. Thus we hope to get precise parameters for this function.

Some research has detailed their image collection approaches [9, 10, 11, 13, 14, 15]. However majority of them implemented special cameras rather than ordinary mobile phone built-in cameras. Hence, we developed our own data collection protocol which is quite simple and helps us to produce accurate data for the system.

It is necessary to state the issues first. In terms of image matching, the characteristic of indoor environment is that it is always full of self-repetitive structures as shown in Figure 3.2. Thus we have to pick up the most distinctive locations in this indoor environment for photographing, so that a query image can more likely be matched. Another issue is that the system shall enable user to navigate the interior of the building, and also locate their position by uploading their own images. That means there must be enough images in the database for navigation and matching purpose. But how many images should be enough? The answer was not evident in the literature. The solution we arrived at is explained in the next paragraph.

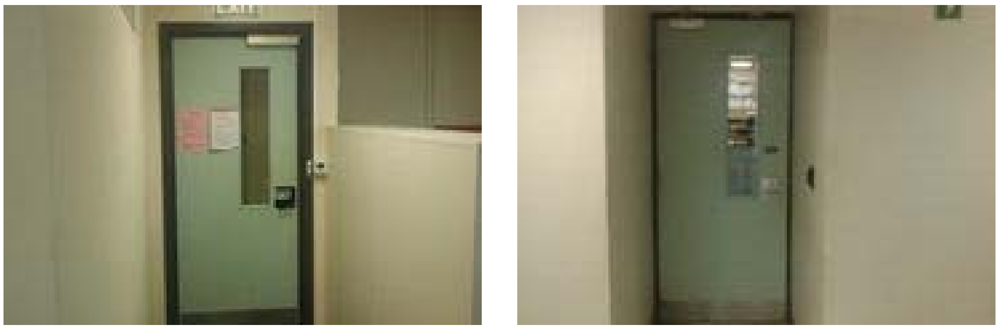


Figure 3.2 One example of self-repetitive structure.

The data collection was taken place at the first floor in our department building. Before we actually collect data, we did a “pilot run” first. The basic idea is to take photos along the corridors on this floor. We walked along the corridor on the first floor in a clockwise route to identify the “waypoint”. A waypoint could be the both ends of a hallway, the intersection of hallways or a particular location with distinct structure. A user can navigate through the entire floor by viewing a collection of images taken at the waypoints. So that fulfils the navigation purpose. We identified around 25 waypoints in the pilot run, which may not produce enough images for our purposes. So we took more images between the waypoints to support positioning function. How many images are needed is actually a problem. Our idea was that, extra photos between waypoints were taken at a certain distance interval. Here the interval between 5 and 10 meters, depending on the actual layout. Photographing images at such distance intervals can ensure the scene in the photos to be consistency, so users do not feel that they “hop” from one location to another location. It also gives enough space for a query image to be placed on a map. The area that is covered by the images is shown in Figure 3.3.

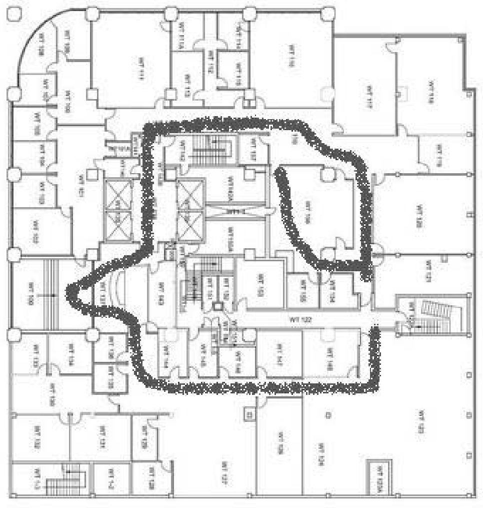


Figure 3.3 Image photographing route.

The black line shows our routes for taking photographs.

After we decided on all the locations at which to capture photos we started collecting the data. We used HTC Hero G3 smartphone with built-in camera to capture photos. The focal length of the camera is set to default. It is important to take photographs without changing the focal length because the focal length is a parameter in IPM (Inverse Perspective Mapping) function, so we have to keep it unchanged. For a given image, the focal length can also be retrieved by analysing EXIF metadata. However, the metadata is not available for images in PNG format, which the system also accepts.

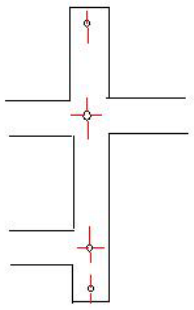


Figure 3.4 Image-capturing approach, where black lines form the corridor, the circles mean the locations where to take photographs, the red lines show where the camera is facing.

At each point, we took at least two photos using the mobile phone built-in camera. One was taken while we pointed the camera to front. Then we pointed to the back and took another photo. At certain locations, we may have to take 3 or 4 or even more images. Another important point is that the camera is held parallel to the ground when images are taken. It is good to retain the same value for all images because the camera tilt angle is one parameter in IPM function. Otherwise the distance calculated may not be accurate.

According to our image collection protocol we collected 112 images. Corresponding geo-data was to be assigned after the map is created. This is discussed in Chapter 6. Figure 3.4 illustrates how we take photos.

The data for distance calculation is relatively easy to be collected. It can be compiled after all photographing locations are confirmed. We collected 20 values by roughly measuring the distance between where we took images and the location the camera is pointing to.

3.4 Methodology for Testing and Evaluating the System

The system testing primarily focused on the aspects of function and performance testing. It is quite critical to perform both types of testing to ensure that the system does what it is meant to do and performs adequately. Techniques include black / white box testing and regression testing are used in the project. The testing was performed to track and discover bugs that could cause the system to crash or return incorrect results. Performance testing was also to be conducted at different software level; some of this is discussed in the system implementation part, and the rest in evaluation part. The methodology for system testing / evaluation can be broken down into three parts [96, 97].

Method testing After programming each method, function or even class, we use particular test approach to check its correctness. This testing is the foundation of the other high level testing. So we have to ensure the functions and methods work as expected. Some approaches used to detect faults are explained in the following steps:

- Errors in the code, such as typos, wrong function calling or misuse of variables are easily spotted. Sometimes even the IDE (Integrated Development Environment) can help us locate the errors.
- The method itself might not be well designed, such as loss of accuracy during value type cast (it is usually a warning), missing / wrong condition for if statement or wrong stop condition in a loop. White box testing identifies these errors.
- The idea behind the algorithm might be wrong. This is possible due to wrong selection of data structure or incorrect algorithm design. This is not easy to spot. A review of system requirements plus randomised testing may help us locate the fault and refine the algorithm [96, 97].

Module testing We may think that if all the functions in a module work perfectly, the module should work as expected. This idea usually is true. But there is no guarantee that it happens every time. Different function interface or thread confliction may cause the module to crash even though every function has been thoroughly tested. To test the module, we can treat the module as a whole and apply function testing approaches.

System testing / evaluations System level testing is always difficult because of its scale and testing workload involved. One possible difficulty we face is to trace the system status. In this case we use assertion test to ensure the correctness of system status. We also apply regression testing to validate the modifications done to the program. These modifications are recorded to help manage the testing and trace the changes [96, 97]. Good test cases can also help us to go over every function and spot the possible bugs efficiently. With respect to system evaluation, we mainly focus on the performance, especially the response time of each task.

Having considered and decided on the methodology for this research, we now focus our attention to system requirements specifications which is discussed in the next chapter.

Chapter 4

System Requirements Specification

System requirements specification is one of the most crucial steps in developing software. Without these, we don't know what to achieve at the system level; we are also unable to plan our development. In this chapter, we summarize and list the system requirements by using natural language and use case analysis approaches. The two approaches complement each other in the context of system requirements.

4.1 Requirements in Natural Language

The Single Statement of Need (SSON):

By using iNavigation, a user shall be able to locate his / her current position in a building by uploading a photograph taken by his / her mobile camera and get to the destination in a building by viewing a set of images shown to the user in a timely fashion.

We come up with system requirements by following the list shown in Chapter 3. Note that requirements such as internal interface requirements, security requirements and human performance requirements are not applicable in case, so we left them out.

User interface requirements

- The UI shall display a map, an image slide section, a thumbnail display section.
- The map shall indicate the location of current displayed image (where the image was taken).
- The map shall indicate the user's current location after a photo is uploaded and successfully processed by the system.
- The image slideshow section shall display clickable arrows indicating directions.
- The user shall be able to view next image by clicking an arrow.
- The user shall be able to upload images (one at a time) and submit annotations along with the image via the UI.
- The UI shall be viewable on a mobile phone (developed on a desktop).
- The UI shall return 4 most similar images for the user to choose.

- The UI shall update its image slide and thumbnail display section after selection of one of the three images is confirmed.
- The user shall be able to search a location by using search bar.
- The UI shall display search results once the key words meet its search criteria.
- The UI shall display a slideshow starting from current image to the destination image once the user click on a result.

Capability Requirements

- The prototype shall be capable of extracting features from query images.
- The prototype shall be capable of finding the images in the database which is similar to the query image in a timely fashion.
- The prototype shall be capable of roughly estimating linear relationship between the distance in the image and the real world (by using the IPM mentioned in Chapter 3).
- The prototype shall be able to update all datasets.

Data requirements

- Images in JPEG and PNG format are both accepted.
- Images shall be stored in server folder(s).
- Image link information, annotations and geo-data shall be stored in XML(s).
- Image features shall be stored in .mat file(s).

Design constraints

- The development shall not use any extra expensive equipment.
- The prototype shall not utilize data about actual distance for navigation purpose at this stage.

Quality factors

- System response time shall not exceed 15 seconds [93, 94] (A complex task might be break down into pieces to evaluate).

4.2 Use Case and Functional Requirements

We introduced the methodology for use case Analysis in section 3.1. In this section, we derive the use cases and functional requirements.

We define three user types with different primary tasks: 1) Users who just "navigate around", maybe just enjoy viewing the interior view of the building. 2) Users who try to find the path to the destination. 3) Users who need to know where he / she is. Figure 4.1 presents the overall system UML diagram.

This kind of classification can help us analyse users and functional requirements more specifically. Detailed use cases are presented in Figure 4.2, 4.3 and 4.4 respectively. Tables 4.1, Table 4.2 and Table 4.3 show use case indices of Figure 4.2, 4.3 and 4.4 accordingly.

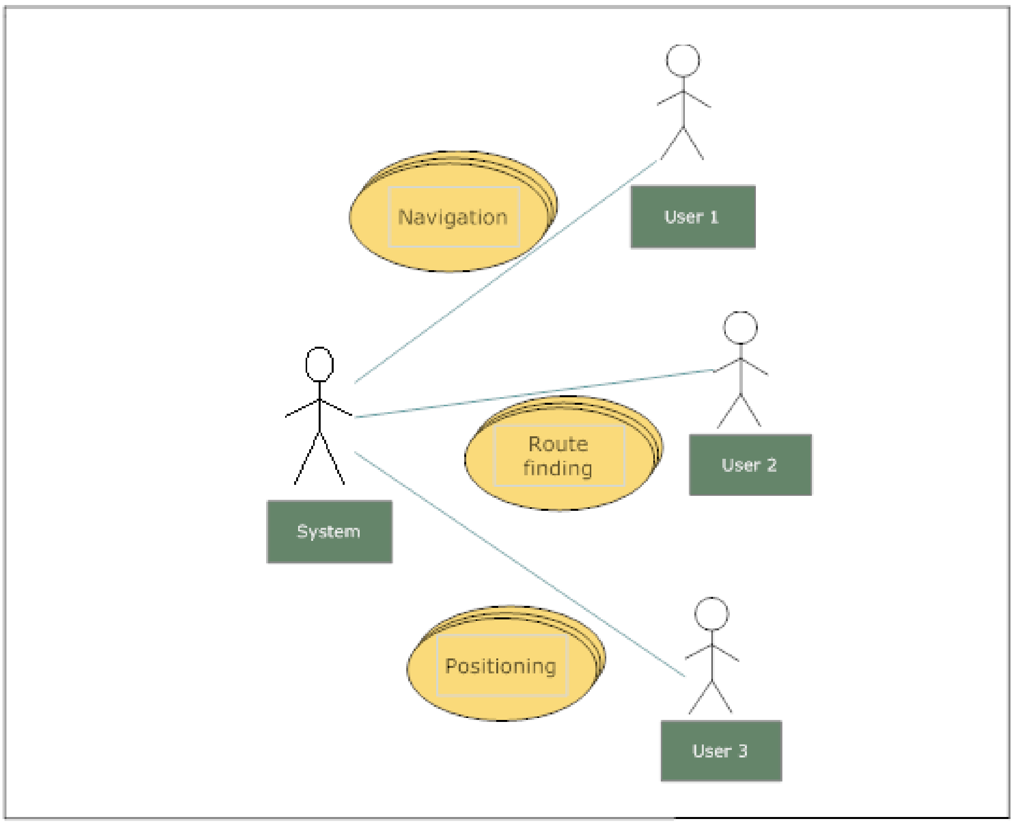


Figure 4.1 System UML diagram

This is the overall use case showing the goals of the three user types. The following use case breaks down the use case to more detail for the first type of user.

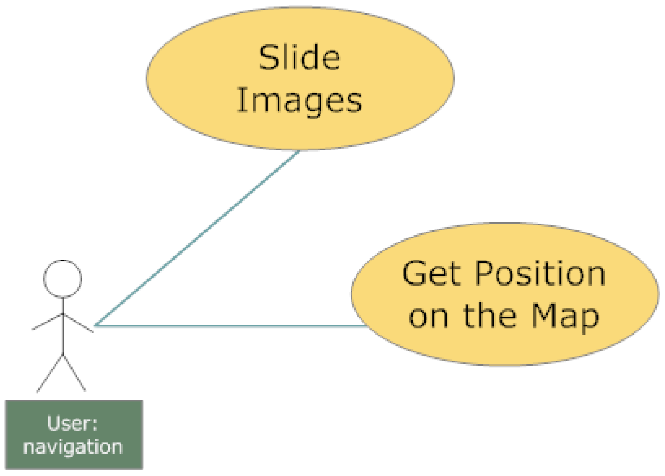


Figure 4.2 Use case one

Figure 4.2 shows the UML diagram for the user type of navigation. In this case, the user mainly navigates the interior of the building rather than performs tasks. The user slides images to obtain the view of building interior. Meanwhile, the position of each image displayed is also indicated on the map. Table 4.1 shows Use case index for the UML diagram in Figure 4.2. Both Use cases are taken place in scope within system boundary, and they have the highest priority.

Table 4.1 Use case index (one)

Use case ID	**Use case name**	**Primary actor**	**Scope**	**Complexity**	**Priority**
1	Slide images	Generic user	In	Med	1
2	Get position on the map	Generic user	In	Med	1

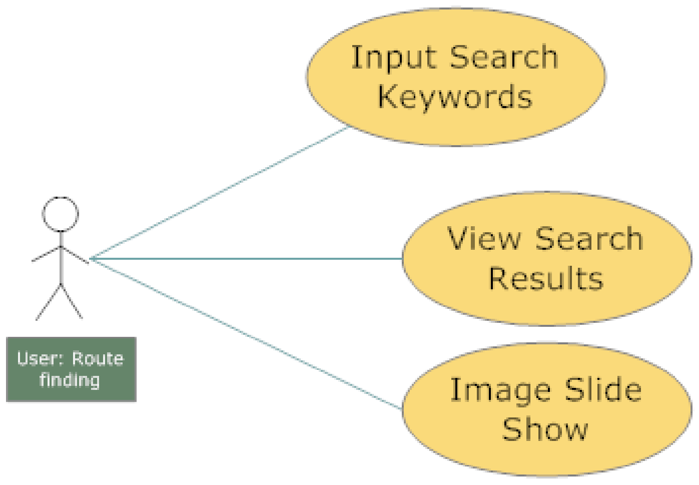


Figure 4.3 Use case two

The UML diagram for the user who performs routing task is shown in Figure 4.3. In order to perform path finding, the user must input search keywords first, and then the system processes the input the return results. Finally the user can invoke an image slide show to view the route. Table 4.2 indicates the Use case index. The Use case 'View Search Result' is more complex than other two and labelled high complexity.

Table 4.2 Use case index (two)

Use case ID	Use case name	Primary actor	Scope	Complexity	Priority
1	Input search keywords	Generic user	In	Med	1
2	View Search Results	Generic user	In	High	1
3	Image slide show	Generic user	In	Med	1

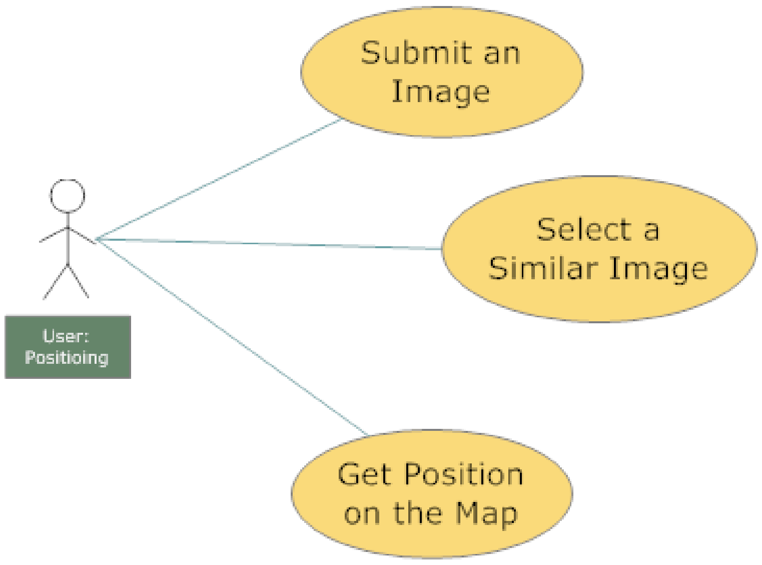


Figure 4.4 Use case three

Finally, the user type of positioning consists of three Use cases. To carry out positioning task, the user submits a query image to the system first. Then a few images are returned as results for user selection. Afterwards, the position of the image is updated on the map. The system must perform image search and matching to enable result selection, which involves the most complex parts in the whole system. Also, update user position requires distance estimation, which is believed to be computational complex. So both Use case 'Select a Similar image' and 'Get position on the map' shown in Table 4.3 are set to the level of high complexity.

Table 4.3 Use case index (three)

Use case ID	**Use case name**	**Primary actor**	**Scope**	**Complexity**	**Priority**
1	Submit an image	Generic user	In	Med	1
2	Select a Similar image	Generic user	In	High	1
3	Get position on the map	Generic user	In	High	1

Functional Requirements We summarize the functional requirements elicited from use case analysis as follows:

Image Submission The function should allow users to upload images which are stored on their mobile phones.

Retrieval The retrieval function should search for the data from the database and return it to the system for calculating user's position or showing images to the user. In this case, it is responsible for retrieving similar images for a given query image. So the input is one image, and the output might be a set of images.

Map creation and mapping This function is for creating a 2D map representation of the floor layout and indicating user’s current location. The system does not require any extra sensors, so all the data is encoded with the images. Some of the data such as image position has to be predefined.

Geo-coding If the mapping function is for showing the users’ location, then the geo-coding function computes the coordinate of a point. In this case, a point is the position of an image. There are two cases, first the position might be hard coded beforehand. Then this is easy to get position information. Second, the position is unknown but related to other predefined location data. Then some algorithms need to be designed to figure out the position. The input to this function should be an image, and the output should be the coordinate of the given image.

Route finding The route finding function estimates the optimal path from the user’s current location to their desired destination. The main data required is corridor topology networks which can be a graph. The search criteria are predefined mainly based on name of the location and annotations. New criteria can possibly be added by the users. The input of this function is the search key word. The output is the computed path or several paths shown on the map.

Direction The direction function is responsible for providing guidance for users to travel from one location to another.

4.3 Non-Functional Requirements

Image format The feature extraction module is based on SIFT. The implementation of SIFT works on greyscale images in .png and .jpg format. So validation of the format of query images should be performed before they are uploaded to the system. Also, colour images should be converted to greyscale images in order to work with SIFT.

Software environment The prototype is mainly developed in .NET framework using C# on Windows 64bit system. For simplicity sake, some complex functions such as high dimensional search and inverse perspective mapping are coded in Matlab. These are converted into .NET com components to be used in .NET framework. One issue is that the integration of .NET and Matlab is not very efficient. We have to implement the .NET wrapper for Matlab, which takes time to initialize. The feature extraction module is programmed in C/C++ in order to work with OpenCV. The program is in ANSI C standard. It is fast and compatible with major operating systems. Such an implementation also ensures the efficiency of feature extraction. OpenCV library is partially used in the feature extraction module to improve its stability.

Code compilation For C# and C/C++ implementations, all programs are supposed to compile in release mode to generate .exe file if applicable. Matlab code doesn't require compilation to run, but it is compiled to .NET com component to be integrated with .NET framework.

Hardware environment and development tools We developed the prototype on a desktop running Windows XP 64bit. The hardware configuration is:

CPU: AMD Phenom 965 3.4GHz

RAM: 4GB

The main development tool is Visual Studio 2010 C#/C++, Matlab R2010b and Notepad++.

Chapter 5

System Design

*In this chapter, we implemented MVC pattern to aid the system design. The system architecture is demonstrated. We also present system GUI and discuss database construction in this chapter***.**

5.1 MVC Model

We basically follow the MVC pattern to design the system.

View It comprises the declaration of the UI and its bind events. After the user interface layout is defined, desired events are bound to the functions of the controller.

Controller This module deals with every event sent by the view and the majority of events sent by the model. For example, once the user clicks on a button, the event is handled by the controller first. Then the controller notifies the model to perform the desired action if necessary.

Model The model sends Ajax calls required by the iNavigation application. It also applies automatic mappings between XML representations and JavaScript objects initialized by the user interface.

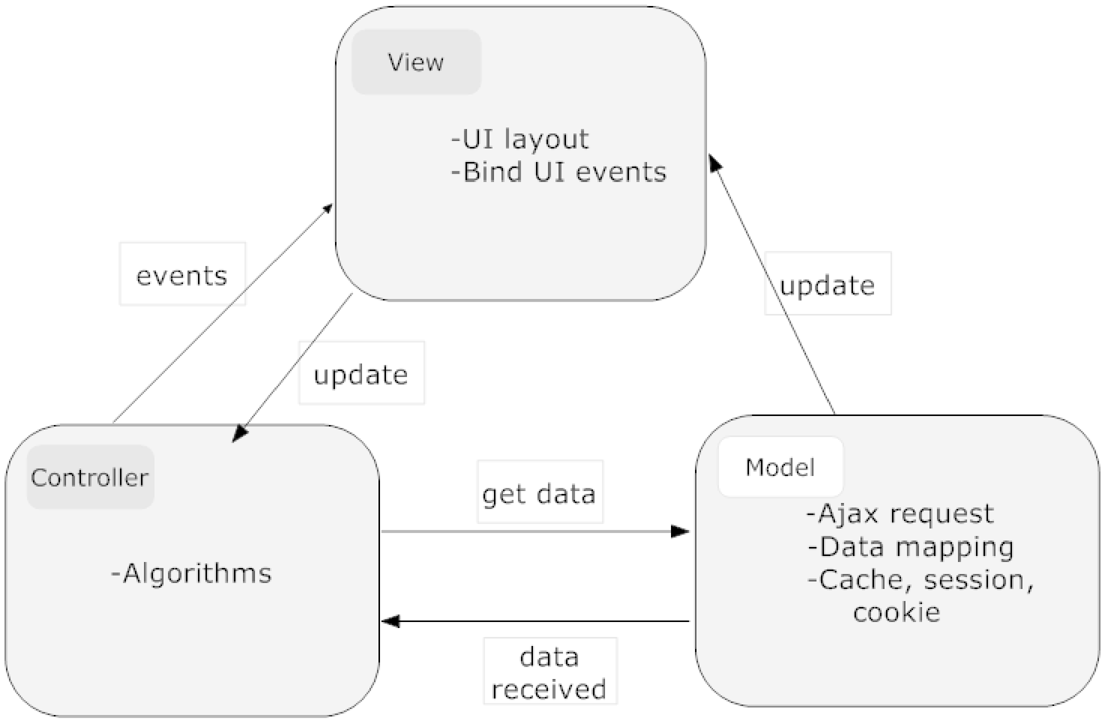


Figure 5.1 MVC model of iNavigation

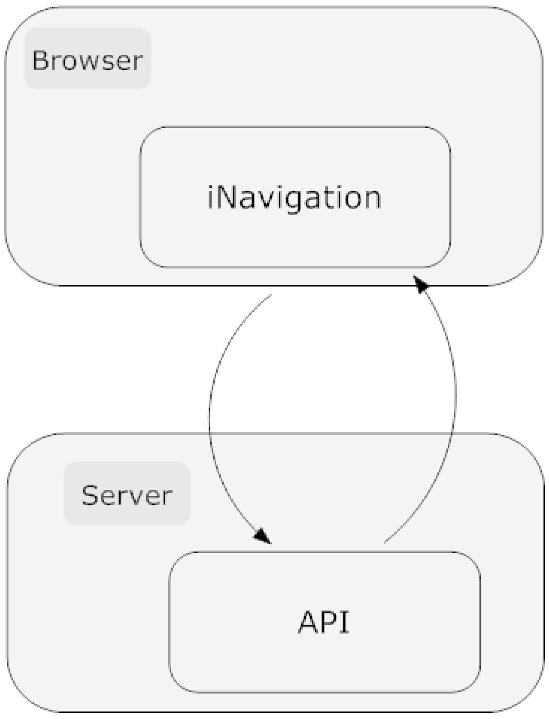


Figure 5.2 iNavigation Overview

Figure 5.2 demonstrates the high-level structure of iNavigation. The system can be divided into client side and server side. Client side is rendered by the browser. Server side is constructed by code behind and APIs of libraries, frameworks. The interaction between the two sides is based on the .NET MVC framework.

5.2 System Architecture

The system design must adhere to our defined objectives. The overall system architecture is shown in Figure 5.3. The system compromises five subsystems such as routing, mapping, positioning, user interface and database. Major components are shown within the subsystem blocks. Figure 5.4 shows the detailed positioning system architecture. When the client uploads an image, the features will be extracted from the image first. Then they are compared to all features stored in the database to find similar images. Afterwards, distance estimation and mapping algorithms determine where to put the image on the 2D map. Finally, the position is updated on the map.

The data consists of three parts, the image dataset, the feature dataset and the .xml database. Image dataset is used for image display and providing reference images. Feature dataset is used to search for similar images. The .xml dataset stores image information and its links to other images. The format is uniform and could be referenced by other subsystems. The relationships between the images and the 2D drawing map are also included in this database. The links are necessary for displaying “You are here”. The core function of this .xml database is to create such a graph of relationships and later we use the graph for the routing search algorithm.

The client side of iNavigation is web based, and is developed using HTML/CSS, Javascript and AJAX techniques, we implement ASP.NET framework for the http server to use. The code is mainly written in C#.

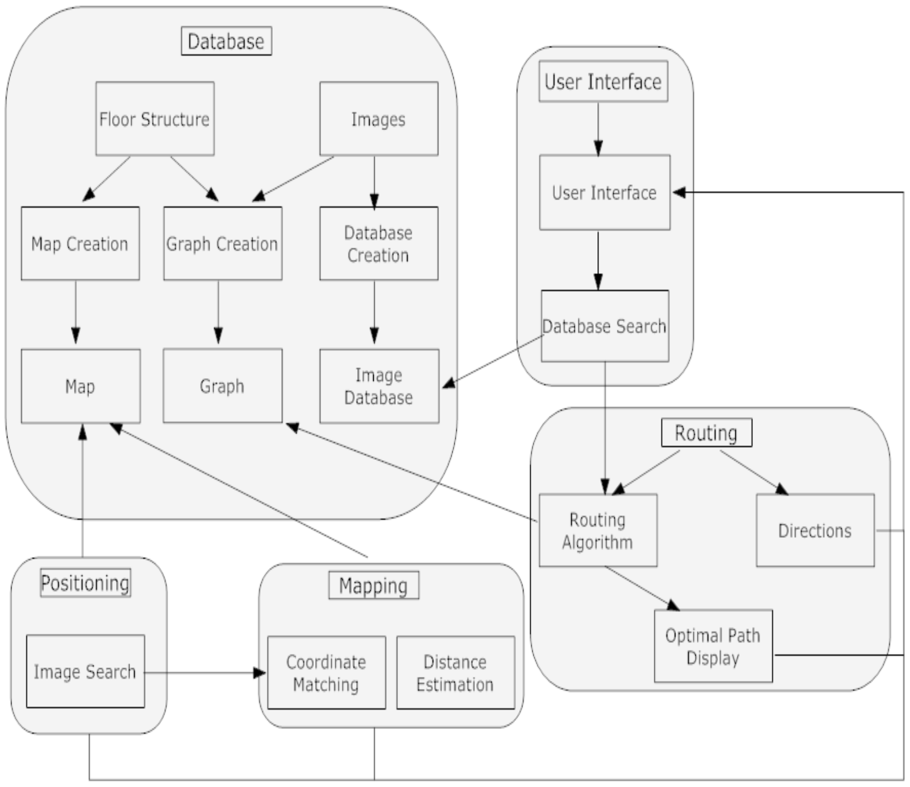


Figure 5.3 Overall system architecture

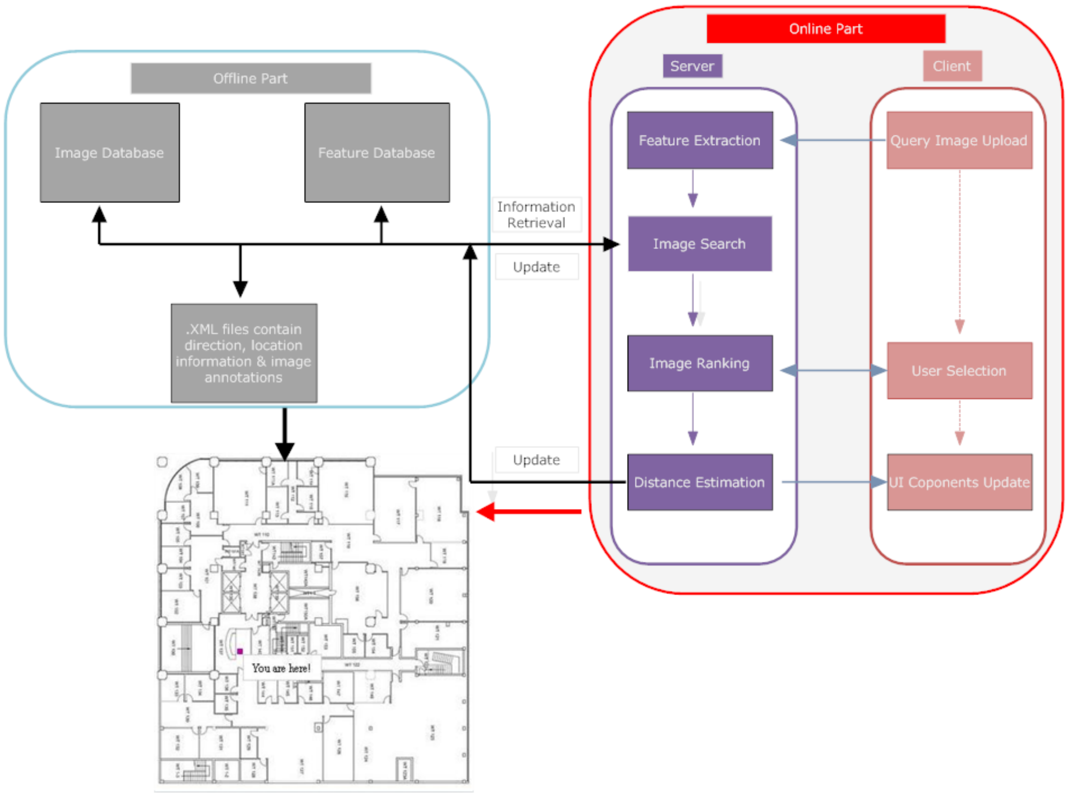


Figure 5.4 Architecture of the positioning module

5.3 GUI Design

The user interface of the iNavigation system is shown in Figure 5.5. Although it runs in a web browser, we optimize the layout for mobile phone users. It is very straightforward to use. The navigation panel is located in the middle. Users can “move” in any direction by clicking the transparent arrows shown on the image. When a mouse cursor is hovering over an arrow, a tooltip will pop up if applicable. The tip indicates direction of the next corresponding image. On top-left, a bird’s-eye view 2D drawing map displays the layout of entire indoor environment. It also uses a red blinking dot to indicate user’s current position on the map. At the bottom, below the image display section, the image gallery section is shown. It allows users to view an arbitrary photo in the database. Users can type query in the search bar, search results are displayed in bottom left block. In the bottom of the image gallery selection, an uploading section is used to upload the query image. Figure 5.6 demonstrates the pop-up dialog for image annotation submission.

5.4 Database Design

Image database is the core part of our system. We don’t directly stream the images into a relational database. Instead, images are stored in the server directory. A unique ID number is assigned to each image as its file name. Metadata of the images are stored in a XML file separately as shown in Figure 5.7.

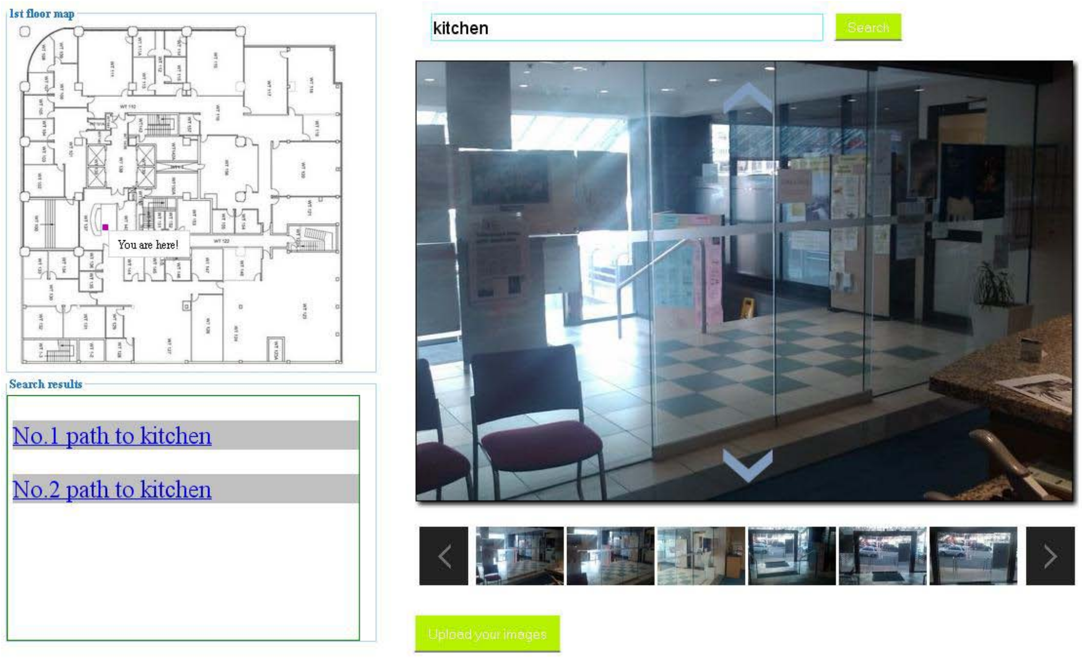


Figure 5.5 System GUI Interface

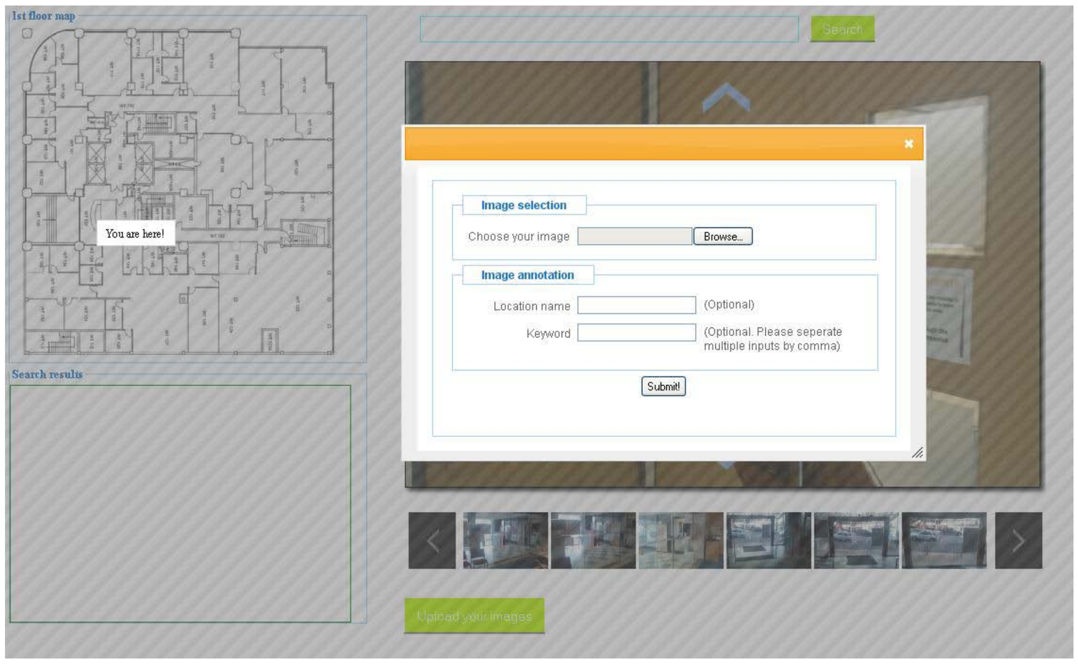
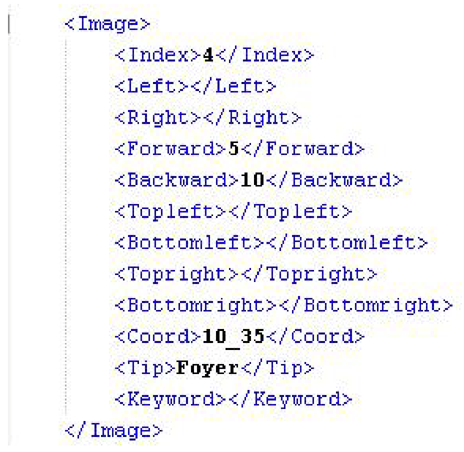


Figure 5.6. Dialog for photograph annotation.



```
<Image>
    <Index>4</Index>
    <Left></Left>
    <Right></Right>
    <Forward>5</Forward>
    <Backward>10</Backward>
    <Topleft></Topleft>
    <Bottomleft></Bottomleft>
    <Topright></Topright>
    <Bottomright></Bottomright>
    <Coord>10_35</Coord>
    <Tip>Foyer</Tip>
    <Keyword></Keyword>
</Image>
```

Figure 5.7 A child element in the XML file used to associate images stored.

The image database is populated with images taken from inside the buildings. Each image has labelled location information associated with that of other images, directions, such as forward/backward, left/right, top-left / bottom-right, top-right / bottom-left, text descriptions of location, and a 120 × 90 JPEG thumbnail image. The thumbnails are stored in a separate folder. We assign the same filename to a full-size image and its thumbnail so as to make sure they are associated correctly. An example of a child node in the XML file used is shown in Figure 5.7. It contains several elements. Indexed elements represent the image filename associated to this piece of code. In this example, the image is named as "4.jpeg", the following eight elements indicate the images associated to it. In this example, if a user move forward, it links to "5.jpeg". The coordinate element indicates the image location on the 2D drawing map and shows a tooltip saying "You are here". Location information for the route search is stored in the keyword element; meanwhile the tip element contains text information for the popup tooltip that is the direction information of those arrows shown on the image.

In terms of image features database, we store them in both text format and .mat format for the Matlab of MathWorks Inc. We generated all text format feature files and one .mat file that contains all features beforehand. The .mat file will be loaded when system starts. While the system is on the fly, the text format features are directly generated by feature extraction programs once a query image is accepted. Then it is loaded into the system and converted to a matrix via .NET MATLAB wrapper. If necessary, the matrix then is merged into the .mat file.

Chapter 6
System Implementation

In this chapter, we follow the MVC design pattern and the system architecture shown in the previous chapter to implement all modules. Some important and novel components are demonstrated and discussed, such as path finding, image feature extraction, image retrieval and distance estimation. A comparative experiment is also carried out to evaluate our image ranking approach as well as the performance of the image retrieval module.

6.1 Photo Capturing and Pre-processing

The first challenge of system development is to populate the database with location-annotated photos. A database has been constructed with photos taken in our office building. The photos were captured strictly following our data collection protocol. We walk through the corridors and make sure every designated waypoint is traversed. All the images are grouped in event order. In this experimental system, 112 images are stored in the image database.

Currently a mobile photo usually has 3 - 5 Megabytes. In order to optimize image size for fast transmission, the system employs colour matrix provided by .NET that can resize the original images to designated dimensions without too much quality loss. Once a photo is uploaded to our server, it is automatically resized to the resolution 720 × 670. Then the file size is shrunk to around 100KB in JPEG format. This ensures that the image will be transmitted faster when the image is displayed to mobile users.

6.2 Instructions of Direction

So far we have the database populated with photos captured. To enable people to navigate inside the building using these reference photos, we have to "link" all photos in correct order. This is achieved via labelling each photo with direction information. For the labels, such as directions to other photos and location information, we add the annotations of each photo manually to the XML file. We use up to eight directions to represent the connectivity between the current image and other images. Eight directions include: front, back, left, right, top left, top right, bottom left and bottom right. Note that the latter four directions are sometimes used to indicate a shortcut to a particular location. It allows users to “get to” that location directly without sliding lots of images. However, it might not actually connect to the current location in real environment. Some detail of .xml file is shown in Figure 5.7.

We also add eight clickable arrows for the eight directions as visual aids on the navigation panel. If the current image has connections with others, the transparent arrows will pop up on the panel. Once the user clicks the arrow, the panel leads the user to the next associated image. We are confident that the system is very easy to use, similar to any ordinary image slideshow.

6.3 Image Thumbnails Gallery

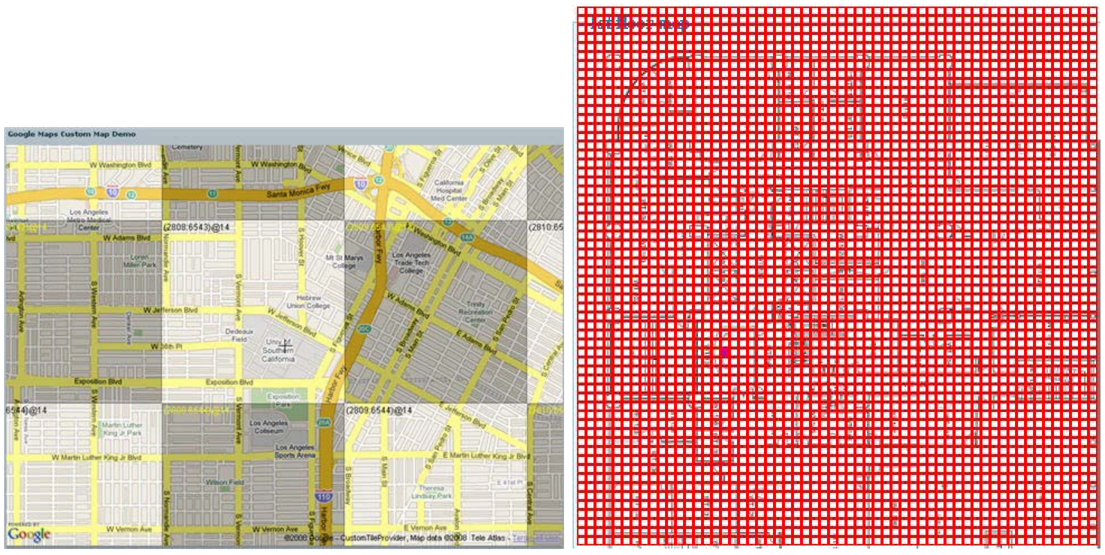
People sometime may want to view any images at will, other than using navigation panel to take a virtual tour in the building. The gallery feature is to aid users to quickly jump to any arbitrary image stored in the database. It accommodates all thumbnails of the full size photos in navigation panel. New thumbnails generated from successfully processed images uploaded by users will be appended to the existing thumbnails. Once a client selects a thumbnail in the gallery, the image shown in the navigation panel will be changed to the corresponding larger image. Using the two arrows at the ends of the panel, we can select any thumbnails quickly.

6.4 Map Creation

In order to present an image position, we have to pin the location on a 2D drawing map so as to indicate “where we are” [17]. In Chapter 2, we discussed some map creation approaches using CAD files which are able to present rich information. In this case, we are able to find a simple 2D map drawn by CAD to represent the floor layout of our building. Although the map’s resolution is low, the floor components such as rooms, corridors, doors are clearly indicated on it. Because, all position information has to be added to .xml files manually beforehand. We construct a layer made up of grids at 4×4 output resolution on client side. It is similar to the Google map tile. Higher resolution of the layer means more grids created and slower performance. So we alter the resolution according to the dimensions of the map. The 2D drawing map used in the system is at the resolution of 360 × 360. We could lower the resolution if we have a bigger map. Once the grids are created, they can be used simply as the coordinates of the map. Then we can assign a coordinate to the image and store the coordinate in XML files. Our current iNavigation system can provide location guidance if a user starts the system as he gets into the building and uses it while walking around. Figure 6.1 demonstrates the layer with map grids.

6.5 Routing Algorithm Implementation

In our iNavigation system, all the images are linked together according to the directions. Then we create a graph consisting of all nodes representing these images. Each node comprises the information of an index and annotations of the image. Figure 6.2 shows a part of the graph representing image relationships. It is weighted and directed (we assume all costs are 1). Some directions lead to shortcuts, which might cause inconsistency to the route. So we only take four directions (front, back, left and right) into account. This graph can be used for more than just localization; it enables the system to figure out the paths from current position to a destination. This makes image based approach a complete solution of indoor navigation.



(a) (b)

Figure 6.1 Google map tile and our map grid. (a) shows Google map tile, (b) is our map grid.

The first step in path finding is to parse the query and search the database to find images with the same key words. A query string is broken down into parts by space. Then each part is compared to key words in image annotations to find a match. Because a few images may have the same key words, multiple search results may be returned for one search query.

Once the current image and the destination image in the graph are determined, a path through the graph that links them needs to be found. There might be hundreds of paths between the two nodes, but we only want the one with least cost which means fewest images in this case. This can be stated as a single-source shortest path problem [83]. Figure 6.2 displays a graph representation of images captured. Note that nodes linked by a red line indicating images taken at roughly the same position but opposite directions (e.g. front and back).

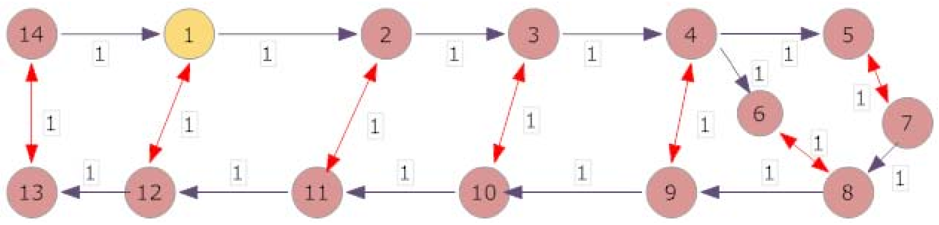


Figure 6.2 Graph represents current photo relationships.

In the proposed system, we implement Dijkstra’s algorithm [82] to solve this problem. This is one of the most popular routing algorithms. It works by evaluating a single node at a time, starting from the initial node. At each step in the loop, the algorithm finds a node with lowest cost between current node and every other adjacent node. Then the current node is marked as visited and will never be visited again, while the node with lowest cost is marked as optimized and becomes the “current node” for the next loop. Once the destination node is marked as visited, the algorithm finishes. It had been mathematically proven that Dijkstra’s algorithm is guaranteed to find the shortest path [83].

6.6 Slideshow

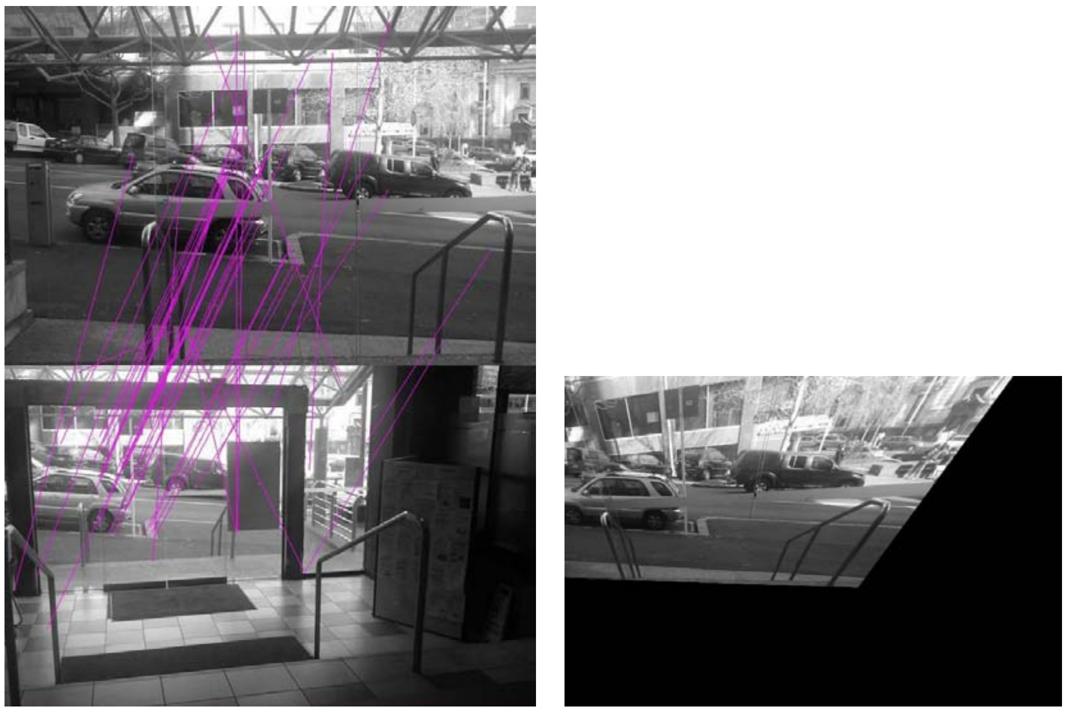
Once the optimal paths are found, they have to be shown to users in a suitable way. Usually, a line is drawn on a 2D map to indicate the accessible path. In our system, the feature slideshow is used to illustrate the optimal route. It does a slideshow that displays all images in the path. The position of all images displayed in the slideshow is also marked as a flickering red dot on a 2D drawing map while the slideshow is going on. The main motivation of creating this feature is to provide a novel user experience. This feature can be invoked by clicking the search result shown in the bottom left block of the user interface.

6.7 Image Feature Extraction

We need features to be extracted from a query image before proceeding to image matching. Our feature extraction implements SIFT algorithm, which consists of the DoG feature detector and SIFT feature descriptor. The rationale behind the selection boils down to two facts that we learned in the literature. First, the SIFT algorithm gives consistently good performance in various experiments conducted over a wide range of dataset. Since we do not set many restrictions on how users take photos, we assume the query images to be dynamic and very unpredictable. So what we need is pretty much a “jack of all trades”. We have gone through a number of local feature detectors and descriptors as described in Chapter 2. Evaluations have been done to assess the performance of various combinations of them. Also different dataset has been used in experiments. The test images vary in rotation angle, blurring, illumination, viewpoint and scale. Basically, there is no such method that works perfectly in all evaluations. However, the original SIFT stands out among the majority of other algorithms and delivers robust and decent results over different datasets. In a particular experiment, it might not be the top performer, say top 3. However, if we take the overall performance into account, it is definitely the right choice. Another reason is that SIFT is very popular and has been successfully implemented in some applications [59, 65, 69, 91, 105]. That means we could have access to resources such as documentation and implementation results for this algorithm.

However, original SIFT implementation has distinct drawbacks. First, it is only partial Affine invariant, its performance drops drastically when the viewpoint is larger than 40 degrees. Second, its speed is slower than most scale invariant algorithms, but still faster than major Affine invariant algorithms. At this stage, SIFT fulfills our requirements, and possibly improvements will be discussed in Chapter 8.

In our system, the implementation of SIFT is basically based on Rob Hess's code [91]. We replaced some routines with OpenCV functions and also changed the output file format so as to cater for the data format requirements of our system. Some results of the module are shown below:



(a) (b)

Figure 6.3 Image matching using SIFT algorithm. (a) shows matched SIFT features with outliers, (b) indicates the homography of the two images.

6.8 Image Matching

The capability of an image matching approach decides the accuracy of image search. In a simple pair-wise image correspondence problem, the image matching is carried out after all features are detected and extracted. It is usually performed by comparing the correct ratio, as shown in equation 6.1 and 6.2.

$$\textit{Correct Ratio1} = \frac{Number\ of\ matched\ features}{Total\ features\ detected} \tag{6.1}$$

or

$$\textit{Correct Ratio2} = \frac{Number\ of\ correct\ matches}{Number\ of\ matches} \tag{6.2}$$

High ratio indicates two images are very likely to match. However, our system does not perform pair-wise search (linear search), as it is slow. Instead, we use an algorithm that searches for approximate nearest neighbors. This is discussed in the next section.

Algorithms like ORSA [89] and RANSAC [76] are able to give quite high accuracy on deciding if two images match. But they are basically designed for pair-wise matching, not suitable in our case. Hence we use a method to rank the search results rather than find the best match directly, and return the most likely images to the user. Then it is the user's call to pick up the most similar image.

Before we discuss the ranking approach, we go through what our system returns after performing image search function.

- The system performs approximate nearest neighbor search first.
- Then the algorithm returns indices of nearest neighbors.
- The system maps vector indices to image indices.
- Finally it returns a list of images indices.

Our ranking approach works as follows:

- First we rank the images by number of matched features recorded and select top *N* images. *N* can be controlled by altering the value in the code.
- Then we calculate Mean Squared Error (MSE) [100] of distance metric between the query image features and features of every candidate image using equation 6.3.

$$\text{MSE(x, y)} = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{6.3}$$

- An image will be downgraded if its MSE is larger than the other while its number of matched features is not significantly larger than the other, say at least 20% larger.

Performance of this ranking approach will be discussed in the next section.

6.9 Image Search

In a search problem, there is no doubt that we can find the best result if we do an exhaustive linear search. However, in cases where the size of dataset could be large, and the data consists of high-dimensional data, such linear search would be extremely slow. For example, in our research, a collection of 112 images is used as the dataset. We extract features from every image using SIFT algorithm. The initial dataset we produced is a 128×54075 matrix, which produces a 5MB .mat file, and it could get bigger once new features extracted from uploaded images are added. If we look for an image in the dataset which is similar to the query image using a linear search, it would probably take 6 minutes to complete this task, which is unacceptable.

In Chapter 2, we went through some elegant algorithms which are designed to perform high-dimensional search. Here, we implement two algorithms to achieve the image matching function: the Local Sensitive Hashing (LSH) [65, 66, 69] and the Approximate Nearest-Neighbor algorithm (ANN) [101, 102]. Both of them have been successfully implemented in several multimedia applications. However, we have to decide which one works better on our dataset.

Experiment setup This comparative experiment aims at answering three questions listed as follows. It is also conducted as module testing to check if the algorithms as well as the modules are working properly.

- Which search algorithm is suitable for our dataset?
- The performance of the matching algorithm proposed.
- How many results (images) should be enough for the system to return to users for selection?

Both search algorithms, LSH and ANN are initially implemented in MATLAB R2010b and then converted to .NET com component. We developed our own version of LSH. It implements L_2 distance metric. Such an implementation is also called E2LSH [103]. We use 24-bit keys, and the bucket size is set to 10. The ANN implementation is from Statistical Learning Toolbox [104] which is available for download on MATLAB File Exchange. The author stated that it is free to use for research and educational purpose. The distance metric used in ANN was also set to L_2 distance. So both distance metrics are the same.

In the experiment, all aspects of two trials are the same except for the different search algorithms. First of all, the parameter N in image ranking approach was set to 12. Note that sometimes the search algorithm may not be able to return this number of images. In such a case, we just work with the number of images returned rather than N. We also made sure that the two trials perform the same image search task on our dataset running on the same desktop. The test dataset is a collection of query images consisting of 50 images. 40 of these are from our indoor dataset. For each test image, there is a very similar image that can be found in our dataset. We assume that if the best match can be displayed to the user, then he/she will be able to identify the match in a list of returned images and select the best match based on their own justification. In this experiment, the selection of the best matched image is reached through our observation rather than using any matching algorithms, because algorithms may fail to identify two matched images even if they are actually a match. The 50 pairs of images are the ground truth of our experiment.

Evaluation criteria There are two control variables in this experiment:

1) *i*-NN search. The values of *i* range from 3 to 10.
2) Number of result images. The value is from 1 to 10.

We implemented two criteria in our evaluation. The first criterion is specifically designed for evaluating the system performance. Basically, we measure the retrieval accuracy under every search conditions (*i*-NN search and number of result images):

$$R = \frac{True\ cases}{Total\ cases} \times 100\% \tag{6.4}$$

If a case is true, it means that the most similar image can be found in the results, or else the case is false. For example, we perform the search using 6-NN and output top 10 ranking images, then we test one of our 50 query images. If we cannot find the most similar image in the output, then we reckon this is one false case.

The second criterion is Recall-Precision ratio. It is only used in a particular search condition, say, 10-NN and 6 results. The ratio is defined as Table 6.1.

Table 6.1 Confusion matrix

	Predicated Positive	**Predicted Negative**
Positive Cases	True Positive	False Negative
Negative Cases	False Positive	True Negative

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative} = \frac{Number\ of\ similar\ images\ found}{Total\ number\ of\ similar\ images\ in\ dataset} \quad (6.5)$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} = \frac{Number\ of\ similar\ images\ found}{Number\ of\ results} \quad (6.6)$$

Note that total number of similar images for a given query image is determined by observation.

Experiment procedure The experiment is carried out as follows:

First, the two trials perform 3-NN search on our indoor set where the algorithm returns 3 nearest neighbors without the query image itself. Every image in the test dataset will be submitted to the system. The system will return the top 3 ranked images for every query image. We observe the result and check if the most corresponding image is in the 3 returned images. If so, then we reckon the task is successfully performed, otherwise the task is failed. Time consumption of the task performance is also recorded by setting a timer in the code.

Then, we keep at performing 3-NN search, but this time the test return top 4 ranked images. We also record the performance for every query image. So we continue the experiment by increasing the number of returned images until it reaches 11. (We do not perform the 11th experiment). However, if we perform 3-NN and display 10 images, it is more likely that there are 10 images that can be displayed. In this case, we only consider the results in the displayed images.

After the 3-NN search is done, two systems perform 4-NN search and rest of experiment is exactly the same as before. We continue running this experiment by increasing the desired number of nearest neighbors until it reaches 11-NN search. The overall performance result is shown in Table 6.2 and Table 6.3.

Table 6.2 ANN search performance

No. of img / ***i*-NN**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
3-NN	76%	84%	84%	88%	88%	88%	88%	88%	88%	88%
4-NN	76%	84%	86%	90%	90%	90%	90%	90%	90%	90%
5-NN	78%	84%	86%	92%	92%	92%	94%	94%	94%	94%
6-NN	82%	86%	92%	92%	94%	94%	96%	98%	100%	100%
7-NN	82%	86%	94%	94%	94%	96%	96%	98%	100%	100%
8-NN	82%	86%	94%	94%	94%	96%	96%	100%	100%	100%
9-NN	82%	86%	94%	94%	94%	96%	98%	100%	100%	100%
10-NN	82%	86%	94%	94%	94%	96%	98%	100%	100%	100%

Table 6.3 LSH search performance

No. of img / ***i*-NN**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
3-NN	70%	74%	78%	82%	82%	82%	82%	82%	82%	82%
4-NN	70%	76%	80%	86%	86%	86%	86%	86%	86%	86%
5-NN	74%	78%	84%	86%	90%	92%	92%	92%	92%	92%
6-NN	76%	78%	84%	86%	92%	92%	94%	96%	98%	100%
7-NN	80%	84%	90%	92%	94%	94%	96%	98%	100%	100%
8-NN	80%	86%	90%	94%	94%	96%	96%	100%	100%	100%
9-NN	82%	86%	92%	94%	94%	96%	98%	100%	100%	100%
10-NN	82%	86%	92%	94%	94%	96%	98%	100%	100%	100%

Results Discussion The results were interesting:

- Both of the trials carried out the approximate nearest neighbor search, there is no guarantee that they can find the nearest neighbor in every case.
- Results of 1-NN and 2-NN search were very poor, so they are not shown here. The results themselves are not necessary to be discussed. However, considering the behavior of 1-NN, 2-NN search and the sudden improvement on 3-NN search, it implies that our ranking algorithm needs more samples to work on.
- The performance of proposed ranking approach can be assessed by the results with only one return image. Under this circumstance, it is like a pair-wise matching that gives only one result. We conclude that if we can provide more samples to the algorithm, like getting more nearest neighbors, the matching accuracy may be decent.
- Fewer nearest neighbors result in fewer candidate images. 3-NN and 4-NN search are very likely not be able to produce 10 images for ranking. So the accuracy in the later experiments almost stays the same.
- Basically the ANN search outperforms LSH by a little in every experiment with 5 ranked images or less. The two get more close results when more ranked images are given. We believe this is due to the parameters of both algorithms not being tuned. What we used are all "default settings". However, such results also reflect the conclusion shown in Lowe's work. His experiment on 1 million SIFT features also

indicated that LSH performs a little worse than ANN on the basis of neighbor correct rate.

- LSH is a lot slower than ANN. The time we recorded indicates that the average execution time for ANN to perform a 6-NN with 6 ranked images task is around 12 seconds. However, it took an average 22 seconds for LSH to get it done. But we believe this to be a result of my poor implementation rather than the design of the algorithm. My LSH implementation can only search for a single query vector at a time, so the outer loop that iterates the query matrix was done in a .NET wrapper for MATLAB, which is reportedly not quite efficient. However, ANN implementation takes the whole query matrix as the input. Thus there is no need to use a great deal of .NET wrapper in the code.
- We notice that some query images seem to be very hard to find its match for one of the algorithm but quite easy for another. This is probably due to the algorithm parameters setting.
- It is intuitive that if more images are ranked and returned to the user, it is more likely that the user can find the most similar image. However, too much feedback can cause usability issues. First, if there is no image which is more significantly similar to the query image than others, then the user may take time to select a result. Note that in this experiment, we selected the ground truth image pair by observation beforehand, thus we could quickly identify if there is the best match, even if there are 10 images

displayed. However, users may not be able to do this. Secondly, it may not be suitable to display many images at the same time on a smart phone, primarily due to the limited screen size. We could consider resizing images to smaller sized ones, but it still would be difficult for users to view images. With ANN implementation, the system gave good results when the number of images reaches 5, which is a good balance between performance and usability. In the final system, we implemented it to show only 4 images for this reason.

- Note that the results shown in the table don’t mean that all similar images are found for a given query image. It shows only the accuracy of finding the best matching image, which fulfills our task in practice. Figure 6.4 demonstrates the recall-precision curve of retrieving all similar images using ANN search. In this case, the number of returned images is set to 10, and then we use 6-NN search, 8-NN search and 10-NN search to carry out the experiment. In term of precision, the figure shows that results yielded by 6-NN, 8-NN and 10-NN are very similar. The 10-NN search slightly outperforms the other two, while 6-NN search gave the worst result, which is predictable according to the results shown in Table 6.2. The drop in the figure is basically due to the fact that input images contain large viewpoint changes for which SIFT cannot extract enough meaningful features. This influences the image retrieval accuracy.

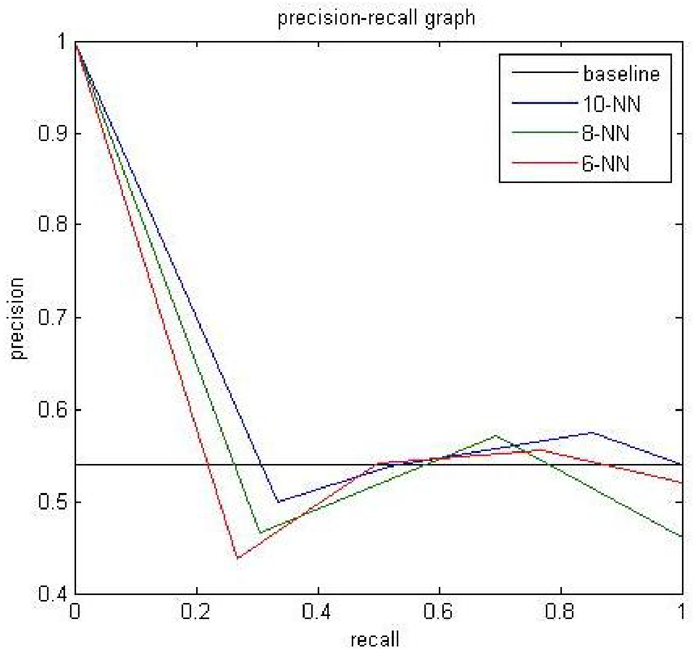
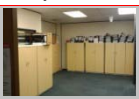
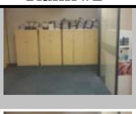
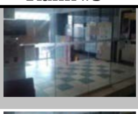
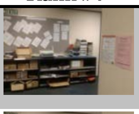
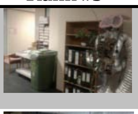
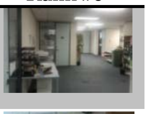
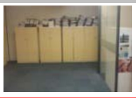
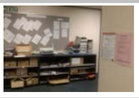
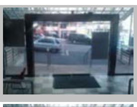
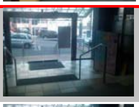
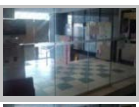
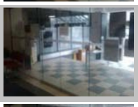
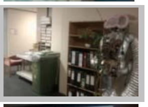
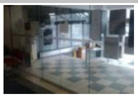
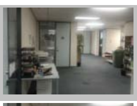
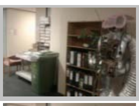
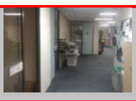
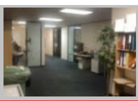
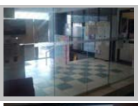
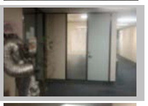
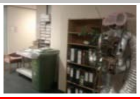
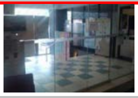
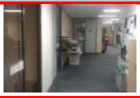
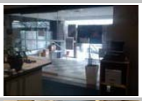
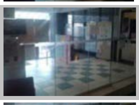
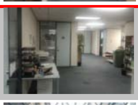
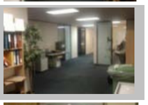
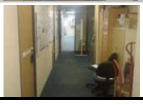
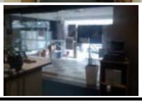
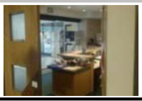


Figure 6.4 Recall-Precision curves for ANN search with 10 returned images

Table 6.4 Qualitative analysis of image matching results.

	Query Image	**Rank #1**	**Rank #2**	**Rank #3**	**Rank #4**	**Rank #5**	**Rank #6**
ANN							
LSH							
ANN							
LSH							
ANN							
LSH							
ANN							
LSH							

According to the results above, we applied ANN search, which gives better results among all evaluations to the image searching module. The parameters are set to 10-NN search by default. We also decided to set the number of output images to four for user selection. Table 6.4 lists the result produced by the two search algorithms under the condition of 6-NN search and 6 returned images. Note that images with red border are the most similar images in the database.

6.10 Distance Estimation from a Single Image

In order to estimate the distance between the object in the image and the image plane, we implemented inverse perspective mapping technique [84] to roughly figure out the distance. With the aid of this technique, we are able to do distance estimation from a single image by mapping it to a top-down view image. However, this technique is better to be used in the circumstance where the camera is fixed and calibrated.

Figure 6.5 presents the relationship between the world coordinate system and the camera coordinate system. The mapping of the ground (x_w, y_w, z_w) to the camera plane (u, v) has to be established [85]. Equation 6.7 indicates the mapping.

$$(u, v, 1)^T = KTR(x, y, z)^T \tag{6.7}$$

where R presents the rotation matrix [85]:

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.8}$$

T presents the translation matrix [85]:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\dfrac{h}{\sin\theta} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.9}$$

where *h* presents the height above ground to the camera. *K* presents the camera parameter matrix [85]:

$$K = \begin{pmatrix} f \times ku & s & u_0 & 0 \\ 0 & f \times kv & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6.10}$$

where *f* presents camera focal length, $ku \times kv$ is the aspect ratio and the *s* is skew factor which is $tan\theta$ in this case.

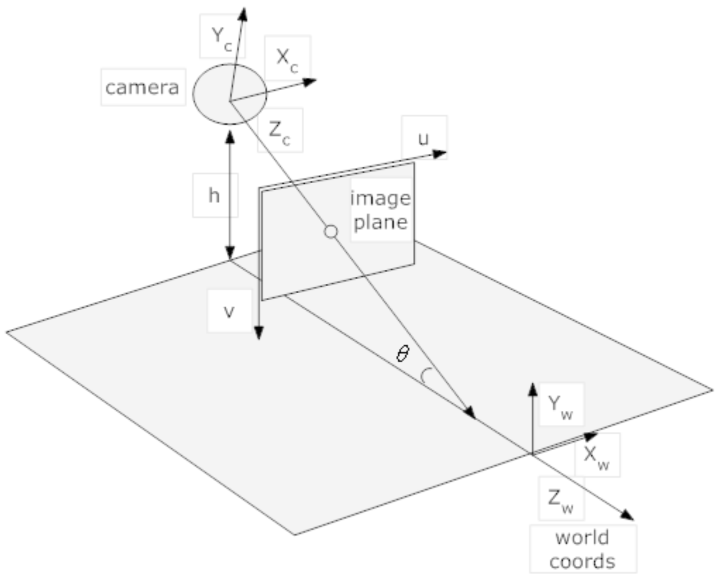


Figure 6.5 Relationship between image coordinate and world coordinate

The next step is to determine the camera parameters. Here, the focal length f is around 6 millimeters (We checked out the specification of the smart phone built-in camera). $ku \times kv$ is set to 640 × 480. The h is then set to 1.7 meters, which is the restriction we followed while we were photographing. However, we do not know the exact value of tilt angle θ, although we basically kept the camera parallel to the ground. So we can use distance value collected to estimate the θ. We only change θ and keep other parameters unchanged to yield different results. Then we observe the results and compare them to actual values. An example is shown in Figure 6.6. The actual distance to the door is around 7 meters; the estimated distance is 7.5 meters.

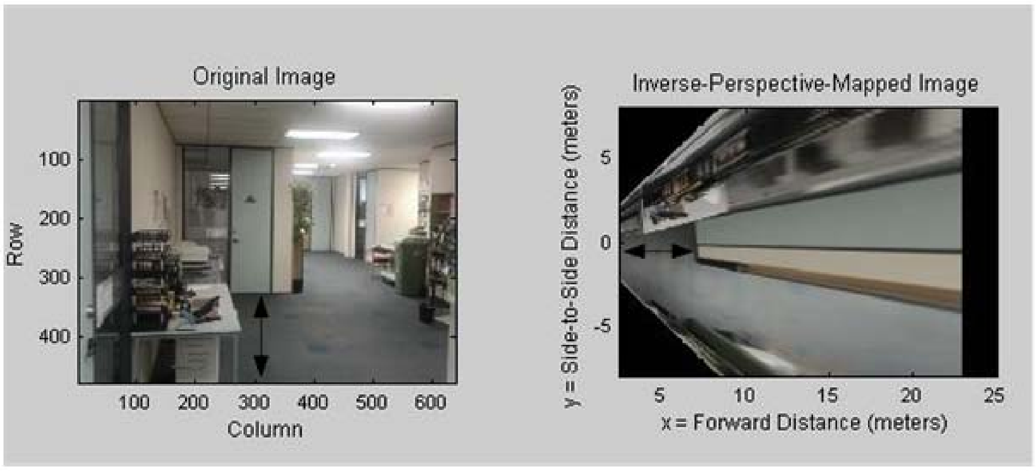


Figure 6.6 Distance measurement result. The black arrows are added manually to indicate the distance measured in the image.

A value of θ should be selected when all results are close to the actual values (error $\varepsilon < 1.5$ meters). We initially set the θ to –3 degrees, then we test all 20 images with actual distance values to determine the angle. We tried to alter the value of angle (either decreasing or increasing it) to get a better result. Finally we found the value –0.14 degrees yields the best results.

Similar to other algorithm implementations, we coded the distance estimation module in Matlab and then converted it to a .NET com component. Note that this implementation only works in this case. Alteration of parameters may cause unexpected results.

To calculate distance, we simply reckon center of the image is the focal point, which is also the area in which the user is interested. Note that the focal point is not always located in the center, so it is an assumption. We first retrieved the coordinate of the pixel in the center, and then we performed the image segmentation to the IPM mapped image shown in Figure 6.7 and labeled all segments. The segmentation was carried out by using a color-based segmentation function built in Matlab Image Processing Toolbox. Afterwards, we determined which labeled segment contains the selected pixel by checking the coordinates. Finally, we picked the coordinate of the leftmost pixel in that segment. By using the IPM mapping equation 6.7, we can estimate the desired distance.

We acquired more actual distance values of images. A simple test shows that the average absolute deviation is 2.2 meters. Because the distance estimation module is used only to compare distance measured from two images (e.g. determine which one is further), such accuracy is acceptable.

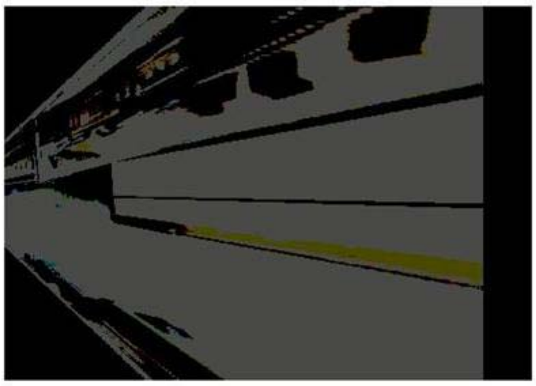


Figure 6.7 Image segmentation result

6.11 Map Update

Once a similar reference image in the database has been confirmed, the position of the query image (denoted as I_q) on the 2D map will be figured out on the basis of the position of the selected reference image. To achieve this task, we localize the query image between two reference images. Because one reference image is already confirmed, and we denote it as I_{r1}, another reference image should be the previous image of I_{r1}, and we denote it as I_{r2}. Figure 6.8 shows the physical relationship between I_{r1}, I_{r2} and I_q.

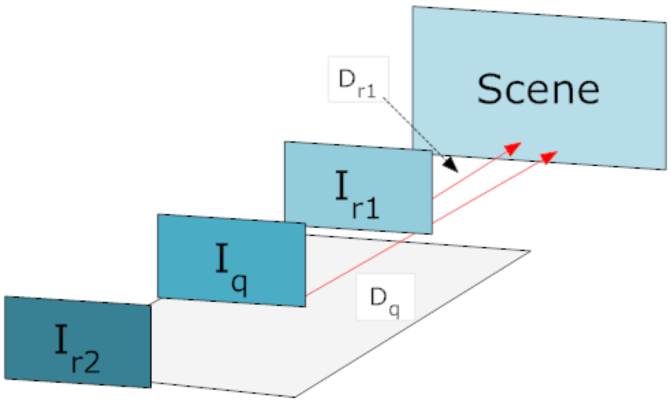


Figure 6.8 Positions of images on the map.

Two cases needed to be discussed. First, there is no other image between the two waypoints. This means the query image is the first image to be added between the two reference images. In this case, we simply add the query image in the middle of the two reference images without considering its actual position. This won't affect the accuracy much, because the maximum interval of two waypoints is around 10 meters. Second, if a query image is not the first image to be added between the two waypoints, then we have to figure out the sequence of the images by comparing the estimated distance values. After the sequence is determined, the new query image is also added in the middle of other two images.

If there are not enough grids or the query image and the reference image are too close ($(D_q - D_{r1}) < 2$ meters), then the query image is dumped. We also dump query images which are too similar to the reference images. For example, the Mean Square Error is 0.

In terms of updating directions, we basically set the directions of an accepted query image the same as its closest image, either I_{r1} or I_{r2} in this case. Programmatically, it is done by updating the XML files accordingly.

Chapter 7

System Testing and Evaluations

Some of the module testing and evaluations have been evolved in the previous chapter of system implementation. For instance, we compared the ANN and LSH search algorithm performance over our own dataset. The image ranking approach also was examined along with the comparison. In this chapter we discuss the testing of SIFT feature extraction module. The main idea behind it is to confirm that the implementation of SIFT works properly on our dataset. We also demonstrate the system level performance. System response time, accuracy as well as limitations are taken into account.

7.1. Feature Extraction Testing

From a software development point of view, every module has to be fully tested in order to make sure it performs properly. We have already demonstrated the testing and evaluations of image searching and distance calculation module. In this section, we present a simple comparative evaluation that reveals the performance of our feature extraction module. We then compare the results to the intrinsic characteristics of SIFT shown in other studies [42, 64, 62, 75] in order to justify the performance of feature extraction module.

Evaluation Criteria A number of evaluation criteria have been used as evaluation metrics in recent years such as recall and precision, average precision, repeatability rate and the ROC curve. With the available data in hand, we implement repeatability rate as our evaluation criteria. Here, we referred to Mikolajczyk's work [49], and define repeatability as equation 7.1.

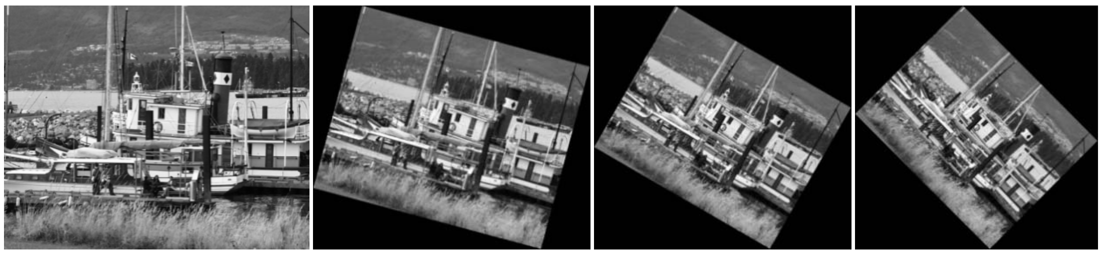
$$\text{Repeatability} = \frac{C(I_1,\ I_2)}{mean(N_1, N_2)} \tag{7.1}$$

where $C\ (I_1, I_2)$ means the number of correspondent feature pairs and (N_1, N_2) denotes number of features detected (matched and unmatched) in two images respectively. If a large number of correspondent points are detected, we implement RANSAC to filter out the outliers to get correct correspondences. Otherwise, we get the correct correspondences by observation. The correct correspondences $C(I_1,\ I_2)$ are the ground truth data.

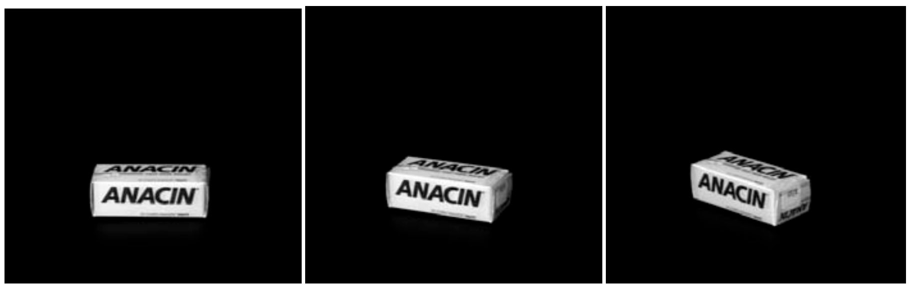
Other than repeatability, we also use the ratio of correct matches in respect to total matches as a criterion.

$$Correct\ Ratio = \frac{C(I_1,\ I_2)}{M(I_1,\ I_2)} \tag{7.2}$$

where $M(I_1,\ I_2)$ denotes number of total matches (true positive and false positive).

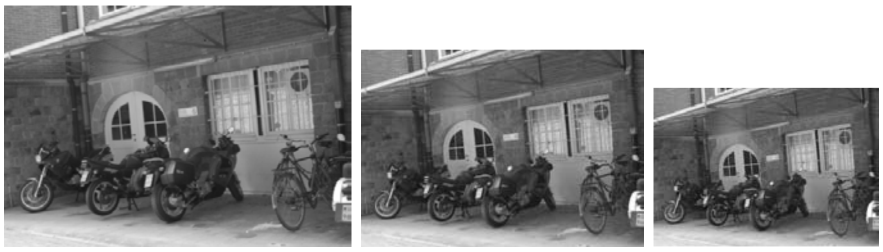


(a)

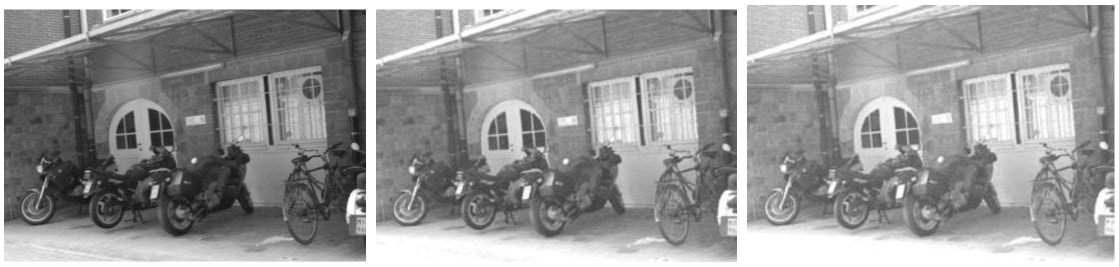


(b)

(c)



(d)



(e)

Figure 7.1. Some images used in the test. (a) Rotation, (b) Viewpoint change, (c) Varying blur, (d) Varying scale, (e) Varying illumination.

Evaluation Set We selected images from both VGG Affine invariant dataset used in [57] and Columbia University Image Library [90] as our test data. They are both collected or created for evaluating local feature detectors and descriptors. All selected images are converted into greyscale image and specifically altered to images with desired properties in order to fulfill our requirements. Five categories of images are created. Each category consists of one reference image and several test images.

1. *Rotation.* One image was used as the reference image. Then the Matlab function *imrotate* was applied to the reference image to produce desired rotated images. The angles range from 0 – 165 degrees in steps of 15 degrees.
2. *Affine transformation* (*viewpoint*). This transformation is quite difficult to quantify. One sequence of images was directly used as the test image. Viewpoint change angle values from 0 – 75 degrees in steps of 5 degrees.

3. *Blur*. The blurring of images was obtained by applying the Gaussian filter to the reference image. We basically change the value of sigma in MATLAB function *fspecial* to produce different Gaussian filter effect. The sigma values range from 0.5 pixels to 5 pixels in steps of 0.5 pixel.
4. *Scale*. We simply resized the reference image to desired sizes. The scale value in the MATLAB function *imresize* was altered from 0.8 to 0.2 in steps of -0.1.
5. *Illumination*. This set of test images was also generated using Matlab. The reference image is from VGG database [57]. We load the image in Matlab and alter the intensity values to produce different illumination. The changes range from 35 to 95 in steps of 10.

Some images used in the evaluation are shown in Figure 7.1.

Evaluation Results we show the results as below:

Rotation In figure 7.2 (a), we can see that the number of correct matches is quite stable in the entire experiment. In figure 7.2 (b), the repeatability shows an increase when angle equals 90 degrees. The overall performance is quite good and doesn't fluctuate much. Curves in Figure 7.2 absolutely reflect that SIFT is invariant to rotation.

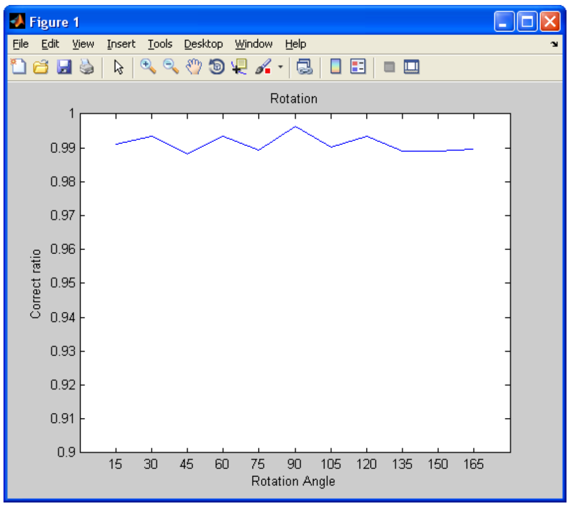
Viewpoint The correct ratio in figure 7.3 (a) shows a U-curve when the angle is less than 25 degrees, but it drops drastically after 25 degree. After 55 degree, the ratio is below 0.4, which is quite low. As shown in the figure 7.3 (b), the repeatability ratio drops along with the increment of view angle. The repeatability is OK until the angle reaches 40 degrees, and it cannot find any correct matches when the angle is greater than 60 degrees. Basically, both curves in Figure 7.3 reflect the major drawback that SIFT is not fully Affine invariant.

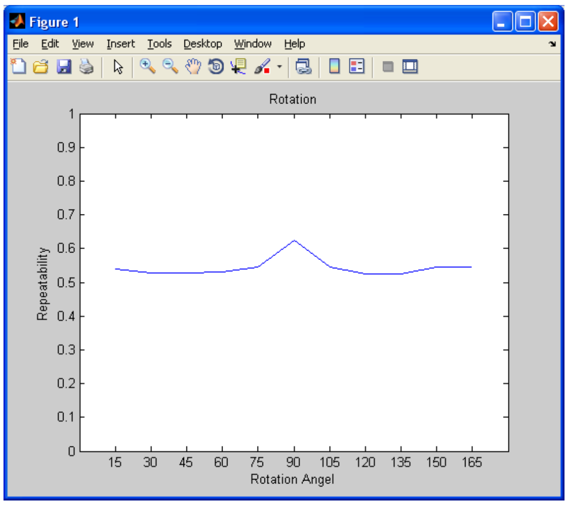
Blur Figure 7.4 indicates the influence of the Gaussian filter on SIFT. The behavior of correct ratio in (a) is quite interesting. After applying Gaussian filter on the image, the correct ratio becomes better than before. Then it slightly drops along with the increment of blurring. In figure 7.4 (b), the repeatability slightly dropped after the radius is greater than 1. However, after this threshold, the number of corrects matches becomes quite stable. Generally, it coincides with the accepted fact that SIFT is invariant to blurring.

Scale Figure 7.5 shows that scaling influences the SIFT algorithm a great deal. In figure 7.5 (a), correct ratio drops drastically after scale factor 0.4. In figure 7.5 (b), the repeatability drops along with scale factor of the query image becoming smaller. However, it is easy to understand that feature detectors can find more interest points on a larger scale due to the rich texture. Hence correspondences are more likely to be identified.

Illumination Both curves in figure 7.6 indicate that SIFT is invariant to illumination change. The correct ratio is quite stable. It decreases a little when the intensity changes from 65 to 85. Although repeatability curve drops slightly along with the increment of intensity, it is very smooth.

The evaluation shows that performance of the SIFT based feature extraction module behaves similar to the results, conclusion illustrated in [42, 64, 62, 75].

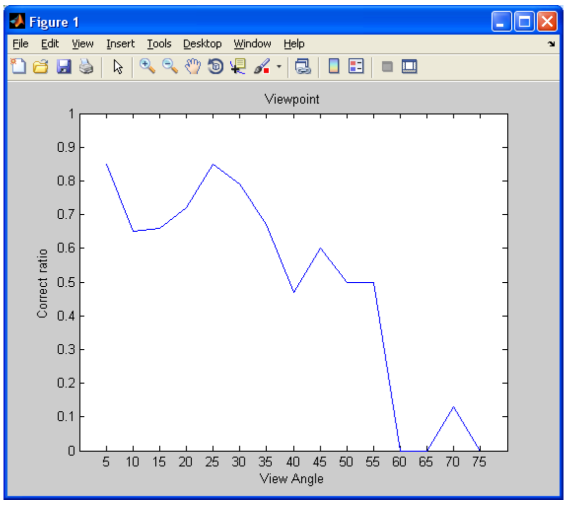


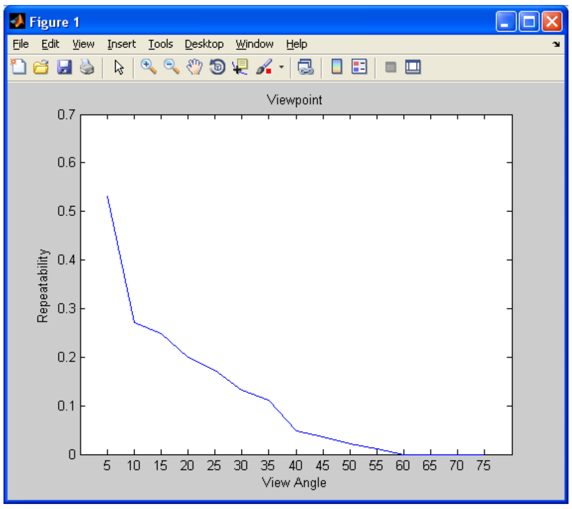


(a) (b)

Figure 7.2 Results of testing using rotated images.

Figure 7.2 (a) shows correct ratio curve under different rotation angle, 7.2 (b) indicates the repeatability curve under the same circumstance.

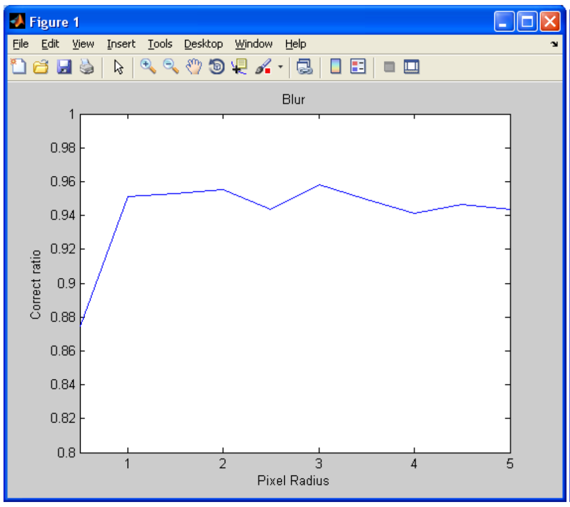


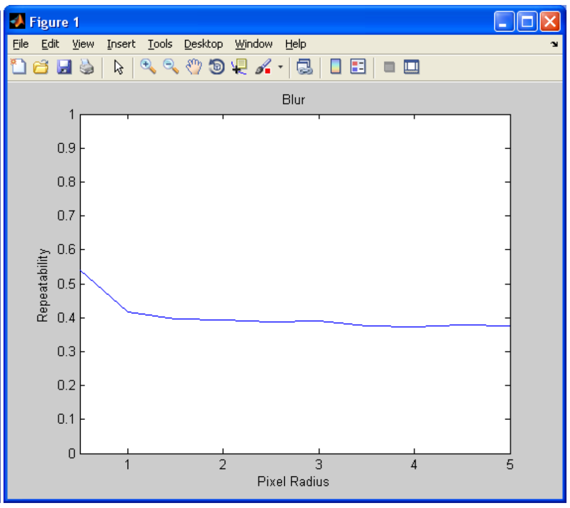


(a) (b)

Figure 7.3 Results of testing under viewpoint change.

Figure 7.3 (a) shows correct ratio curve under viewpoint change, 7.3 (b) indicates the repeatability curve under the same circumstance.

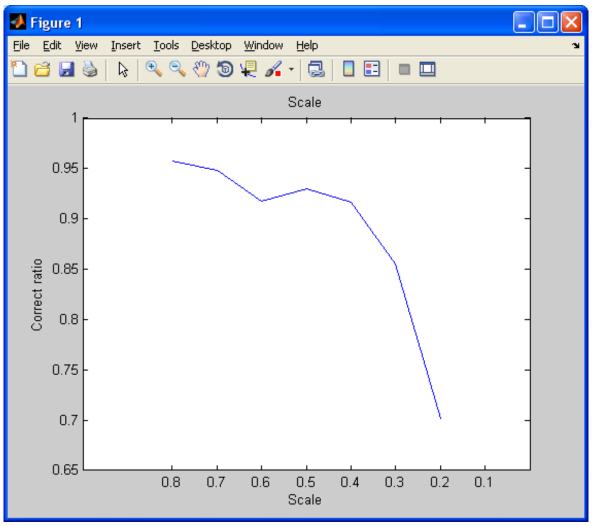


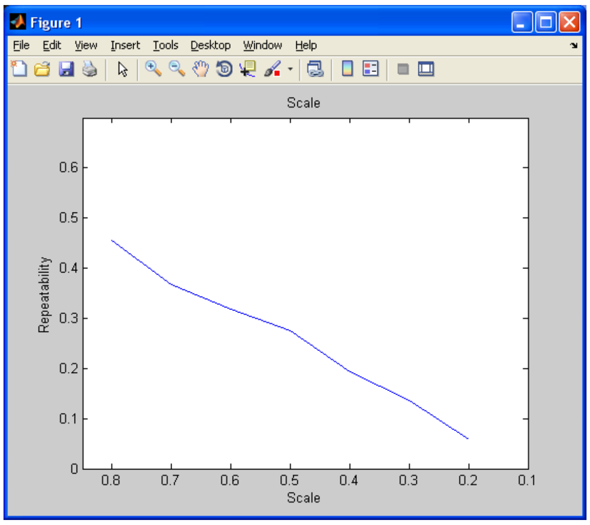


(a) (b)

Figure 7.4 Results of testing under different blurring.

Figure 7.4 (a) shows correct ratio curve under different blurring, 7.4 (b) indicates the repeatability curve under the same circumstance.

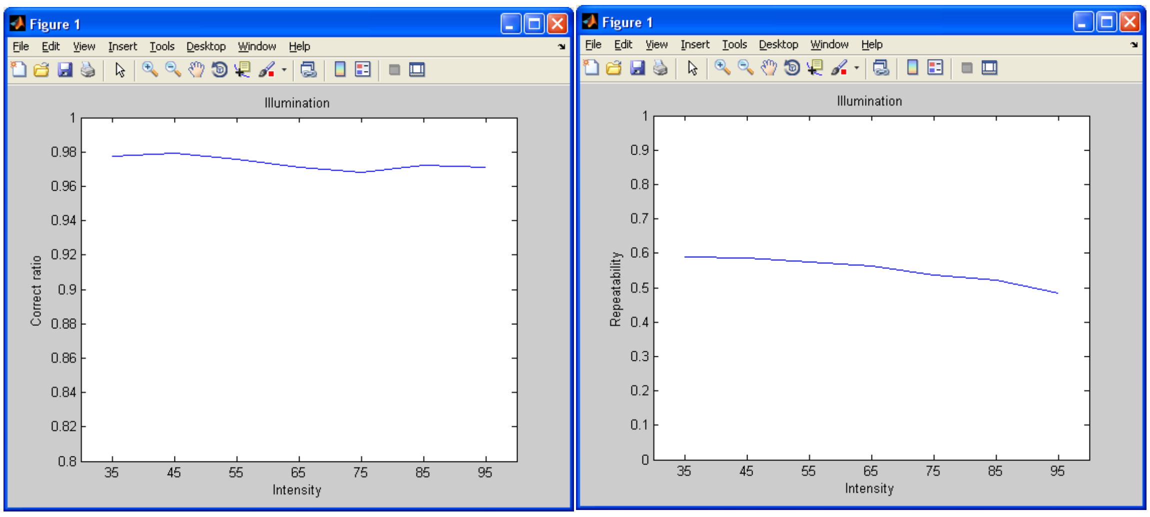




(a) (b)

Figure 7.5 Results of testing under different scale change.

Figure 7.5 (a) shows correct ratio curve under scale change, 7.5 (b) indicates the repeatability curve under the same circumstance.



(a) (b)

Figure 7.6 Results of testing under illumination change.

Figure 7.6 (a) shows correct ratio curve under illumination change, 7.6 (b) indicates the repeatability curve under the same circumstance.

7.2 System Evaluation

We have presented the evaluation and testing of major components in our iNavigation prototype. Using current configuration, the feature extraction module works as expected; we noted accuracy of around 94% with regard to providing the most similar image to the user (10-NN search and 4 returned images for selection). The accuracy of distance estimation is also acceptable with average absolute deviation of 2.2 meters.

Here, we carry out a simple overall system evaluation. At this stage, we generally take into account the system response time and whether the map is updated correctly. 10 images are used as the test dataset. Three of them are from the original database, the others are not. All of them are part of the test dataset used in image search evaluation. The three duplicate images are supposed to be discarded; if the other images are photographed by following the image collection rules, they are supposed to be indicated at roughly the correct position on the 2D map. We also split the positioning task into three parts. Average system response time of each task is recorded. We were not too concerned about usability issues other than system response time. Table 7.1 presents the results.

Table 7.1. Average system response time

Process	Average response time
Image search (feature extraction + search)	13.7 seconds
Distance estimation	10 .4 seconds
Map update	2 .2 seconds
Total	26.3 seconds

We can see that the total average response time is 26.3 seconds, which is not remarkable. According to the system requirements, it should not exceed 15 seconds [94, 95]. However, it is a quite complex task. If we were to split the task into parts and present them to users in the order of execution, no single part’s response time exceeds 15 seconds, especially for image search and distance estimation steps.

The system correctly identified duplicate images and dumped them. Five of the remaining seven images are roughly mapped on the 2D map. They are correctly inserted into two reference images. The 3 remaining images are not updated correctly because of either distance estimation failure (image is not photographed by following restrictions) or there are not enough grids on the map.

Chapter 8

Conclusions

We followed constructive design methodology, discussed our proposed indoor navigation prototype iNavigation using software development lifecycle and implemented the system. Critical comparative evaluations and experiments were carried out to show the correctness of our approaches and performance of the system. In this chapter, we conclude this thesis. Limitations and possible future work will also be addressed.

Conclusions

This research presents a novel image based indoor navigation system following the steps of software development. It realizes the goals of our research, which include enabling navigation by using photos taking by mobile phones and making use of the query images via distance estimation to provide interactive navigation and knowledge sharing to users. Thus the research questions are solved. Recall that research questions are: *what approaches can be used to guide users to destination by using ordinary images taken by mobile phones? And how is the new information from query images to be utilized to perform more interactive search?*

It also achieves the system requirements that were elicited by using natural language and use case at the beginning. After completing the entire process of software development and necessary experiments and testing, we present out main achievement as follows:

1) The prototype is able to locate the user’s current position in a timely manner and uploading images. Considering that we did not implement any actual distance measurements or support devices, the accuracy of positioning is acceptable.
2) The prototype allows the user to view the interior of the building and guides the user to the destination by sliding images.
3) The prototype can make use of query images and annotations submitted by users to enrich its database and subsequently share the information with other users.

Although our prototype has some limitations, it basically shows a “proof of concept” which provides probable answers to our research questions. Limitations of our system can be summed up as:

1) We used HTC Hero G3 built camera to capture building interior photos, other Smart Phones also work with the system as long as they have a built-in camera. However, we set a few restrictions on photograph capture, e.g. approximate height above ground is 1.7 meters and lens should be parallel to the ground. We also used a fixed focal length (e.g. 6 millimeters). Although this may vary depending on different types of mobile phone cameras, focal length can be retrieved from EXIF metadata to update the IPM function as long as the image is in JPEG format. If users do not follow the rules, the results could be incorrect.
2) In order to provide direction guidance, we have to manually assign all direction information to all images in the initial database. However, the directions of accepted query image are correctly updated.
3) The Map grid design limits the total number of images that can be possibly located on the map.
4) Accuracy of distance calculation partially depends on image segmentation approach. If segmentation fails to be performed well, the results could be incorrect.

5) Feature extraction module based on SIFT is only partially Affine invariant, so the system did not show remarkable adoption to viewpoint change images.

The future work can be focused on following aspects:

1) Exporting current UI to an operating system running on smart phone, such as Android or iOS. This could possibly enable some processes to be done on the smart directly rather than on the server side. For example, after a photo is captured, it can be resized before it is sent to the server, which shortens the transmission time. Also, image features can be extracted on the smart phone and then sent to the server. If too few features are extracted from the photo, we can prompt the user to capture another one. We know that the value of focal length can be retrieved from EXIF metadata, if the UI runs on the smart phone, we are able to get focal length via the operating system real time. It is also possible to make use of the embedded sensors on the smart phone to resolve current limitations of photographing. Also sensors like compass, camera calibrator might be worth trying.
2) The database could be scaled up. Current database contains only the images taken on the first floor. More images captured from other floors can be populated into current database to complete the system.
3) Improvements can be made on implementation. It may be worth trying to include implementation of GPU based SIFT, which is extremely fast; implementation of ANN or LSH in C/C++ which is more efficient than Matlab implementation;

implementation of multi-thread programming, which can speed up the system; implementation of Affine invariant feature detectors.

4) Enable the system to recognize objects in the photos. The current system makes use of distance estimation to group images. It enables information sharing but the rich knowledge stored in photos is not made full use of. Object recognition is a possible way to extract the knowledge in the photos. Although it requires a lot offline learning process, the systems may be able to identify the objects in the photos automatically and record types of objects in every image. Then the knowledge can be used for routing and navigation purpose. For example, if a user wants to find a landline on this floor by searching ‘telephone’, the system can find images that contain ‘telephone’ and provide the route. One good example is the application ‘LabelMe’ developed by MIT [99]. It labels objects in the image and provide annotations automatically.

Bibliography

[1] M. Cattenoz. Indoor navigation 2, In Proc. of Advances in Media Technology. pp. 27-34, 2011.

[2] G. Satalich. Navigation and Wayfinding in Virtual Reality: Finding Proper Tools and Cues to Enhance Navigation Awareness. Master's thesis, University of Washington, 1995.

[3] I. Ekman and P. Lankoski. What Should It Do? Key Issues in Navigation Interface Design for Small Screen Devices. In Proc. of International Conference on Human Factors and Computing Systems, pp. 622-623, 2002.

[4] C. Kray, C. Elting, K. Laakso, V. Coors. Presenting Route Instructions on Mobile Devices. In Proc. of International Conference on Intelligent User Interfaces, pp. 117-124, 2003.

[5] B. Ozdenizci, K. Ok, V. Coskun, M. Aydin. Development of an Indoor Navigation System using NFC technology. In Proc. of Fourth International Conference on Information and Computing, pp. 11-14, 2011.

[6] Y. Lu and E. Delp. An Overview of Problems in Image-Based Location Awareness and Navigation. In Proc. of Visual Communications and Image Processing, pp. 102-109, 2004.

[7] H. Karimi. Indoor Navigation. Universal Navigation on Smartphones, Springer New York, pp. 17-51, 2011.

[8] D. Bradley, A. Brunton, M. Fiala, G. Roth. Image-based navigation in real environments using panoramas. In Proc. of IEEE International Workshop on Haptic Audio Visual Environments and their Applications, 2005.

[9] H. Kawaji, K. Hatada, T, Yamasaki, K. Aizawa. Image-based indoor positioning system: fast image matching using omnidirectional panoramic images. In Proc. of the

1st ACM International Workshop on Multimodal Pervasive Video Analysis, pp. 1-4, 2010.

[10] H. Kang, A. A. Efros, M. Hebert, and T. Kanade. Image Matching in Large Scale Indoor Environment. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Egocentric Vision, June, 2009.

[11] C. Slipa-Anan, R. Hartley. Localisation using an image-map. ACRC, 2004.

[12] H. Huang, G. Gartner. A survey of Mobile Indoor Navigation Systems. Lecture Notes in Geoinformation and Cartography. Berlin: Springer, pp. 305-319.

[13] L. Ruotsalainen, H. Kuusniemi, and R. Chen. Visual-aided Two-dimensional Pedestrian Indoor Navigation with a Smartphone. Journal of Global Positioning Systems Vol.10 (1), 11-18, 2011.

[14] M. Werner, M. Kessel, and C. Marouane. Indoor Positioning Using Smartphone Camera. In Proc. of International Conference on Indoor Positioning and Indoor Navigation (IPIN), Portugal, 21-23, 2011.

[15] N. Yazawa, H. Uchiyama, H. Saito, M. Servieres, and G. Moreau. Image based view localization system retrieving from a panorama database by surf. In Proc. of the IAPR Conference on Machine Vision Applications, 2009.

[16] T. Cornford, S. Smithson. Project Research in Information Systems. Macmillan, London. pp. 44. 1996.

[17] V. Renaudin, O. Yalak, P. Tome, B. Merminod. Indoor navigation of emergency agents. European Journal of Navigation, 5, pp. 36- 45, 2007.

[18] S. Gezici, Z. Tian, G.B. Giannakis, H. Kobayashi, A.F. Molisch, H.V. Poor, Z. Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. In Proc. of IEEE Signal Processing Magazine, 22(4): 70-84, 2005.

[19] S. Thongthammacharl and H. Olesen. Bluetooth Enables Indoor Mobile Location Services. In Proc. of the 57th IEEE Semiannual Vehicular Technology Conference, 3, pp. 2023-2027, 2003.

[20] P. Bahl, V. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In Proc. of IEEE INFOCOM, 2, pp. 775-784, 2000.

[21] L. Ni, Y. Liu, Y. Lau, A. Patil. LANDMARC: indoor location sensing using active RFID. Wireless Networks, 10(6): 701-710, 2004.

[22] T. Kitasuka, T. Nakanishi, A. Fukuda. Wireless LAN based indoor positioning system WiPS and its simulation. In Proc. of IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 1, pp. 272-275, 2003.

[23] Y. Wang, X. Jia, H. K. lee, G. Y. Li. An indoors wireless positioning system based on wireless local area network infrastructure. Presented at the 6th International Symposium on Satellite Navigation technology including Mobile Positioning & Location Services, Melbourne, Australia, 2003.

[24] A. Bekkali, H, Sanson, M. Matsumoto RFID Indoor Positioning Based on Probabilistic RFID Map and Kalman Filtering in Proc. of the 3rd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, pp. 21-21, 2007.

[25] R. Want, A. Hooper, V. Falcao, J. Gibbons. The active badge location system. ACM Transactions on Information Systems, 10(1): 91-102, 1992.

[26] R. Ivanov. Indoor navigation system for visually impaired. In Proc. International Conference on Computer Systems and Technologies, Sofia, pp. 143- 149, 2010.

[27] C. Kee, D. Yun, H. Jun, B. Parkinson, S. Pullen, T. Lagenstein. Centimeter-Accuracy Indoor Navigation. GPS World, 2001.

[28] N. Lenihan. A local Optimal User Position System for Indoor Wireless Devices. 2004.

[29] F. V. Diggelen. Indoor GPS theory and implementation. IEEE Position, Location & Navigation Symposium, 2002.

[30] R. Taylor, J. Sennott. Navigation System and Method. US Patent 4445118. NASA, 1981.

[31] K. W. Kolodziej, J. Hjelm. Local Positioning Systems: LBS Applications and Services. Taylor & Francis Group, 2006.

[32] L. Jing, P. Liang, C. Maoyong, S. Nongliang. Super-resolution time of arrival estimation for indoor geolocation based on IEEE 802.11 a/g. In Intelligent Control and Automation, 2008. 7th World Congress on, pp. 6612 –6615, 2008.

[33] C. Feng, W. S. A. Au, S. Valaee, Z. Tan. Compressive sensing based positioning using RSS of WLAN access points. In Proc. of the 29th Conference on Information Communications. pp. 1631-1639.

[34] D. Milioris, G. Tzagkarakis, P. Jacquet and P. Tsakalides. Indoor positioning in Wireless LANs Using Compressive Sensing Signal-Strength Fingerprints. In Proc. of 19th European Signal processing Conference, Spain, 2011.

[35] P. Gilliéron, B. Merminod. Personal Navigation System for Indoor Applications. In Proc. of 11th IAIN World Congress, Berlin, 2003.

[36] T. Nelen, M. Weyn, M. Klepal. Indoor Navigation for Complex Environments, Master Thesis, 2010-2011.

[37] L. E. Miller. Indoor Navigation for First Responders: A Feasibility Study. National Inst. of Standards and Technology (ITL), Gaithersburg, MD. Advanced Network Technologies Div.

[38] S. Jo, Y. Kwon, H. Ko. Indoor Environment Modeling for Interactive VR - Based Robot Security Service. ICAT, Vol. 4282, Springer, pp. 1253-1262, 2006.

[39] Y. Li, Z. He. 3D Indoor Navigation: A Framework of Combining BIM with 3D GIS. In Proc. of 44th ISOCARP Congress, 2008.

[40] H. Du, P. Henry, X. Ren, M, Cheng, D. B. Goldman, S. M. Seitz, D, Fox. Interactive 3D modeling of indoor environments with a consumer depth camera. In Proc. of the 13th International Conference on Ubiquitous Computing, pp. 75-84, 2011.

[41] E. Jeno, S. Jo. Real-time building of a 3D model of an indoor environment with a mobile robot. In Proc. of the 11th International Conference on Control, Automation and Systems (ICCAS), pp. 818-823, 2011.

[42] D. G. Lowe. Object recognition from local scale-invariant features. In Proc. of the 7th International Conference on Computer Vision, Corfu, Greece, pp. 1150-1157, 1999.

[43] M. Pierre, P.Pietro. Evaluation of Features Detectors and Descriptors based on 3D objects. ICCV 2005.

[44] M. Swain, D. Ballard. Color indexing. International Journal in Computer Vision, 7(1), pp. 11–32, 1991.

[45] H. Murase, S. Nayar. Visual learning and recognition of 3D objects from appearance. International Journal on Computer Vision, 14(1), pp. 5–24, 1995.

[46] I. T. Joliffe. Principal Component Analysis. Springer-Verlag, 1986.

[47] T. Tuytelaars, K. Mikolajczyk. Local Invariant Feature Detectors: A Survey. Foudations and Trends in Computer Graphics and Vision, 3(3): 177-280, 2008.

[48] C. Harris, M. Stephens. A Combined Corner and Edge Detector. In Alvey Vision Conference, pages 147-151, 1988.

[49] K. Mikolajczyk, C. Schmid. Indexing based on scale invariant interest points. In Proc. of the 8th International Conference on Computer Vision, Vancouver, Canada, pp. 525–531, 2001.

[50] K. Mikolajczyk and C. Schmid, Scale and Affine invariant interest point detectors. International Journal of Computer Vision 60(1): 63-86, 2004.

[51] K. Mikolajczyk and C. Schmid. An Affine Invariant Interest Point Detector. In Proc. of 7th European Conference of Computer Vision, Denmark, volume I, pp: 128-142, 2002.

[52] M. Brown, R. Szeliski, S. Winder. Multi-image matching using multi-scale oriented patches. In Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp.510–517, 2005.

[53] T. Kadir, M. Brady. Scale, saliency and image description. International Journal of Computer Vision, 45(2): 83–105, 2001.

[54] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60(2): 91-110, 2004.

[55] T. Lindeberg, J. Garding. Shape-adapted smoothing in estimation of 3-D shape cues from Affine deformations of local 2-D brightness structure. Image and Vision Computing, 15(6):415–434, 1997.

[56] A. Baumberg. Reliable feature matching across widely separated views. In Proc. of the Conference on Computer Vision and Pattern Recognition, pp. 774–781, 2000.

[57] K. Mikolajczyk, T. Tuytelaars,C. Schmid,A. Zisserman,J. Matas, F. Schaffalitzky, T. Kadir, L. V. Gool. A Comparison of Affine Region Detectors. International Journal of Computer Vision, 65(1-2): 43-72, 2005.

[58] J. Matas, O. Chum, M. Urban, T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. Image and Vision Computing, 384-393, 2002.

[59] J.M. Morel, G. Yu. ASIFT: A New Framework for Fully Affine Invariant Image Comparison, SIAM Journal on Imaging Sciences, 2(2), 2009.

[60] T. Lindeberg. Feature Detection with Automatic Scale Selection. International Journal of Computer Vision, 30(2): 79-116, 1998.

[61] F. Schaffalitzky and A. Zisserman. Multi-View Matching for Unordered Image Sets. In Proc. of the 7th European Conference on Computer Vision, pp. 414-431, 2002.

[62] N. Y.Khan, B. McCane, G. Wyvill. SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset. In Proc. of International Conference on Digital Image Computing: Techniques and Applications, pp. 501-506, 2011.

[63] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool. SURF: Speeded Up Robust Features. Computer Vision and Image Understanding (CVIU), 110(3): 346--359, 2008.

[64] R. Szeliski. Computer Vision: Algorithms and Applications. Springer, 2010.

[65] J. S. Beis, D. G. Lowe. Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1000-1006, 1997.

[66] S. A. Nene, S. K. Nayar. A Simple Algorithm for Nearest Neighbor Search in High Dimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(9): 989-1003, 1997.

[67] Grauman, K. and Darrell, T. Efficient image matching with distributions of local invariant features. In Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 627–634, 2005.

[68] D. Nister, H. Stewenius. Scalable recognition with a vocabulary tree. In Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2161–2168, 2006.

[69] M. Muja, D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In Proc. of International Conference on Computer Vision Theory and Applications (VISAPP), Portugal, 2009.

[70] C. Schmid, R. Mohr, C. Bauckhage. Evaluation of interest point detectors. In International Journal of Computer Vision, 37(2):151-172, 2000.

[71] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Learning, 27(10): 1615-1630, 2005.

[72] Y. Ke and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In Proc. of Conference on Computer Vision and Pattern Recognition, Washington, USA, pages 511-517, 2004.

[73] K. Mikolajczyk, B. Leibe, B. Schiele. Local features for object class recognition. In Proc. of International Conference on Computer Vision, 2005.

[74] G. Babbar, P. Bajaj, A. Chawla, M. Gogna. A comparative study of image matching algorithms. International Journal of Information, Technology and Knowledge Management, 2(2): 337-339, 2010.

[75] L. Juan, O. Gwun. A comparison of SIFT, PCA-SIFT, and SURF. International Journal of Image Processing (IJIP) 3(4): 143-152, 2009.

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

[77] J. Sivic, A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In Proc. of International Conference on Computer Vision, 2003.

[78] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In Proc. of International Conference on Computer Vision, 2007.

[79] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In Proc. of ACM International Conference on Image and Video Retrieval, 2007.

[80] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In Proc. of the 21th Conference on Computer Vision and Pattern Recognition, 2008.

[81] R. Ke, Y. ad Sukthankar and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In ACM Multimedia, 2004.

[82] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. Numerische Mathematlk 1, pp. 269-271, 1959.

[83] D. B. West, Introduction to Graph Theory second edition, Singapore: Pearson Education, pp. 97-98, 2001.

[84] H. A. Mallot, H. H. Biilthoff, J. J. Little, S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. In Biological Cybernetics, pp. 177-185, 1991.

[85] D. O'Cualain, E. Jones, M. Glavin. Distance Determination for an Automobile Environment using Inverse Perspective Mapping in OpenCV. In Proc. of Signals and Systems Conference, pp. 100-105, 2010.

[86] P. Moreels, P. Perona. Evaluation of features detectors and descriptors based on 3D objects. In Proc. of the International Conference on Computer Vision, pp. 800–807, 2005.

[87] G. Krasner, S. Pope. A cookbook for using the model view controller user interface paradigm in smalltalk-80. Object Oriented Program, 1(3): 26–49, 1988.

[88] K. Wiegers. Software Requirements: Practical Techniques for Gathering and Managing Requirements throughout the Product Development Cycle. Redmond, WA: Microsoft Press, 2003.

[89] L. Moisan, B. Stival. A Probabilistic Criterion to Detect Rigid Point Matches between Two Images and Estimate the Fundamental Matrix. International Journal of Computer Vision, 57(3): 201–218, 2004.

[90] S. A. Nene, S. K. Nayar, H. Murase. Columbia Object Image Library (COIL-20). Technical Report CUCS-005-96, 1996.

[91] R. Hess. An Open Source SIFT Library. ACM Multimedia, 2010.

[92] L. Routsalainen, H. Kuusniemi, R. Chen. Visual-aided Two-dimensional Pedestrian Indoor Navigation with a Smartphone. Journal of Global Positioning Systems, 10(1): 11-18, 2011.

[93] I. Sommerville, Software Engineering (8th ed.). Harlow, England: Pearson Education, 2007. ISBN 0-321-31379-8.

[94] J.A. Hoxmeier, C. DiCesare. System response time and user satisfaction: an experimental study of browser-based applications. Proceedings of the Americas Conference on Information Systems, pp.140-145. 2000.

[95] F. Nah. A study on tolerable waiting time: how long are Web users willing to wait? Behaviour & Information Technology, 23(3):153-163.2004.

[96] L. Copeland. A Practitioner's Guide to Software Test Design. Artech House, Norwood, MA, U.S.A., 2003.

[97] G. J. Myers. The Art of Software Testing, Second Edition. John Wiley & Sons, Inc., Hoboken, New Jersey. 2004.

[98] A. Cockburn. Writing Effective Use Cases. Addison-Wesley, c. 2001.

[99] B. Russell, A. Torralba, K. Murphy, W. T. Freeman. LabelMe: a database and web-based tool for image annotation. International Journal of Computer Vision, 2007.

[100] Z. Wang, A. C. Bovik. Mean Squared Error: Love it or Leave it? IEEE signal processing magzine, 98-117, Jan 2009.

[101] L. Pauleve, H. Jegou, L. Amsaleg. Locality sensitive hashing: a comparison of hash function types and querying mechanisms. Journal of Pattern Recognition Letters 31(11):1348-1358, 2010.

[102] A. Andoni, P. Indyk. Near-Optimal Hashing Algorithms for Near Neighbor Problem in High Dimensions. In Proceedings of the Symposium on Foundations of Computer Science (FOCS'06), 2006.

[103] A. Andoni, M. Datar, N. Immorlica, P. Indyk, V. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions Mirrokni. Nearest Neighbor Methods in Learning and Vision: Theory and Practice, MIT Press, 2006.

[104] D. Lin. Statistical Learning Toolbox. (2006, Sep 20) Available: http://www.mathworks.com/matlabcentral/fileexchange/12333-statistical-learning-toolbox

[105] H. Jaspers, B. Schauerte, G. A. Fink. SIFT-based Camera Localization using Reference Objects for Application in Multi-Camera Environments and Robotics. In Proc. of International Conference on Pattern Recognition Applications and Methods, Portugal, 2012.