Monocular Vision based Particle Filter Localization in Urban Environments

by

Keith Yu Kit Leung

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, August 2007

Abstract

This thesis presents the design and experimental result of a monocular vision based particle filter localization system for urban settings that uses aerial orthoimagery as a reference map. The topics of perception and localization are reviewed along with their modeling using a probabilistic framework. Computer vision techniques used to create the feature map and to extract features from camera images are discussed. Localization results indicate that the design is viable.

Acknowledgements

The author would like to express his appreciation for the positive support and consultation received from his supervisors, Christopher M. Clark, and Jan P. Huissoon, while performing the research presented in this thesis. The author would also like to express his gratitude to his family for their support and encouragement during his studies

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The field of robotics has achieved great success and recognized to have many potential applications. As a definition, robotics can be considered the science of perceiving and manipulating the real physical world through computer controlled devices [1]. Industrial manufacturing is one large sector that has benefited from robotics, where it is now a common sight to see robotic arms and manipulators installed beside assembly lines. However, one thing that manipulators lack is mobility as they are usually fixed at a certain location. Mobile robotics involves robots that have the freedom to travel and this mobility has provided benefits in diverse fields of application. However, possessing mobility also introduces difficulties which need to be addressed, especially when a robot is required to operate autonomously.

There are numerous fields in which mobile robots can be found. Perhaps one of the first things that will come to mind is planetary exploration or aerospace applications, made famous by the NASA Mars missions involving the rovers Sojourner (deployed in the Pathfinder mission in 1997), Spirit, and Opportunity (both of which were deployed in 2004). While the rovers are great achievements, one does not have to go to Mars to find mobile robots. Closer to home, mobile robots are found in humanitarian de-mining operations throughout regions of the world that have had a recent history of conflicts [2]. Robots are also used in the removal of explosives as well as urban search and rescue operations. In less hostile environments, mobile robots can be found as tour guides in museums [3] or even as a vacuum cleaner [4]. As for transportation, warehouses, manufacturing plants, and container ports [5] profit from using teams of robots for moving goods to desired locations. Mobile robots have also found their application in personal transport in urban environments such as the CyCab [6] developed in France. Off the ground, the military is very keen on the usage of autonomous un-manned aircrafts for surveillance on the modern battlefields [7]. For underwater applications, numerous institutes and companies have developed autonomous underwater vehicles (AUV) for research in lakes and oceans [8]. The applications listed above are only a sample from a larger list of applications. In general, mobile robots are employed in performing tasks where either people can not get to or where the environment is too hostile for human presence. They are also used to autonomously perform endless repetitive tasks without consideration of fatigue. In all the applications listed above, accomplishment of the desired tasks requires a robot to recognize how and where it should move.

1.1 Autonomous Navigation

Autonomous navigation continues to be a challenge in mobile robotics. However, the methods by which navigation is carried out is very similar to the methods used by humans. Usually the objective is to travel to a goal defined in the configuration space (defined as all allowable states for the robot), and may involve visiting waypoints in between. Performing this task successfully requires achieving the four navigation fundamentals of perception, localization, cognition, and motion control [9].

Perception involves gathering information of the environment or actions performed by a robot using sensors, and interpreting the data to something meaningful for the navigation system. Humans mainly rely on vision to perceive the surrounding world and robots too can be given the ability to see through a camera. Besides vision, robots can rely on other sensors such as rangefinders to perceive distance to objects or gyroscopes to determine self orientation. Beacon based sensors such as global positioning system (GPS) are also commonly used.

Localization is also known as state estimation. The state of a robot often contains multiple variables in a vector that uniquely define a robot in the its workspace. For a mobile robot, the state vector will usually define the position and orientation (or heading direction) of the robot. Other variables that may be included in the state vector are velocities and accelerations. In general, the variables included in the state vector is application specific, and state variables can be omitted if they have no influence on the task that a robot needs to achieve [1]. Being able to localize is an important part of a robot system because actions taken to cause state transitions are commonly dependent on the latest state estimate. In navigation, a robot needs to have the best estimate of where it is to determine how it should continue to control its own motion to reach the desired goal. In order to localize, it is necessary to have a perception of the surrounding environment by obtaining measurements through sensors. A sensor may or may not be a direct measurement of the robot state and often times it is necessary to determine the state by referring to a measurement model and a map. For instance, a GPS receiver is able to directly measure the position of a robot and only the transformation between the GPS frame of reference and the local frame of reference is required. On the contrary, a laser rangefinder can only measure distance to an object and it is necessary to refer to a map in order to perform state estimation. A map contains information of a robot's workspace. Since robots rely on this information for localization, it should be as accurate as possible. Any deviations from reality may cause a robot to exemplify unusual and undesirable behaviours.

In some applications, a map is unavailable and a robot is required to produce a map for itself while localizing concurrently. The problem is known as simultaneous localization and mapping (SLAM), and it is an area of heavy research interest in mobile robotics. The SLAM problem can be further divided into two types; the online SLAM problem attempts to estimate the latest state and map, while the full SLAM problem attempts to estimate the current and all previous states [1]. The extensive research in this area has produced numerous variations of algorithms. Besides sensor measurements, motion control inputs and the previous state estimate are necessary to account for the state transition between measurements. The dependence of an state estimate on a previous state estimate makes localization a recursive process. Approaches using a probabilistic approach derived from the Bayes filter is common in localization. Two popular methods are the Kalman filter and the particle filter. With a probabilistic approach, a probability density function is used to describe the belief state. Therefore, it is possible to generate a multiple hypothesis belief state which correspond to multiple local maxima in the probabilty density function. This capability is considered superior to a single hypothesis belief state estimation because a single belief state is incapable of handling situations of ambiguities, where for instance the same sensor measurement may be obtainable from being in various states.

Cognition is the decision making step. Having localized or having a certain degree of confidence in the belief state, it is necessary to plan how the robot should move as it works its way towards the goal while avoiding obstacles or potentially dangerous regions in the workspace. There exist many methods in literature [9, 10] for performing path planning. In general, planning algorithms can be classified as a road map, cell decomposition, or potential field.

Motion control involves determining how the actuators on a robot are to operate in order to follow the planned path or trajectory. This is specific to the design of a robot and the configuration of its actuators. Knowledge of the dynamics or kinematics models of the robot system is usually required to calculate the proper control inputs.

The four successive tasks described above form the architecture of map-based (or modeled based) navigation. An alternative to map-based navigation is behaviour-based navigation where perceived sensor data is mapped directly to motion control inputs for the actuators. This approach is computationally less demanding but generally less flexible compared to the map-based approach if the workspace is changed. Hybrid approaches that combine the two navigation models also exists [9].

Many mobile robots have been shown to perform successfully in indoor applications in the past. Indoor or laboratory environments tend to be more structured and consistent, therefore perception, localization, and motion control can be achieved with greater consistency and accuracy. Achieving autonomous operations outdoor however is still generally a more difficult problem compared to indoor applications. Outdoor environments can be more dynamic and unpredictable, and this leads to greater uncertainty. The performance of sensors may be degraded in outdoor environments as well. For instance a laser range finder has limited range and may not detect objects in large open areas. Additionally, it is unpredictable what obstacles or objects the robot will come across. The surfaces of particular objects may be noisy, cause faulty range readings, or prevent the detection of the objects completely. Uncertainty also exists in motion control. While control noises are experienced in all environments, models used in determining control inputs may be insufficient in accounting for all outdoor environments. Consider terrain type as an example, a wheeled robot that wishes to accelerate to a certain velocity will perform differently if the surface of which it sits on is wet, dry, icy, or sand covered unless there exists terrain dependent models which the robot can correctly identify and use. Overall, managing the uncertainty is perhaps one of the biggest obstacles in designing an outdoor robot that can operate robustly [1].

Uncertainty in robot perception and action has promoted the use of probabilistic techniques in robotics, where uncertainty is represented according to probability theory. With this approach, it is not necessary to rely on a single best estimate of sensor measurement or real actuator output. Instead, information is represented using probability distributions where uncertainty can be quantified. When the probabilistic paradigm is applied to localization, a robot can represent its current belief state as a probability density function over the state space (representing all possible robot location and poses). The distribution is then updated according to sensor measurements or robot interactions with the environment.

1.2 Thesis Objective

This thesis proposes the design of an urban outdoor localization system using a high resolution aerial image as the map of an operation workspace while attempting to minimize the cost and complexity of the the sensors involved in the implementation of the solution. The urban outdoor environment is chosen in this investigation to address the problem where beacon based sensing and localization (such as with the use of GPS) is not possible or has degraded performance due to buildings in the operation area that interfere with beacon signals [11].

The use of an aerial image as the map for localization is a new challenge. A computer is required to process the given aerial image to pick out an appropriate set of features that will serve as the navigation map. These same features must also be identified from information gathered from on board sensors. By matching the features obtained from the sensors with those from the aerial image, the robot's pose can be estimated. The ability to perform the feature extraction and matching processes will have a direct influence on the ability to localize correctly. Being able to achieve localization with an aerial image provides an additional degree of autonomy for a robot system, as it will show that a human operator is no longer required to manually compile a map that is comprehensible by machine. So far no research has been published regarding the use of aerial images for urban localization.

Minimizing the number of sensors is beneficial because it reduces the physical space required for installation and thus promotes the use of the proposed solution on robot platforms of various sizes. Furthermore, interfacing of the hardware components to the robot computer becomes simpler, with less components to connect. Besides quantity, the types of sensor that will be used also needs to be considered in terms of cost, and the information that they will provide. The drawback of minimizing the cost and complexity of on board sensors is a reduction in information available. Sensor aliasing (where different objects look the same to a sensor) may also become more noticeable. Therefore, in general it becomes harder to localize. Robots that have been successfully employed in urban outdoor environments in the past rely on an array of different sensors to perceive the environment, such as the one presented in [12] which uses multiple cameras, a laser scanner, proximity sensors, and GPS. So another challenge in this thesis is to achieve localization with less resources.

The end objective is to achieve autonomous localization in an outdoor urban environment defined by an aerial image without knowledge of the initial position and heading.

1.3 The Proposed Solution

The proposed solution uses a single digital camera as the only sensor connected to the localization system for perception. This allows the aim of minimizing cost and complexity of hardware use in the localization system implementation to be achieved.

Other researchers have tried using monocular vision in localization because of the simplicity of the hardware involved. However, the work in this area is currently limited, suggesting that the problem of localization using only monocular vision has not been an area of concentrated research previously. Only recently has related work been published in relation to this problem In recent publications, researchers have tried using a database of images to serve as the map in localization. In [13], images at various points in the operating workspace is tagged with GPS position readings and stored in a database. Matching of features between the database images and the on board camera images was carried out by performing scale invariant feature transform (SIFT). In [14], the appearance of the city skyline from various locations were used instead as the map and similar work are indicated to have been done by looking at the details of building facades. In the most related and recent work presented in [11], a robot is first guided through a course as it captures a video of the scene. The information is used offline where distinct image features are selected to generate a three dimensional map. The robot is then shown to localize itself using on board camera images using a trajectory close to the path which the robot took in generating the map. The methods highlighted share the similarity that a map is created by capturing images at known locations and then compared to on board camera images when localization is performed. The approach that this thesis undertakes is different in that it is not necessary to obtain on board images of the environment before performing localization. Instead, this information will come from an aerial image, which is a highlight of the proposed design.

Image processing techniques are applied to the aerial image to highlight building boundaries (walls), which are obstacles that a robot must avoid as it travels through the workspace. The building boundaries are also a good feature to look for because they can be seen from the on board camera, and thus there is a similar type of object that can be compared between the aerial image and an on board image for localization. A shape detection process based on the Hough Transform is used to convert the building boundary information on the aerial image to a set of line segments, a higher level parameterized representation. For the on board camera images, a similar technique is used to highlight features on building walls. It is assumed that most man-made structures have walls that are planar and orthogonal to the ground. With this assumption, the orientation of a building boundary can be estimated using vanishing points and compared directly to the aerial map. There are existing methods for vanishing point detection and the proposed localization system design enhances these methods to better suit the purpose of the system.

Feature comparison will be carried out as a component of the particle filter. This localization method based on the probabilistic framework of the Bayes filter is used because of its ability to perform state estimation with unknown initial position and orientation. In the implementation and testing of the localization filter, a real robot will not be used. Instead, the equipment that will be involved is only a computer and a camera. Therefore, the motion control component of navigation is done manually. It is assumed that state transition (motion) information (limited to forward velocity and yaw) is known but noise will be artificially added to imitate a robot with known motion control inputs. Also, the system is tested offline using saved camera footage.

The way in which the proposed solution for localization fits into the map based navigation framework is shown in figure 1.1.

This figure shows the four components of map based navigation and their relationships with each other. Since a probabilistic approach is taken for state estimation, it is expressed as a probability density function $p(x)$, where x is the state vector. The state estimate is used in motion planning, which generates the controls u necessary to maintain the planned trajectory. The control inputs causes the actuators to interact with the environment to produce physical motion, and at the same time this information is made available to the particle filter to allow the localization system

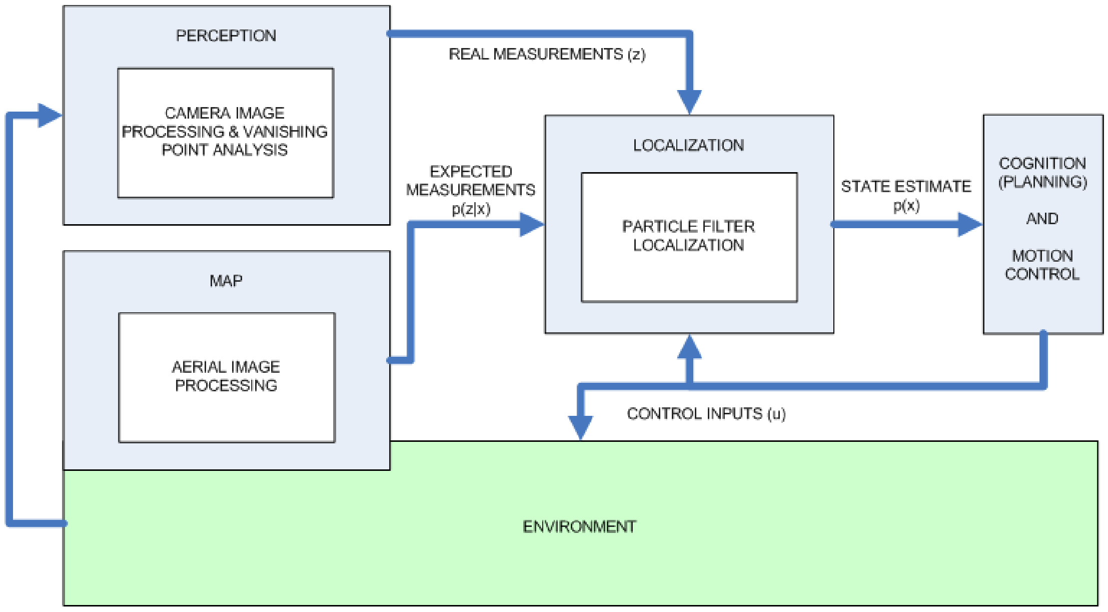

PERCEPTION
CAMERA IMAGE PROCESSING & VANISHING POINT ANALYSIS
REAL MEASUREMENTS (z)
LOCALIZATION
PARTICLE FILTER LOCALIZATION
EXPECTED MEASUREMENTS p(z|x)
STATE ESTIMATE p(x)
COGNITION (PLANNING) AND MOTION CONTROL
MAP
AERIAL IMAGE PROCESSING
CONTROL INPUTS (u)
ENVIRONMENT

Figure 1.1: Layout of the proposed localization system design in relation to the navigation process

to account for state transition due to the control inputs. The information regarding the changed state of the robot in the real world is captured by sensors. The sensor measurements z are then compared to the expected measurements, which is expressed as a probability density function using the sensor model and the perceived state. The remainder of this thesis contains discussions on perception and localization in greater details. Implementations of the proposed solution will follow, explaining the image processing and computer vision techniques involved in manipulating camera images and the ariel map, as well as how the information is used by the particle filter. Results and discussion on the localization result is presented at the end.

Chapter 2

Perception

Sensors are vital to an autonomous mobile robot to allow it to perceive its operating environment in relation to its own position, whether the robot is trying to localize itself in order to navigate, create a map of the surrounding, or simply avoid obstacles in its path of travel. This thesis investigates the ability to perform autonomous localization using a camera as the only available sensor.

The rationale behind using only a single sensory device is threefold, with the hypothesis that the objective of autonomous localization can be achieved at the end. First, it simplifies the hardware requirement, which is beneficial in some applications. This constraint may be due to physical spacing, power requirement, hardware interfacing issues, or even cost. Second, the amount of information obtainable from a camera is greater than from other sensors, but only a single camera will be used to minimize the computation requirement. Third, the result of this investigation can be used as a benchmark for other systems that uses a vision based approach in urban environments similar to that used in this study. Theoretically, providing additional sensory capability besides a single camera should improve localization results by reducing the uncertainty in the estimation and possibly the time required to determine the correct state estimate.

This chapter will provide a general overview of sensors commonly used in mobile robotics. This will include a discussion of sensor types and classification. It will then focus on the laser rangefinder and global positioning system (GPS), highlighting advantages and disadvantages of each device, as they too can conceptually be used in the presented localization problem. This will be followed by a section on the camera and the chapter will finish off with a look at how the probabilistic approach is applied to sensor readings.

2.1 Sensor overview and classification

Sensors can be classified as proprioceptive or exteroceptive and as active or passive [9]. Proprioceptive sensors allow system internal measurements to be made. An example of this type of measurement for a mobile robot is the rotational speed or position of a wheel. These sensors allow a robot to know what it is doing but does not provide information in relation to the surroundings. Attempting to track or perform localization based on proprioceptive sensors alone will likely yield inaccurate results because a robot's internal measurements or belief in its own actions will often deviate from reality due to sensor noise and error. An exteroceptive sensor on the other hand is able to take measurements of the robot's environment. These measurements can further be used to extract features that are representative of the surrounding environment. For the localization problem being studied, exteroceptive sensors are required, and a camera is one example of this type of sensor.

An active sensor is one that emits energy and measures the response to the emission. Note that an active sensor does not necessarily have to be an exteroceptive sensor, and an optical encoder that is used for measuring the rotation of a wheel is one such example. Another example of an active sensor is a laser range finder which emits a laser beam and takes measurements on the reflecting beam. A passive sensor is one that measures the energy present in the environment without emitting any energy. Hence it does not look for any reflections but examines the energy presently existing in the environment. A camera is a passive sensor and gives information regarding light in the environment.

Sensors commonly installed on mobile robots can generally be classified into seven groups. These are identified in table 2.1 [9].

2.2 Laser rangefinders

Rangefinders are popular devices because the sensor readings represent distances to targets, which are very easy to interpret. These sensors can be electromagnetic, laser, or ultrasonic based. However, for outdoor ground applications, laser rangefinders are usually preferred over ultrasonic sensors because of their effective range. The maximum range for an ultrasonic sensor is roughly five meters, which is not useful in large open spaces. In addition, ultrasonic sensors tend to underperform laser rangefinders in terms of accuracy, bandwidth (frequency of measurements), and cross-sensitivity (susceptibility to noise generated by other sources in the environment).

The working principle behind a laser rangefinder can be based on time of flight measurement, phase shift measurement, or optical triangulation [9]. The time of flight measurement calculates the distance to an object by using the speed the light propagation and the time it took for a laser beam to be reflected back to the sensor. For rangefinders that uses phase shift measurements, the phase of the transmitted signal is compared to the reflected signal to determine distance traveled. However

Classification	Description
Tactile sensors	This is the most primitive type of sensor, which activates when they experience physical contact. Example: Contact Switch
Wheel sensors	Proprioceptive sensors that measure the position or the velocity of wheels on a robot. Example: Optical Encoder
Heading sensors	These sensors are able to determine the orientation of a robot in an external reference frame. Example: Digital compass
Beacons	Beacons emit signals that are used to triangulate a position solution in a given reference frame. At least three beacons with known positions are required to acquire a position fix. Providing more than the minimum number of beacons can help reduce error. Example: GPS
Active-ranging sensors	These popular sensors use either time of flight measurements, phase shift measurements, or optical triangulation to determine the distance to a target. Example: Laser rangefinder
Motion sensors	These sensors are able to detect velocity and acceleration Example: Inertial measurement unit (IMU)
Vision-based sensors	Vision sensors capture the light in the environment and provides a two dimensional representation of the three dimensional world. These sensors are not limited to operating in the visible section of the electromagnetic spectrum. They are able to provide an abundance amount of information about the surrounding. Example: CCD camera

Table 2.1: Sensor classification

the maximum range is limited by the wavelength of the emitted signal. In optical triangulation sensors, the reflected light beam emitted by the sensor is focused by a lens. The position of this focus point will vary depending on the distance to the target object from which the light beam is reflected.

The sophistication of laser rangefinders available in the market varies greatly, and the cost can range from tens of dollars to thousands of dollars. Low cost ranging devices usually only provide a measurement to objects directly in front. More sophisticated units are able to perform quick plane sweeps and return multiple range measurements over an arc with sub-degree resolution. Laser rangefinders capable of performing such sweeps are referred to as LIDAR (light detection and ranging) sensors, as shown in figure 2.1. Further details on LIDAR design and operation can be found in [15]. Many groups involved in outdoor mobile robotics applications have used these sensors, and a few examples can be found in [16, 17, 18].

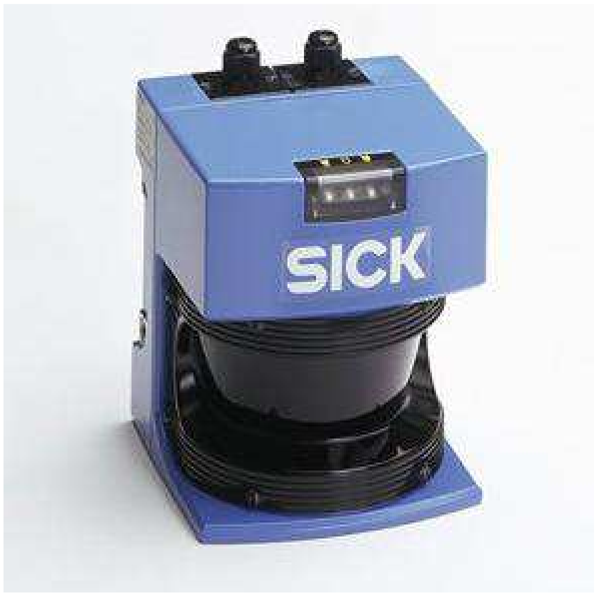


Figure 2.1: SICK laser scanner (LIDAR)

Top of the line LIDAR can have a range of 250 meters while performing sweep

Laser Scanner Model	Maximum range [m]	Power Consumption [W]
SICK PLS	50 m	17 W
SICK LMS 221-30206	80 m	20 W
SICK LD-LRS	250 m	36 W

Table 2.2: LIDAR range vs power consumption

scans over a 360 degree arc. In practice however, the effective range of these devices may be significantly less (up to 90% less) than the maximum range, depending on the reflectivity properties of objects being scanned.The effective range of these sensors is proportional to their power, as shown in table 2.2 [19].

As evident from table 2.2, a LIDAR sensor with scanning range suitable for outdoor applications can consume a lot of power for a mobile robot, which usually depends on an onboard battery to supply power. Also, the size and weight of a LIDAR sensor may make it difficult to package on a mobile platform. A long range scanner can weigh up to 3 kg.

For the urban environment localization problem presented, a LIDAR sensor is a viable tool to use as it possesses excellent sensor range, and accuracy. Its performance is reflected in high cost compared to other sensors, and this is a drawback in some situations, aside from power consumption, physical size, and weight. It is also susceptible to detecting unforeseen obstacles that a robot may encounter (such as people) in the operating environment. In such a case, it is up to the sensor supporting software to perform the necessary filtering.

2.3 Global positioning system

Beacon based positioning systems use multiple emitters placed at known locations and with known signal emitting times to triangulate the position of the receiver in a reference frame by measuring signal time of flight. Satellite systems are especially useful in outdoor environments where satellites orbiting the Earth serve as beacons. The global positioning system (GPS) commissioned by the Unites States of America is one such system, but it is not the only one in existence. Russia also has a satellite positioning system known as the global navigation satellite system (GLONASS). The European Union is also launching a satellite system called Galileo, which will be functional in the near future. The systems identified above are able to provide global coverage. Some countries are launching their own satellites to provide regional coverage. While there are military incentives in the installation of these satellite positioning systems, they have also benefited civilian applications, one of which is outdoor navigation for mobile robots. The remainder of this section will focus on GPS usage and performance.

The GPS consists of an array of 24 satellites in 6 orbital planes to provide global coverage. In an ideal situation, only three satellites are required to obtain a position fix for ground applications if signal timing information is available. One can visualize this by representing the signal emitted from satellites as spherical surfaces. The intersection of two spherical surfaces is a circle, and with a third sphere yields two points. One of these points should be on the surface of the Earth, which can be identified by the calculated elevation and this point is taken as the solution. Another way to think about this is mathematically: there are three variables to solve for to define a point in 3d space, which leads to the requirement of three satellites [20]. In the real world, timing information also needs to be estimated and as such a fourth satellite is required to determine the solution [21]. GPS signals are distorted by many factors, which manifest as psuedorange (the perceived satellite to receiver distance) errors. The proper name for this error is user-equivalent range error (UERE), which can be considered the statistical sum of all error contributions associated with a particular satellite. To overcome this problem, a GPS receiver will lock on to the signals from more than four satellites to reduce the error in the calculated position using a least squares approach [21]. Commercially available GPS receiver units are usually capable of tracking up to twelve GPS satellites simultaneously and will not provide a solution unless it is able to track at least four satellites. The solution to a position fix is given in geodetic coordinates (longitudes, latitudes, and elevation) according to the world geodetic system (WGS-84). A closed form solution exists to convert the geodetic coordinates to Cartesian coordinates in an Earth centered Earth fixed (ECEF) reference frame. For more information on this matter, refer to [20].

The signal transmitted by a GPS satellite for distance(pseudorange) measurement is called pseudorandom noise (PRN) code. Satellite trajectories and system time are also embedded into the code, allowing a receiver to know when and where a signal is sent so that distance can be calculated. However, timing error and the inability to maintain perfect synchronization will offset the clocks on both satellites and receivers; hence calculated distances are referred to as psuedoranges. The code transmitted by each satellite using a code division multiple access (CDMA) method is performed on two frequencies (L1 at $1575.42MHz$, and L2 at $1227.6MHz$) which allow for two types of services. These are known as standard positioning service (SPS) and precise positioning service (PPS). SPS is primarily for civilian use and relies on the coarse acquisition (CA) code transmitted by the satellites that repeats every 1 ms. PPS is mostly used in military applications and requires the precision (P, or also known as the encrypted, Y) code which repeats itself every 7 days. Cryptographic features known as selective availability (SA, which intentionally induces error in satellite clock and navigation data) and anti-spoofing (AS, for preventing signal jamming) are designed into PPS codes to limit accuracy [20]. Fortunately, the SA feature has been

	PPS	SPS
Horizontal plane positioning accuracy	$22m$	$100m$
Vertical plane positioning accuracy	$27.2m$	$156m$
UTC (coordinated universal time) accuracy	$200ns$	$300ns$

Table 2.3: SPS and PPS comparison

disabled since May 2000 to allow civilian service codes to achieve greater accuracy [22]. Table 2.3 [20] compares the performance of SPS and PPS. The values listed should be achieved statistically 95% of the time.

Realistically, positioning accuracy that is much higher than the specified values can be achieved for outdoor applications. In a GPS error budget presented by [20], the UERE for SPS without SA is estimated at $8.0m$ while that for PPS is at $6.6m$. A wide area differential service (WAAS) exists to serve as an enhancement for standalone GPS service. This system uses four geostationary satellites (known as the INMARSAT civil navigation satellite overlay) to transmit information to ground stations operated by service providers such as the Civil Air Administration (CAA). The ground stations are able to estimate positioning error in an area and relay this information to any GPS receiver to improve their accuracy [20].

There are many sources of error for a GPS solution, and the performance of a receiver depends on its psuedorange measurement quality. The model is used to compensate for certain effects, and the accuracy of the satellite ephermeris (trajectory) data which the receiver obtains. In general, GPS error is mathematically expressed as a product between geometric error (caused by the relative location of satellites and the receiver) and the pseudorange error [20].

Selective availability (SA) used to be a major source of psuedorange error. This feature involves intentionally inducing error in satellite clock and trajectory data that is broadcasted to receivers, and can cause errors of up to $70m$ that oscillates every 4 to 12 minutes. This feature is currently disabled to allow improved SPS accuracy, but when activated, its spatially correlated effect can also be overcome by differential GPS (DGPS) services [20, 22].

Multipath is another major source of psuedorange error, and occurs when satellite signals reflect off objects in an environment. These objects may be buildings in an urban environment, or trees in a forest. This is a major concern for using GPS for localization in urban environments. Not only do the reflections cause timing errors, they also create multiple paths to the receiver and at times may confuse a receiver to the point where it loses track of a satellite. Multipath effects are very location dependent but can cause positioning errors of up to $150m$ for SPS and $15m$ for PPS [20, 22].

Other sources of psuedorange error include atmospheric errors that can delay satellite signals. Ionospheric (higher part of atmosphere 70 to 1000 km above the Earth's surface) effect is signal frequency dependent and arises from the interaction between solar activities and the Earth's geomagnetic field. This is very difficult to model and account for but there are models which are able to remove about 50% of ionospheric effects. Typically, signals from satellites that are low on the horizon with respect to a receiver are more prone to this type of error, which can vary from a few to 20 meters in a day. Tropospheric (lower atmosphere) error is a function of the tropospheric refractive index, which depends on the temperature, humidity, and pressure at the receiver location. Most of this signal delaying effect can be accounted for and will only cause a few meters of psuedorange error, but the error can be as high as 20m if the effect is uncompensated. Aside from atmospheric effects, other physical influences include relativistic effects due to the satellites' high speed of orbit, clock errors on satellites and receivers, and receiver noise. For further details and additional sources of GPS error, refer to [20, 22].

Geometric error is quantified by what is known as a dilution of precision (DOP) value and this is affected by the position of satellites and the receiver. Satellite availability also affects the DOP, which is commonly experienced in urban environments when a receiver loses track of satellites due to multipath. The dilution of precision value should be minimized for a more accurate positioning result. Physically, this implies containing the receiver and tracked satellites in an imaginary rectangular box and trying to maximize its volume. Mathematically, the DOP can be thought of as a ratio that compares how big the covariance of the positioning solution is to the covariance of psuedoranges measured. A DOP value can be represented in several ways, the most common of which is called the geometric dilution of precision (GDOP). Other representations that exist include the positional dilution of precision (PDOP), the horizontal dilution of precision (HDOP), the vertical dilution of precision (VDOP), and time dilution of precision (TDOP) [20, 21].

Overall, the GPS SPS available for civilian use is convenient for obtaining position measurements in any outdoor localization and navigation applications. With GPS, it is unnecessary to compare the position measurements to features on a map. All that is required is knowing the coordinate transformation between the global frame of reference used in GPS and the frame of reference used by the robot. However, the unpredictable behaviour of GPS in urban settings due to multipath makes it unreliable for autonomous navigation if it is the only available sensor.

2.4 Cameras

Vision provides an abundance of information about the surrounding. Humans rely heavily on eye sight to carry out many tasks successfully including navigation, and this gives the incentive for providing vision for machines [23]. The benefit of vision is that it is a passive system and theoretically has infinite range as a camera or an eye receives light emitted from any source in the three dimensional world and captures that information on a two dimensional image. Practically, this means that a camera has longer range than most other sensors such as a laser rangefinder.

Providing the equipment to give vision to a machine is not difficult. There are many charged coupled device (CCD) cameras or CMOS cameras available on the market at low cost and they can easily be interfaced with a computer. The difficulty comes in interpreting the data that is retrieved from a camera frame. To a machine, it must make sense of this raw data, which is an array of numbers representing light intensity at each pixel. A human eye captures light in a fairly similar manner to a digital camera, but the processing of a scene is very instinctive and seems to be almost effortless regardless of the complexity or the number of objects in sight. The same cannot be said for computer vision, which is still far from being to mimic human vision. The computation involved in understanding an image can be very demanding, depending on what processes are involved and what information needs to be extracted. Typically, obtaining a high level understanding, such as identifying specific objects and feature extraction in an image is computationally more demanding than acquiring low level features such as edges and corners. Overall, the processing requirement involved in using a camera can be a disadvantage when used on a mobile robot that is limited in computational power.

In some applications, multiple cameras are used to provide stereo vision, just as humans see with two eyes to obtain three-dimensional perception of the surrounding [23]. Stereo vision is definitely an option for localization and has been put into use in the past. However for this thesis, it is desired to determine the performance benchmark with the most basic vision system involving a single camera. Using multiple cameras should improve on what is achievable with a single camera. For further information on this stereo vision and its application to mobile robotics, refer to [9, 24, 25, 26].

The three sensors that were discussed in detail so far (laser rangefinder, GPS, and the camera) are all susceptible to distractions in the environment that may cause errors in measurements. With GPS, it is not possible to identify such noise and filter them before making calculations to determine a position fix. It may be possible to acknowledge the presence of multipath or other sources of error based on inspecting the DOP value but the effects can not be corrected for. With laser rangefinders, it may be possible to take distractions into account by excluding outliers in a measurement scan. However, the information available to base this filtering upon is limited and the source of such distractions is difficult to identify. Filtering distractions is also potentially possible and perhaps can be achieved more successfully with a camera using the abundance of information that it receives from the surrounding, but of course this will come with a price of higher computational power. Mobile robots are always required to work in real time and demonstrating that this is possible using a camera will be a challenge addressed in this thesis.

The camera that will be used to investigate the urban localization problem will be a Logitech Quickcam Pro 5000 webcam as shown in figure 2.2. It has a horizontal viewing angle of 48 degrees which was experimentally verified, and is capable of capturing images at a maximum pixel resolution of 640×480. The maximum frame rate achievable by the camera is 30 frames per second.

2.5 The probabilistic sensor model

In the introductory chapter, the probabilistic approach to mobile robotics was briefly mentioned. In this section, the act of perception will be expressed mathematically. This is essential as it forms part of the formulation for solving the localization problem.

To reiterate, the reason for using a probabilistic approach to robotics is to address

Figure 2.2: The Logitech Quickcam Pro 5000 Webcam

the uncertainty involved in all robot actions. Sensor measurements or information derived from the measurements, whether from a laser range finder, a GPS receiver, or from a camera all contain a certain degree of noise and error that leads to uncertainty. Fortunately, noise can be modeled.

In robotics, it is common to see the vector z used to represent a measurement or a series of measurements. The measurement value that is obtained is dependent on where the measurement is taken, or the state of a robot, which is commonly represented as the state vector x. In a deterministic model, the relation between measurement and state is expressed as

$$z = f(x) \tag{2.1}$$

With the deterministic model, there is no consideration for uncertainty, which exists not only in the measurement, but also in the robot state. The probabilistic approach provides the means for accommodating uncertainty and is able to track its propagation. By defining measurement z and state x as vectors of random variables, a measurement can be described by a conditional probability density function

$$p(z|x) \tag{2.2}$$

Some sensors (such as a camera or a laser rangefinder) provide measurements based on objects present in the surroundings, which should be included in the model as well. Features in the environment collectively compose a map, which can be expressed as the vector m. For the localization problem, the map is considered known or deterministic. Thus the measurement process can be expressed as

$$p(z|x, m) \tag{2.3}$$

The Gaussian distribution is commonly used as a basis for measurement models but this is not a requirement. Using a set of sensor data with a known truth reading is a good way of determining the measurement model. The next chapter will show how a measurement model is incorporated into the localization process.

Chapter 3

Localization

Localization contributes an element of feedback in navigation as shown in figure 3.1. This feedback element is essential for successful navigation due to the errors that accumulate under open-loop control of a mobile robot. This uncertainty in actuator performance and how the actuators interact with the environment (especially outdoor) leads to uncertainty in motion. In mobile robotic applications, localization usually refers to the estimation of the position and orientation of a robot in a reference frame. In general, localization refers to state estimation, and the state should include all the important parameters necessary to define the robot in a specific application. When a robot tries to perform localization without sensors to perceive the environment (in other words, using only proprioceptive sensors or only knowledge of the motion control inputs), the action is known as dead reckoning, and this usually leads to poor navigation results in the long run due to errors inherent with actuator control or interaction with the environment that integrates (or accumulate) over time.

A map is a required component in localization because it is the only item that indicates to a robot how the surroundings should look if it has perfect perception

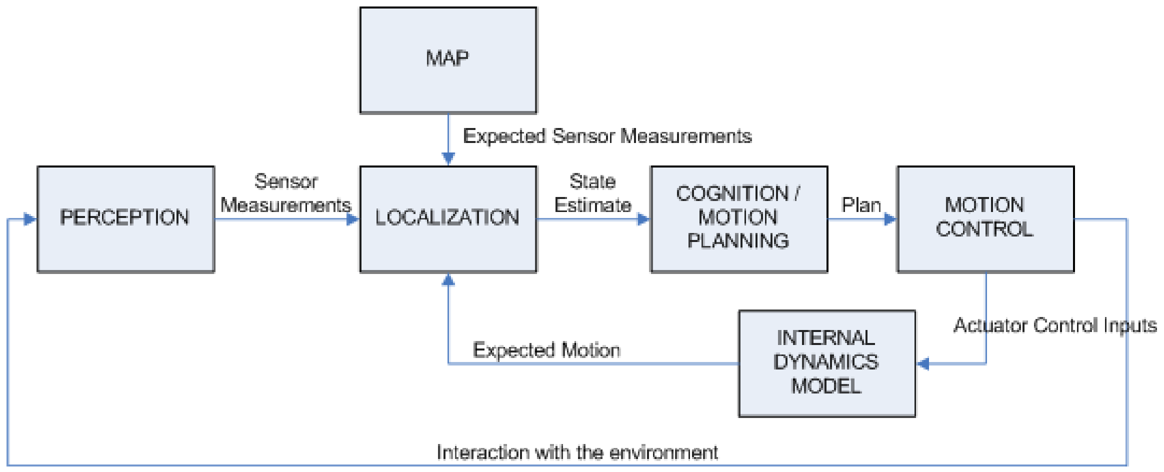


Figure 3.1: The navigation loop

abilities. The map is what is used to estimate the state when sensor measurements are available. The way in which a map is represented should reflect the characteristics of perceived sensor information, and in general there are two types of representation: featured based map and location based map [1]. The feature based map is simply a list that contains all the relevant features in the operating environment and their properties. The next chapter will present the creation of a map from an aerial image using this type of representation, which coincides with the information (features) extractable from camera images. A location based map describes the entire operating environment by defining all the occupied and non-occupied spaces, an example of which is an occupancy grid. Range measurements to obstacles are sensor representations that may work well with a location based map.

The challenge in localization for a mobile robot is to try and infer the state of the robot from measurements and control inputs provided for motion control that contain a certain degree of uncertainty. At the end of the previous chapter, the probabilistic model was introduced to account for the uncertainty in perception due to sensor noise. Another problem that will definitely arise from perception is aliasing

[9]. Due to limitations in the information that can be perceived by a sensor , it is often impossible to distinguish where a measurement was taken to define the state of a robot (see figure 3.2 for an example). The uncertainty that comes from noise and aliasing is carried on to any state estimation that is made using sensor measurements, and hence the probabilistic framework needs to be extended for state estimation.

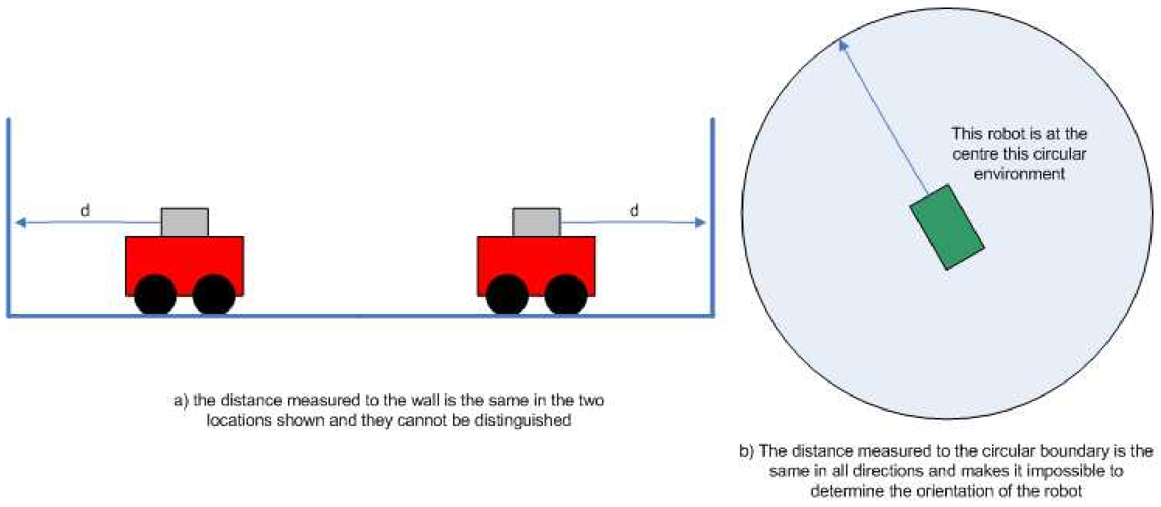

d
d
a) the distance measured to the wall is the same in the two locations shown and they cannot be distinguished
This robot is at the centre this circular environment
b) The distance measured to the circular boundary is the same in all directions and makes it impossible to determine the orientation of the robot

Figure 3.2: Examples of sensor aliasing: In (a) is a 1d aliasing problem where the same distance measurement is obtainable in two locations. (b) shows an aliasing problem in 2d, where again distance measurement is available but it is not possible to determine orientation.

The operating environment of a robot and the detectable objects within it can have a significant influence on the difficulty to localize. Drawing from the example presented in figure 3.2, there is no way for a robot to localize because of the symmetry in the environments shown. In real life fortunately, environments are rarely so symmetrical (especially outdoor), but having similar features in an environment can mean that a robot will not be able to localize itself without having traveled the environment for a distance where it can detect more distinctive features. It would be useful in this case to keep multiple hypotheses for the state estimate, and hence the use of a probabilistic approach again is convenient. Multiple hypotheses can be maintained until distinct features are detected, at which time the number of hypotheses can be reduced. A type of localization method known as active localization aims to control the motion of a robot such that its uncertainty in state estimates is reduced. This more advanced method will usually yield better results compared to a passive method [1, 27], which is used in this thesis.

Moving objects create a dynamic environment in which localization becomes more difficult. This is especially the case if the moving objects are not expected to be present and hence not on the robot's internal map. Unless these dynamic objects can be identified, sensors may mistake them as features that a localization system will try to compare against the map, and lead to wrong inaccurate state estimates.

In some applications, the starting state of the robot is known, in which case the localization problem becomes a tracking problem. A more difficult scenario is where a robot does not know where it is starting, or has been turned off and reactivated at an unknown location (known as the kidnapped robot problem). This is the problem this thesis will address with the localization filter that will be implemented.

The remaining of this chapter will describe the probabilistic approach to localization by first introducing the Bayes filter, followed by its derivatives, the Kalman filter and the particle filter, and then discuss why the particle filter was chosen over the Kalman filter for implementation.

3.1 The Bayes Filter

The Bayes filter is the most general method for state estimation. It is a recursive method, with only two conceptually simple steps. Three inputs are required to generate a state estimation output:

x_{t-1}	the last state estimate
u_t	the motion control inputs
z_t	the latest measurements

All the parameters listed above are random variables with probability density functions associated with them to represent uncertainty. The probability density function associated with the last state estimate can be referred to as the prior probability $p(x_{t-1})$. The new state that is to be estimated can be referred to as the posterior probability $p(x)$ [1, 10].

The first step in the Bayes filter algorithm is to determine the effects of control inputs on the state estimate, given knowledge of the prior probability distribution. This control input propagated (intermediate) state estimate will be represented by x' and mathematically determined using the theorem of total probability as shown in equation 3.1.

$$p(x_t') = \int p(x_t|u_t, x_{t-1})p(x_{t-1})dx_{t-1} \tag{3.1}$$

This prediction step is sometimes informally referred to as the ”act” step. The second step in the Bayes filter involves using sensor measurements to again estimate and correct the believed state by assuming that measurements were made at the intermediate state. This measurement update step is sometimes referred to as the ”see” step, and is expressed mathematically in equation 3.2 in accordance with the Bayes rule.

$$p(x_t|z_t) = \frac{p(z_t|x_t')p(x_t')}{p(z_t)} \tag{3.2}$$

The use of the Bayes rule is important because it is difficult to come up with the conditional probability density $p(x_t|z_t)$. However, it's inverse $p(z_t|x'_t)$ can be modeled as indicated in the previous chapter, and Bayes' rule provides the relationship between a conditional probability density function and its inverse. The conditional probability density describing the new state given the most recent measurements is the posterior probability that is being sought. In the next iteration of the Bayes filter, it will become the prior probability density function that will be used to find the new posterior density function.

Notice that with the Bayes filter, the estimation of the new belief state only depends on the previous estimation, making all previous estimations irrelevant. This is known as the Markov assumption (or the complete state assumption). In practice, the inability to provide complete and perfect models for a robot violates this assumption. For instance, un-modeled interactions with the environment, errors in the map used for localization, or inaccuracies in the probability density function used to model sensor measurements. Even though this is the case, in general the Bayes filter or its derivatives are usually robust towards these violations provided that the effect of not modeling some states is negligible and close to random [1, 9].

The Bayes filter is conceptually straight forward but there is a problem with implementing it in a real robotic system because the mathematical operations of integration and multiplication shown in equation 3.1 and equation 3.2 need to be carried out in unbounded continuous space [10], which makes it difficult to find a closed form solution [1]. One possible solution is to discretize the entire state space of a robot, an approach known as Markov localization [9] which only works well in limited applications due to the computationally intensive nature of this method. Two other solutions to this problem will be presented in this chapter: the Kalman filter, and the particle filter.

3.2 The Kalman Filter

It was previously mentioned that one way of implementing the Bayes filter is to discretize the state space but using such an approach is computationally inefficient, and the computation required would grow exponentially as the size of the state space or the resolution of discretization increases. One approach exists which enables belief states to be determined in continuous space while at the same time remain computationally efficient (an appreciated trait for any program that runs on a mobile robot). This implementation of the Bayes filter is perhaps the most studied method, and it is known as the Kalman filter.

The Kalman filter is able to implement the Bayes filter in continuous space because it assumes a Gaussian distribution for all the probability density functions used by the Bayes filter. In addition, it assumes that the system to which the Kalman filter is applied is linear [1, 10, 9]. More formally, three requirements must hold for the basic Kalman filter so that the act and see steps of the Bayes filter can be carried out in continuous space.

The first rule is that the state transition function which models the effects of motion control inputs must be linear with Gaussian noise, which can be expressed according to equation 3.3 [1].

$$x_t = A_t x_{t-1} + B_t u_t + e_t \tag{3.3}$$

where, x_{t-1} is the previous belief state,

x_t is the control propagated belief state,

u_t is the control input,

A_t, and B_t are coefficient matrices

e_t is the zero mean Gaussian noise

A Gaussian distribution is defined by two parameters: the mean and the variance (or standard deviation). Following the state transition function (equation 3.3), the mean of the propagated state belief is shown in equation 3.4 where μ represents the mean of a Gaussian distribution.

$$\mu'_t = A_t \mu_{t-1} + B_t u_t \tag{3.4}$$

The variance of the propagated state can also be easily computed according to equation 3.5. Here, the covariance of the control input is assumed to be constant [28].

$$\Sigma'_t = A_t \Sigma_{t-1} A_t^T + R_t \tag{3.5}$$

where, Σ'_t is the covariance of the propagated belief state,

Σ_{t-1} is the covariance of the prior belief state,

R_t is the covariance of the control noise

The second requirement for the basic Kalman filter again is a linear system restriction, but this time the requirement refers to the measurement model, which can be expressed according to equation 3.6 [1]. Similar to the control input covariance, the covariance associated with the measurement noise is assumed to be constant [28]

$$z_t = C_t x_t + \delta_t \tag{3.6}$$

where, z_t is the measurement vector,

x_t is a state vector,

C_t is a coefficient matrix,

δ_t is the zero mean Gaussian measurement noise

The third requirement for the Kalman filter is that the initial belief state has to follow a Gaussian distribution [1].

The prediction step of the Bayes filter is implemented through the Kalman filter using equations 3.3 to 3.5, and now it is necessary to address the second half of the problem where measurement data is used to update and correct the belief state. For this, an important matrix known as the Kalman gain needs to be introduced. This matrix can be calculated using the parameters of the measurement model and the covariance of the control input propagated belief state as shown in equation 3.7.

$$K_t = \Sigma_t' C_t^T (C_t \Sigma_t C_t^T + Q_t)^{-1} \tag{3.7}$$

where, K_t is the Kalman gain,

Σ_t' is the covariance for the propagated belief state,

C_t is the coefficient from the measurement model,

Q_t is the covariance for the measurement noise

The Kalman gain can be regarded at as a measure of confidence in the measurement data and therefore how much weighting the measurements will have when the information is used to determine the new belief state. The calculations involved in determining the mean (μ_t) and covariance (Σ_t) of the new state estimate is shown in equation 3.8 and 3.9.

$$\mu_t = \mu_t' + K_t(z_t - C_t \mu_t') \tag{3.8}$$

$$\Sigma_t = (I - K_t C_t)\Sigma_t \tag{3.9}$$

where, I is the identity matrix

In equation 3.8 the expression $(z_t - C_t\mu'_t)$ is often referred to as the innovation, which is the difference between the real and expected measurements [9].

The detailed mathematical derivations for the formulation of the Kalman gain, and the use of the Kalman gain to calculate the mean and covariance of the new belief state can be found in [1, 10].

Since the Kalman filter starts with a Gaussian random variable and only applies linear transformations to it, the resulting output will maintain a Gaussian distribution, and this will remain the case as the algorithm iterates. The ability to calculate new state estimates in closed form makes it very efficient and highly valued in mobile robotics, but there are drawbacks to the algorithm.

One of the problems is the assumption of linear state transition and measurement models, which may be rather optimistic in the real world. More advanced Kalman filters get around this problem by trying to linearize the non-linear transition and measurement models. The extended Kalman filter is one such example where a first order Taylor expansion is used to approximate the non-linear system [1, 10]. Another variant is the unscented Kalman filter which performs stochastic linearization [1].

A problem that is more severe in comparison to dealing with non-linear systems comes from the Kalman filter's inherent dependency on Gaussian distributions. While the distribution is generally adequate in describing control and measurement noise that might be encountered, it is not possible to express multiple hypotheses for state estimation because a Gaussian distribution is uni-modal. Consider the problem presented by this thesis, where it is required for a robot to localize itself without knowing its starting state. A uniform distribution would be more appropriate for the starting state. As the localization process iterates, ambiguities may cause the belief state's probability density function to have local maxima at several locations. In such a case, it is important to keep track of these high density regions but this can not be done with a Gaussian distribution. Therefore, a Kalman filter may be very suitable for a robot tracking problem (where the initial state is known) but it is inappropriate for the kidnap robot problem (where the initial state is unknown).

3.3 The Particle Filter

Two implementations of the Bayes filter for localization have been discussed so far but each presents its own problems. Markov localization is able to represent any form of probability distribution in the entire state space but its computational requirement makes it impractical in many applications. The Gaussian filter is computationally efficient but lacks the ability to represent a belief state estimation in anything other than a Gaussian distribution. What is needed is an implementation of the Bayes filter that is computationally practical and at the same time capable of representing any belief state probability distribution.

The particle filter is an implementation of the Bayes filter using a finite number of parameters to describe the belief state distribution. These parameters are the particles, and they are sampled from the state space according to the belief state probability density distribution. The spatial density of the particle set reflects the shape of the probability density function that it is sampled from, and hence the particle set is an approximation of the belief state probability distribution. A high density of the particles in the state space reflects the high likelihood of the true state in the vicinity [1]. An interesting point to note is that in the limiting case where the number of particles used to sample the belief state distribution approaches infinity, the particle filter will operate as if it is dealing with probability distributions in continuous space [29].

A benefit of the particle filter that should be apparent is that it works for any probability distribution. This implies that it is able to handle the kidnap robot problem, where the initial state of a robot is unknown. It also enables the particle filter to track multiple state hypotheses. Another advantage in using the particle filter is its ease of implementation.

The first step in implementing a particle filter is generating a set of particles that is representative of the starting state. A random sample from a uniform distribution can be used if nothing is known of the starting state. On the other hand, if the exact starting state is known, all the particles can be initiated in the same state (thus occupying the same point in the state space).

Both the "act" and "see" steps of the Bayes filter are still carried out with the particle filter. The effects of motion control inputs (with noise) are applied to all particles using a state transition model [10]. Any type of function (linear or non linear) can be used in this case provided that it generates a propagated state vector based on the input of a previous state vector and a control input vector. The propagated particle set is now a sample of the motion control propagated belief state distribution as expressed in equation 3.10 [1, 30, 31].

$$x_t^{[m]} \sim p(x_t | x_{t-1}^{[m]}, u_t) \tag{3.10}$$

where, m is the particle index,

x_t and x_{t-1} are the output and input state vectors respectively,

u_t is the control input vector

To take into account sensor measurements, an importance factor (weighting) is assigned to each particle based on the probability density function associated with the measurement model as shown in equation 3.11[1, 10, 30, 31].

$$w_t^{[m]} = p(z_t | x_t^{[m]}) \tag{3.11}$$

where, $w_t^{[m]}$ is the importance factor for particle m,
z_t is the sensor measurement vector,
$x_t^{[m]}$ is the state vector for particle m

Particles that are situated in locations in the state space where the expected measurements (obtainable by referring to a map) and the real measurements are similar should receive a high importance factor.

The importance factors are used to resample the particle set. In this step, a new particle set with the same number of particles as the current particle set is generated. Particles with a high importance factor are more likely to be sampled (maybe more than once). On the other hand, particles with a low importance factor may not be sampled and fail to reach the next iteration of the particle filter. The new particle set generated by this survival of the fittest technique is representative of an approximation to the (posterior) state estimation after considering sensor measurements [1, 10, 30, 31]. In the next iteration of the particle filter, the posterior particle set will represent prior belief state.

The way in which the particle filter handles state transition and incorporation of measurements to update the belief state makes it suitable for use with non-linear models. A drawback of the particle filter is that it can still be computationally expensive depending on the number of particles used [30]. There is no set rule for determining how many particles should be used for different types of problem, and this is something that needs to be tuned during application. Generally, a large state space requires more particles than a smaller one to ensure that the belief state distribution is represented properly. Too few particles may lead to a high chance of the particle deprivation problem occurring, where through resampling, particles near the true state does not make it to the next state estimate iteration by chance [1].

Overall, the particle filter is more applicable to the localization problem under investigation, where the starting state is unknown. Additionally, the use of a camera as the sensor and extracting image features leads to a complicated and non-linear model which a particle filter can deal with. The details on the implementation of the particle filter for the localization problem will be presented in a later section, but before that it is necessary to generate a map as the reference for localization, and determine how to extract the useful features from images captured by the on board camera.

Chapter 4

Feature Map Generation

In the previous chapter, it was identified that a map is needed as part of the localization process. It was also mentioned that there are two types of map: the feature based map which is a list of objects and their properties, and the location based map where objects are recorded by locations they occupy. In the vision based localization problem being investigated, it would be ideal if the perceived information from the camera is in a representation that is similar with the map so that they can be compared in the localization filter. In computer vision, a raw image containing low level information (raw data) is usually processed to highlight the information of interest. Other operations are performed afterwards to obtain a high level representation or features. Therefore, it would be appropriate to use a feature based map for vision based localization.

In the current investigation, an aerial image of the operating area is the only information of the environment that is given to the localization system. Aerial images are resources that are becoming more readily available, and they have been used as resources for geographic information systems (GIS). Similar to an image captured with a camera, it contains low level information which needs to be interpreted and processed so that high level features can be retrieved.

The aerial maps that will be used in the localization system are orthoimages, which are images derived from normal perspective images in a way such that displacements caused by sensor (camera) placement and relief of terrain are removed. These high resolution images are in the format of grayscale bitmaps, where $10cm$ in real life resolves to approximately 1 pixel length. A bitmap is basically an array of intensity values, where the value at every pixel is defined. The depth of the bitmap is 8-bit, therefore, a pixel can hold one of 256 (2^8) values from 0 to 255. Figure 4.1 is an example an aerial image. All the aerial images used in this research are properties of the Regional Municipality of Waterloo, Ontario, Canada.

A robot will benefit most if a map contains information about obstacles in the surrounding so that they can be avoided while the robot navigates. Therefore, high level features of interest are the boundaries of buildings or any large objects on the ground. This however is not an easy task due to the complexity of outdoor environments and how they appear on an image.

The goal of extracting boundary features from aerial imagery is similar to the practice of building detection. Automatic detection of buildings from aerial images is of great interest in many GIS related fields such as cartography, urban mapping, urban planning, land use analysis, and geo-information engineering [32][25][33]. In general there are three ways to approach the building detection problem: stereo vision, line analysis, and using auxiliary information. However, there is unlikely a method that can perfectly detect all buildings in every aerial image. In line analysis approaches, a rectangular building model or one that consists of several rectangles is usually assumed as a hypothesis and used for interpreting groups of detected features [34]. Lines are considered appropriate since most man made structures are rectangular or contain

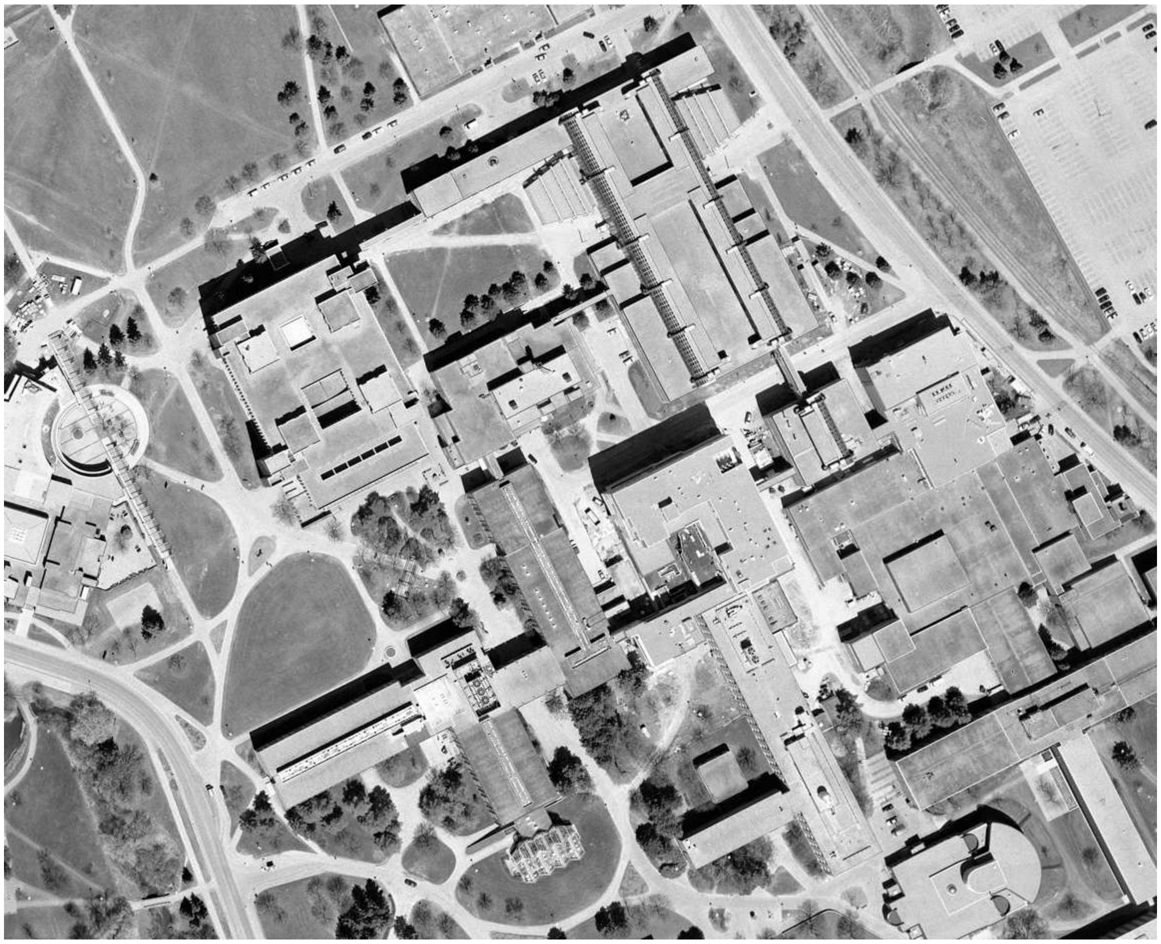


Figure 4.1: An aerial image showing the central portion of University of Waterloo campus

mostly straight edges. Shadow analysis is an example of using auxiliary information. A building model is also assumed for this approach along with information on the source of illumination (or sunlight) [25]. Most building detection methods start with low level image processing methods of edge and line detection. The end result is very much influenced by how well relevant low level information are extracted or filtered for further use. The problem of building detection is made difficult with the presence of shadows, surface markings, vegetation, and other distractions which may add unwanted boundary lines or fragmented boundaries of interest. These effects together are known as the figure-ground problem, and it has a much more significant impact compared with sensory noise [32]. Having obtained the low level features, higher level pattern recognition techniques and graph search on connected line features is applied to interpret the presence of closed contours or edges that may belong to a building [25].

The sections in this chapter will sequentially explain the aerial image feature extraction process which is summmarized in figure 4.2. The first step is the processing of raw data using edge detection techniques to highlight the pixels likely to be part of building boundaries in an edge map. This is followed by the removal of edge pixels from the edge map that were produced by shadows instead of true building boundaries. The edge map is further filtered by masking edges that may have come from vegetation or other distractions in the aerial image. High level line segment (boundary) features are identified using a special version of the Hough Transform and its algorithm will be reviewed along with the standard Hough Transform. In the closing sections of this chapter, it will be shown how the high level features are filtered and the feature extraction results will be shown for the area covering the localization filter test site.

4.1 Edge Detection

Buildings on an aerial map can be identified by the way they appear compared to everything else on the ground. More specifically, boundaries of buildings in a grayscale map can be spotted where the image intensity changes, and these are known as edges in image processing. Edge detection is a common operation used on images containing intensity (low level) data so that information of interest (or edges) are highlighted to provide a higher level of image understanding.

Edges in an image occur when there is a sharp change in local intensity in the

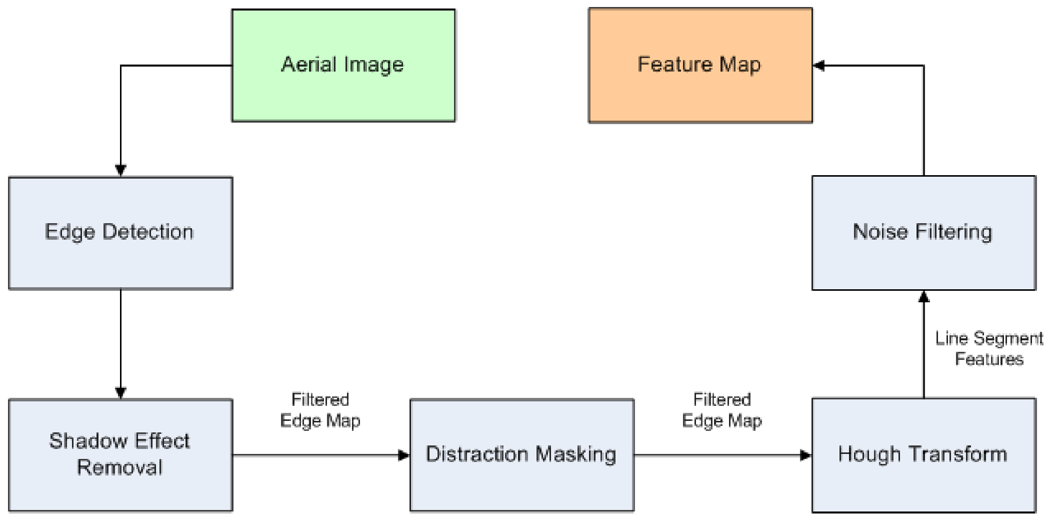


Figure 4.2: An overview of the feature map generation process

spatial domain of the image. Another way of defining edges is by considering an image to be a signal (by performing a Fourier transform on the image), where edges are represented by high frequency signals in the spatial-frequency domain. Image processing researchers have come up with many different approaches to edge detection in the past. For spatial domain approaches, a kernel based on difference equations is usually used to convolute the image. Such a kernel is actually an approximation to taking the derivative of image intensity, and the response to such a kernel will be high if an edge exists. Some well known kernels that use a first order approximation for image intensity derivative include the Roberts Cross, the Prewitt operator, and the Sobel operator [23]. Perhaps the most well known operator is Canny edge detector [35].

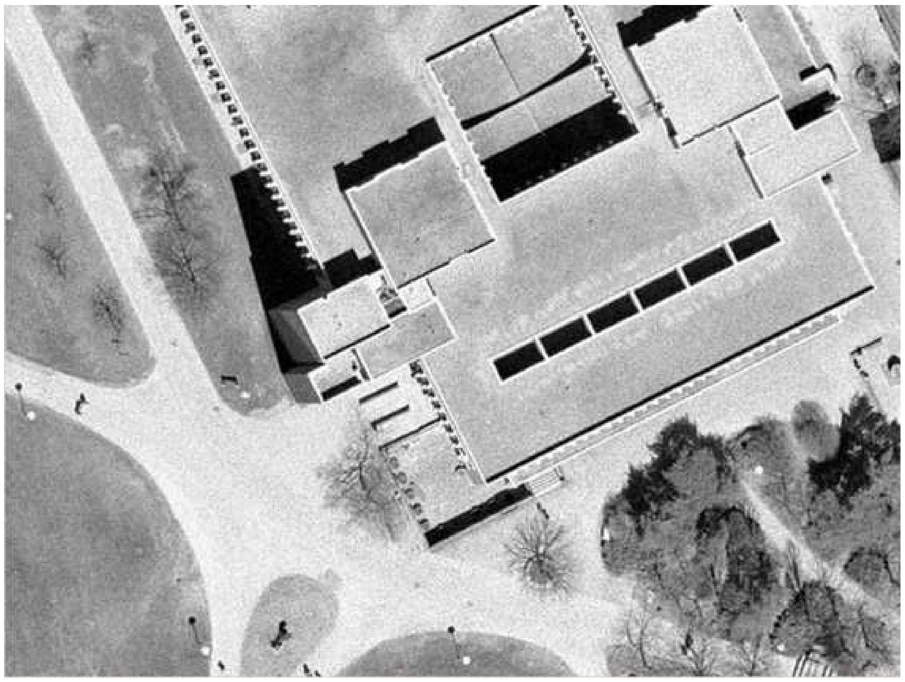
The Canny edge detector is considered optimal as it is able to maximize the signal-to-noise ratio (SNR) of the intensity gradient so that as many true edges are identified as possible while minimizing false edges. Additionally, the Canny edge detector can accurately localize a detected edge so that the error in location between the true edge and detected edge is minimized. Furthermore, the Canny edge detector is able to minimize multiple responses to a single edge so that an edge is only identified once.

The Canny edge detection algorithm consists of four steps: noise reduction, intensity gradient calculation, non-maxima suppression, and double thresholding. First, a grayscale input image is usually smoothed with a Gaussian filter to reduce noise. The magnitude of the local intensity gradient and the gradient's direction are then calculated (commonly using the Sobel operator). In the non-maxima suppression stage, a pixel is discarded if a neighbouring pixel at a perpendicular direction to the intensity gradient direction has a greater gradient magnitude. In the final step, double thresholding is performed to generate the binary edge response image. Here, two thresholds are defined with the upper threshold at approximately 1.5 times the lower threshold. Pixels with an intensity gradient magnitude greater than the upper threshold are automatically defined as edges. Pixels with gradient magnitudes less than the upper threshold but greater than the lower threshold are defined as an edge if it is adjacent to another edge pixel [23]. The end result is a binary edge map.

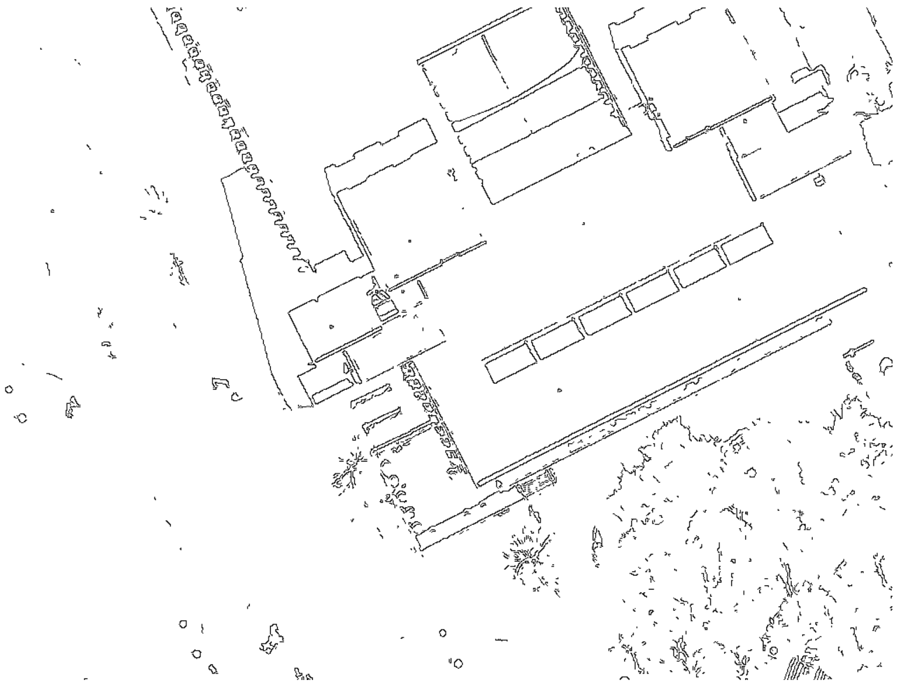
For the vision-based urban localization problem being studied, the Canny edge detector will be used on aerial images to highlight and extract the boundaries of interest to an edge map. An example of an image that has been processed by the Canny edge detector is shown in figure 4.3.

4.2 Shadow Edge Removal

The strong gradient that exists between an area overcast by shadow and an adjacent area that is not shaded implies that it is inevitable that the edge of a shadow will induce a response from the edge detector, which can be seen in parts of figure 4.3. This means that during the feature extraction process, shadow edges will appear as



(a) Original image

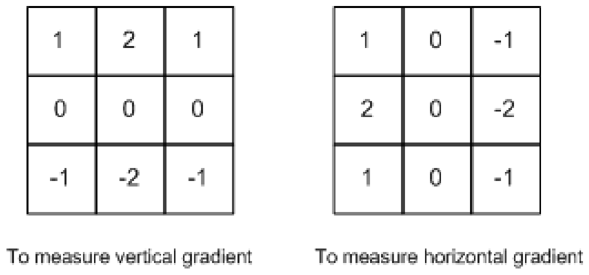


(b) Processed image - the edge map

Figure 4.3: Canny edge detection on an aerial image

building boundaries.

Shadows can be easily identified in an aerial image as they appear much darker (or have a much lower intensity) compared to all other objects on an aerial image. This distinctiveness makes it easy to segment shaded areas from an aerial image. The Canny edge detector can again be used to determine where the edges of shadows lie. However, to correctly remove the effects of shadow, it is necessary to distinguish whether a shadow edge is shared with a building boundary (which should not be removed), or shared with the ground (which should be removed). One way of doing this is by taking into account the source of illumination, or the direction of sunlight. The intensity gradient over an edge of a shadow calculated using Sobel operators (shown in figure 4.4) can be used to speculate whether an edge should be part of a building. If this is the case, the intensity gradient should be decreasing in the direction of the illumination (away from the light source). If this is not the case, the shadow edge is likely not shared with a building and thus can be eliminated.



1	2	1
0	0	0
-1	-2	-1

To measure vertical gradient

1	0	-1
2	0	-2
1	0	-1

To measure horizontal gradient

Figure 4.4: The Sobel operators

It must be acknowledge that more advanced techniques of shadow detection exist in model-based building detection research (such as the method presented in [36]), but the method used is adequate for generating the maps required for the localization method developed in this project.

4.3 Distraction and Vegetation Masking from the Edge Map

Edge detection has allowed pixels that are on the boundary of a building to be highlighted. However, aside from shadows, pixels from other irrelevant objects and distractions have also been highlighted by the edge detection process as evident in figure 4.3(b). These unwanted edge responses need to be removed before trying to extract higher level information from the image to prevent false features from being identified as a building boundary.

Many of the distraction edge responses come from vegetation in the environment. Image intensities in vegetation areas as well as other distraction areas vary in a way that give the appearance of rough texture. It is desired to find these regions and mask the edge responses that appear within so that they will not be considered as building boundaries. The use of Gabor filters is an option as it has been proven to work well in texture segmentation in images [37, 38, 39, 40]. However, classification of textures is computationally very expensive since the number of Gabor filters required corresponds to a pattern recognition problem with a high number of feature dimensions. A variety of other texture analysis methods exist [41], but a much simpler method will be used instead.

The rough and random appearance of the texture in vegetation areas as well as other distraction objects indicate that image intensity is changing in multiple directions. In the process of detecting building boundaries which appear as intensity change in a single direction, the edge detector inevitably also responded to the intensity gradients from distractions. Therefore, it is proposed that pixels showing gradients in multiple directions be masked. In image processing, these pixels are known as corners, and so the corner response measure will be used to discriminate vegetation and other distractions.

A pixel can be labeled as a corner if it has a sharp change in gradient in two orthogonal directions. However, theses two directions does not necessarily have to coincide with the axes of an image and therefore, strictly computing the intensity gradient or the derivatives in the image axes directions (x, y) is inadequate. The Harris corner detector [42] is able to provide a measure of corner response using only the x and y derivatives $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ in the intensity gradient covariance matrix M shown in equation 4.1.

$$M = \begin{bmatrix} \sum \frac{\partial I}{\partial x}^2 & \sum \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum \frac{\partial I}{\partial y}^2 \end{bmatrix} \tag{4.1}$$

The elements in M are sums of derivative products within a pixel of interest's neighbourhood (kernel). The diagonalized version of M can be determined by finding the eigenvalues (λ_1, λ_2) of the matrix [43]. These eigenvalues correspond with the gradient strength in the principal directions defined by the corresponding eigenvectors. A corner is found if both eigenvalues are greater than zero, indicating that there are strong changes in intensity in two directions. If one of the eigenvalues is equal to or close to zero, it indicates that a strong intensity gradient is found in one direction and it can be concluded that only an edge is discovered. In the case where both eigenvalues are approximately equal to zero, it indicates that the local intensity is constant or changing very slowly. As an alternative to evaluating both eigenvalues, they can be considered together by defining a corner response measure R, which expresssed in equation 4.2.

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{4.2}$$

In equation 4.2, k is a tunable constant usually set between 0.04 to 0.06. However, since evaluating eigenvalues is computationally expensive, the above expression can be rewritten as equation 4.3 using the trace and determinant of M.

$$R = |M| - k(tr(M)^2) \tag{4.3}$$

A corner is considered found if R is greater than zero. In practice, a corner response threshold is usually set and corners are labeled at all local maximums in a corner response image. An edge is found if R is less than zero, and a R value of zero indicates constant intensity.

It was found that most distractions that appear in an aerial image to be within a certain range of corner response values as shown in figure 4.5. In this case, the constant k from equation 4.3 was set to equal 0.05. The original version of this image has been shown previously in figure 4.3(a). From the corner response image, an edge response mask is created by identifying pixels where the corner response value is between 0.00001 and 0.001.

The resulting corner response (distraction) mask at this point contains some small gaps and point noise that need to be eliminated. This will be accomplished using morphological operators. These operators are used frequently in image segmentation where there is a need to merge or split regions or boundaries. Morphology is a mathematical based set-theory image processing technique mostly applied to binary images. All morphological operations involve an image with pixels of interest (in this case they are the masking pixels of the corner response filter). These pixels will be identified to belong to set A. In addition, a structuring element that can take on any shape is required and pixels from this can be labeled as belonging to set B. Likewise, pixels belonging to the resulting morphological operation is labeled as belonging to

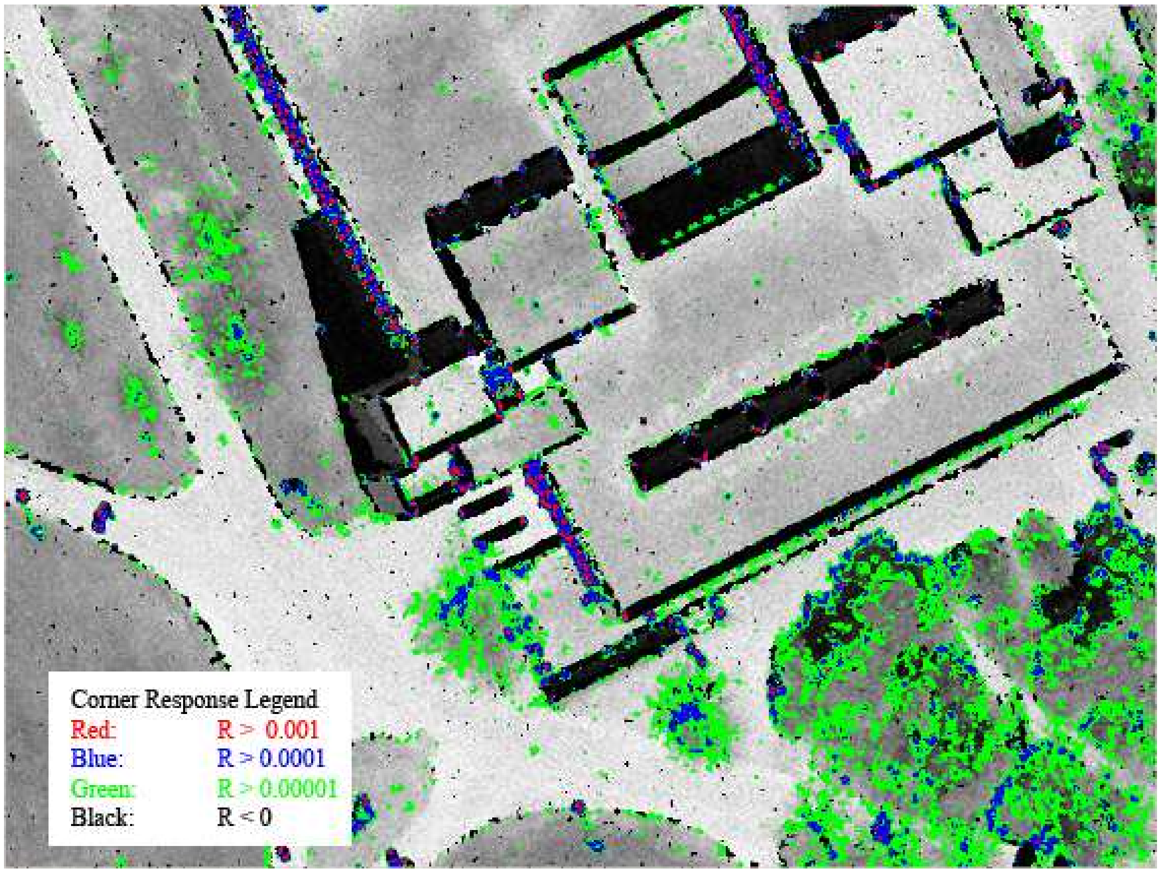


Figure 4.5: The corner response of an aerial image

set C. The two basic morphological operators are dilation ($\oplus$) and erosion ($\ominus$) [44]. If a, b, and c represent image coordinates of pixels in sets A, B, and C respectively, then the two operators can be mathematically expressed as equations 4.4 and 4.5.

$$A \oplus B = \{c | c = a + b, a \epsilon A, b \epsilon B\} \tag{4.4}$$

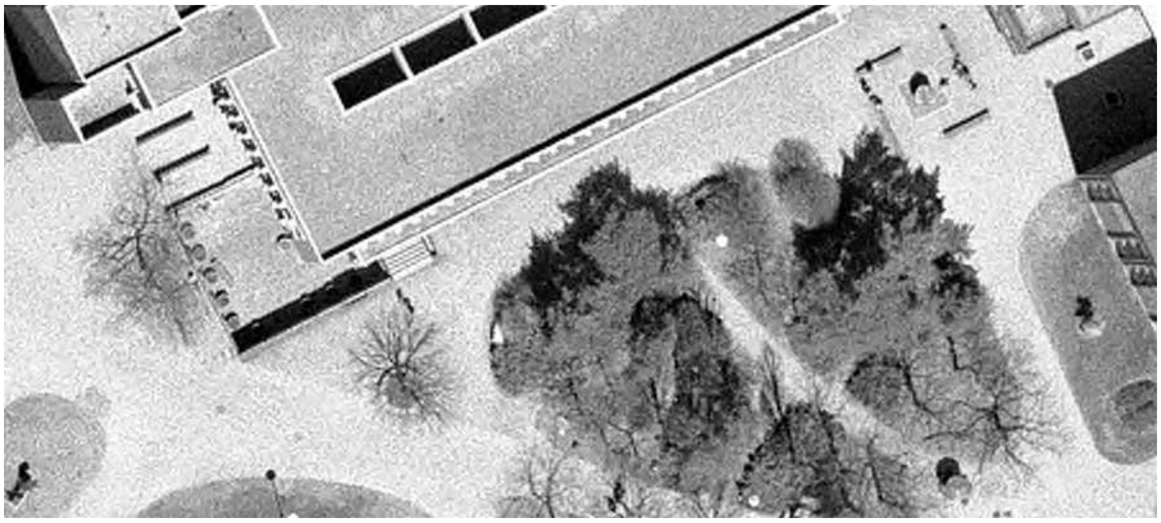
$$A \ominus B = \{c | c + b \epsilon A, b \epsilon B\} \tag{4.5}$$

With the dilation operator, a pixel that contains the value 1 will cause all neighbouring pixels to hold the same value. In erosion, a pixel will be set to 0 if any neighbouring pixels hold the value of 0.When dilation is followed by erosion, the combined operation is known as closing. Similarly, if erosion is precedes dilation, the combined operation is known as opening. Closing operations are applied to the corner response mask to remove small gaps over vegetation and distraction areas, and figure 4.6 is an example of the resulting mask. This mask image will be applied to the edge map from the Canny edge detector using the binary *and* operator, causing any edge responses under the mask to disappear. Figure 4.7 is another example of the masking operation over the area where the particle filter localization system will be tested. Further examples of this masking operation can be found in appendix A.

4.4 The Hough Transform

Edge detection allowed building boundaries from an aerial image to be highlighted. However, the boundaries are still no more than a collection of pixels in a binary image and a higher level representation such as simple geometric entities is still seeked. Straight line segments are appropriate features for representing the information in the aerial image derived edge map since building boundaries in an urban environments are often straight. The Hough transform (HT) is a powerful tool in computer vision used for shape detection. It was first introduced as a method for detecting complex patterns of points in a binary image [45]. The idea behind the HT is to take a spatial domain pattern consisting of pixels that may be spatially spread out and transform it into a parameter space where the original pattern can be identified as a spatially compact feature [46]. As a simple example, consider the detection of straight lines. In an image, one mathematical representation of a line in Cartesian coordinates using the parameters θ and r is expressed in equation 4.6 and illustrated in figure 4.8.

$$x \cos\theta + y \sin\theta = r \tag{4.6}$$


(a) Original image

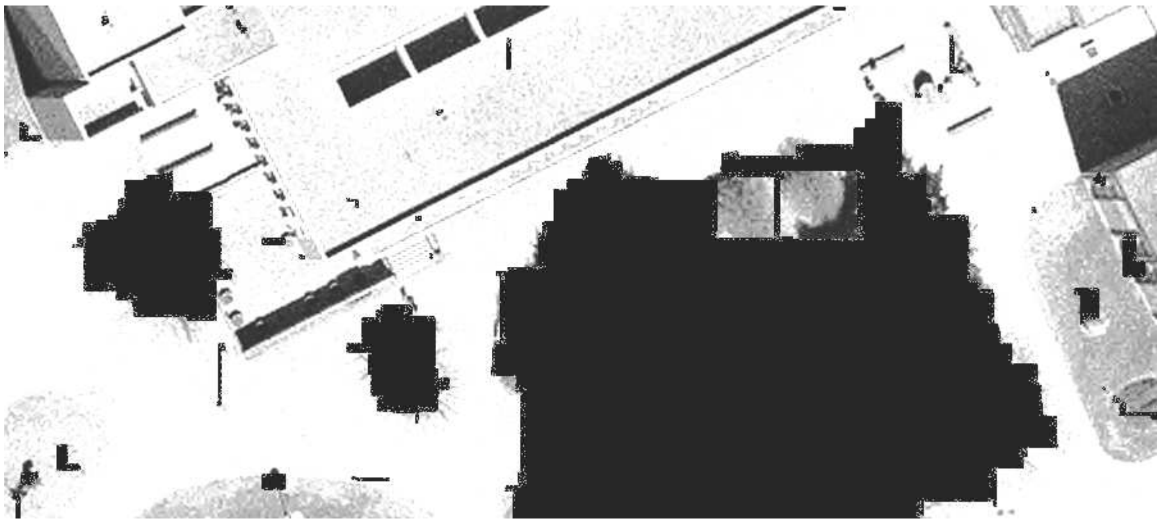
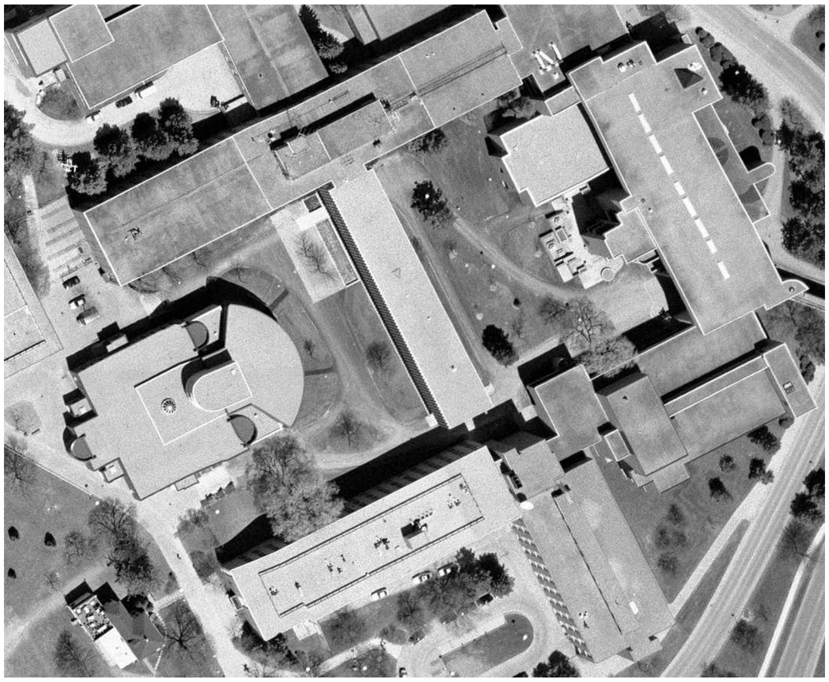
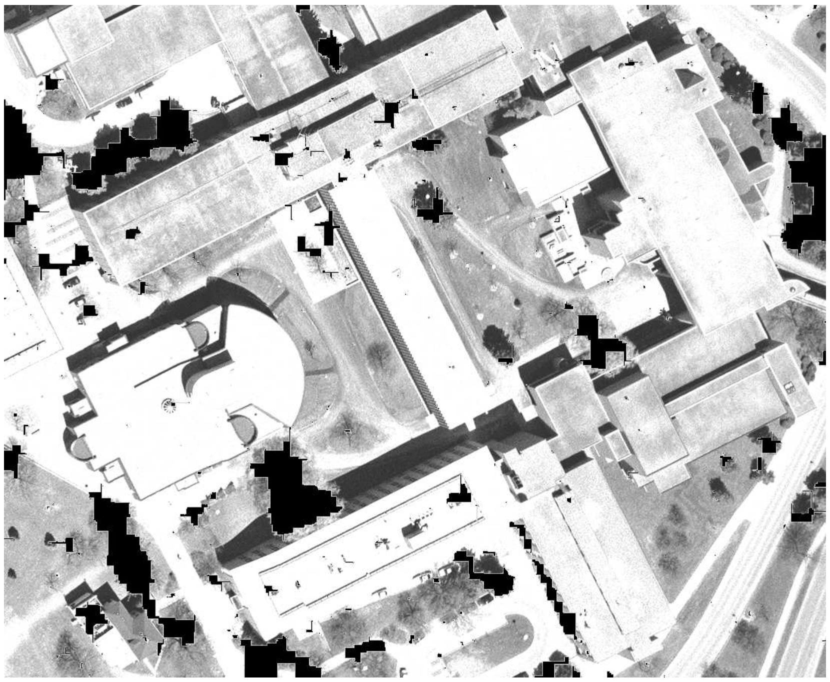

(b) Original image superimposed with the mask

Figure 4.6: The corner response / distraction and vegetation mask

In this representation, r is the length of the perpendicular of the line that runs through the origin, and θ specifies the angle of the perpendicular. A line that passes through a point (x, y) in Cartesian space is represented by a sinusoidal curve in parameter space, and a point in parameter space represents a line in Cartesian space. Therefore, it can be expected that a line that passes through many points in Cartesian space will be the point (or a spatially compact feature) where many sinusoidal curves



(a) Original image



(b) Original image superimposed with the mask

Figure 4.7: The corner response mask over the localization system testing area

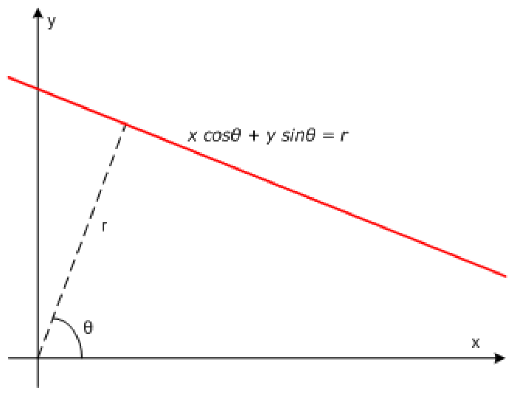


Figure 4.8: The normal form for the equation of a line

in parameter space converge upon as shown in figure 4.9.

In practice, to perform the HT, the parameter space is discretized into accumulator cells in which a vote will be cast if a curve passes through the cell. A line is considered detected if the vote in an accumulator cell is greater than a threshold value. Overall, the HT makes it considerably easier to detect extended point patterns in image space [46][45].

Many variation of the HT exists, and the Progressive Probabilistic Hough Transform (PPHT) [47][48] is a variation that falls into the Monte Carlo (or probabilistic) class of HT methods. The objective of probabilistic HT (PHT) class is to achieve the same detection result as a standard HT (SHT) method using only a subset of points from the input binary image [49][50]. Therefore, PHT methods are considerably faster than the SHT and are suitable for real-time applications [47]. Three threshold values are required for this algorithm: the accumulator threshold, the gap threshold, and the minimum length threshold. For a flowchart of the PPHT algorithm, refer to figure 4.10. As a brief outline, the algorithm starts by randomly selecting a pixel and removing it from the input image while updating the accumulator space accordingly. Pixels are continually sampled until the highest peak in the accumulator is greater

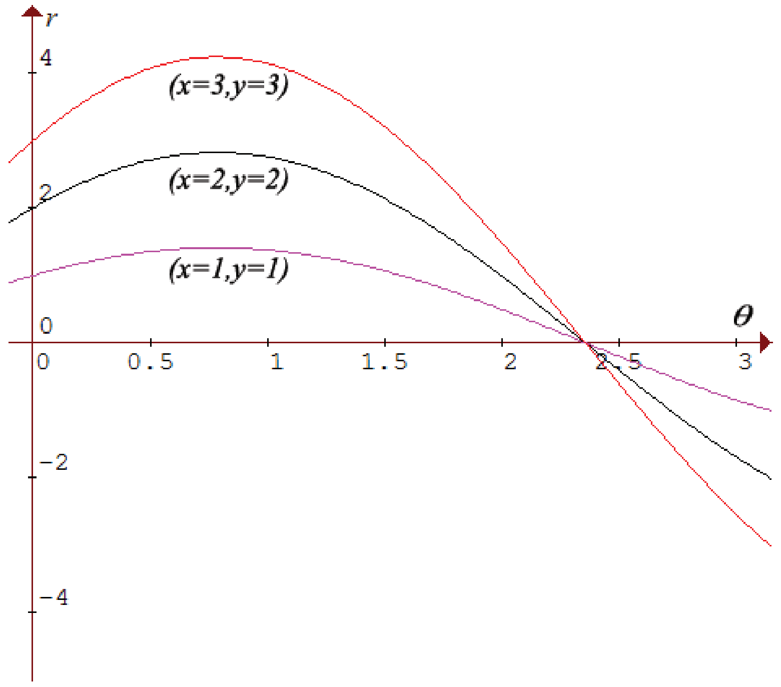


Figure 4.9: An example showing the basic concept behind the Hough Transform - Three points are sampled from a line in Cartesian space and are shown as three sinusoids in parameter space. The three curves intersect at a single point, leading to the conclusion that the three sampled points belong to the same line in Cartesian space with the equation $x \cos 3\pi/4 + y \sin 3\pi/4 = 0$ or $x = y$.

than a threshold. At this point, the longest line segment corresponding to this peak is found in image space. Small gaps are allowed in this line segment provided that their size is less than the gap threshold. All the pixels belonging to the line segment are then removed from the image and the corresponding votes in the accumulator from those pixels are also eliminated. As a last check, the line segment found is only labeled as a detected line segment if it is longer than the minimum length threshold (in Cartesian space). The entire process iterates until the input image is empty. For additional information regarding the PPHT algorithm, refer to [47][48].

An algorithm extended from the PPHT algorithm will be used for extracting line

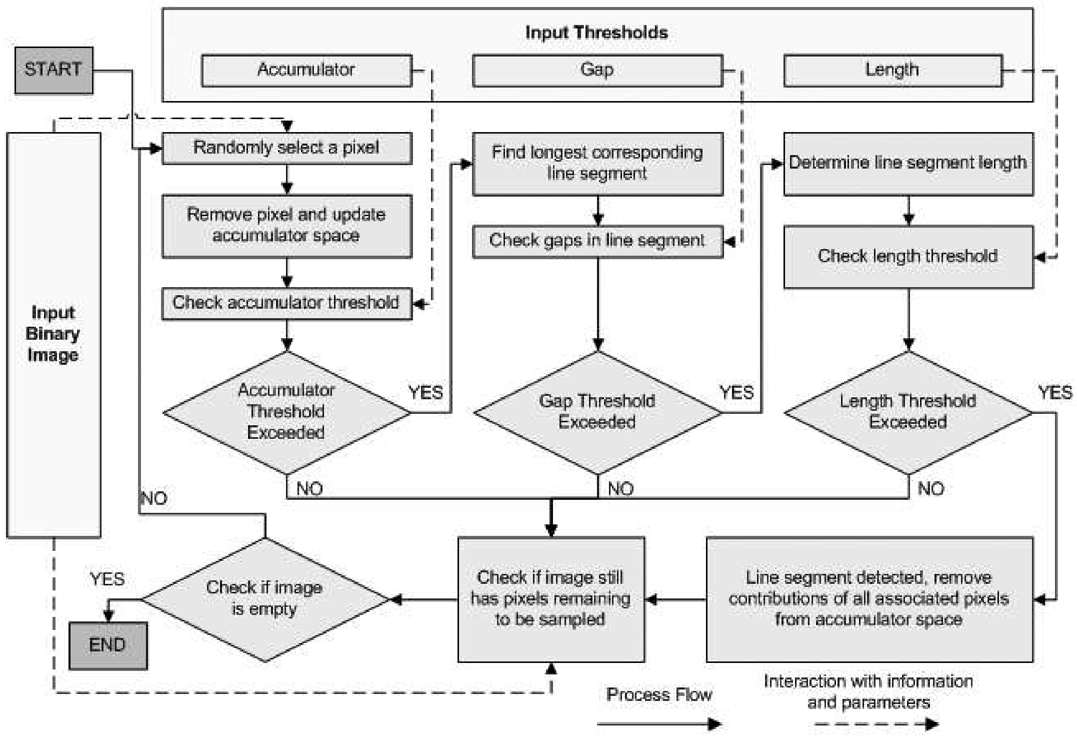

Input Thresholds
Accumulator
Gap
Length
START
Randomly select a pixel
Remove pixel and update accumulator space
Check accumulator threshold
Find longest corresponding line segment
Check gaps in line segment
Determine line segment length
Check length threshold
Input Binary Image
Accumulator Threshold Exceeded
Gap Threshold Exceeded
Length Threshold Exceeded
YES
YES
YES
NO
NO
NO
NO
YES
Check if image is empty
END
Check if image still has pixels remaining to be sampled
Line segment detected, remove contributions of all associated pixels from accumulator space
Process Flow
Interaction with information and parameters

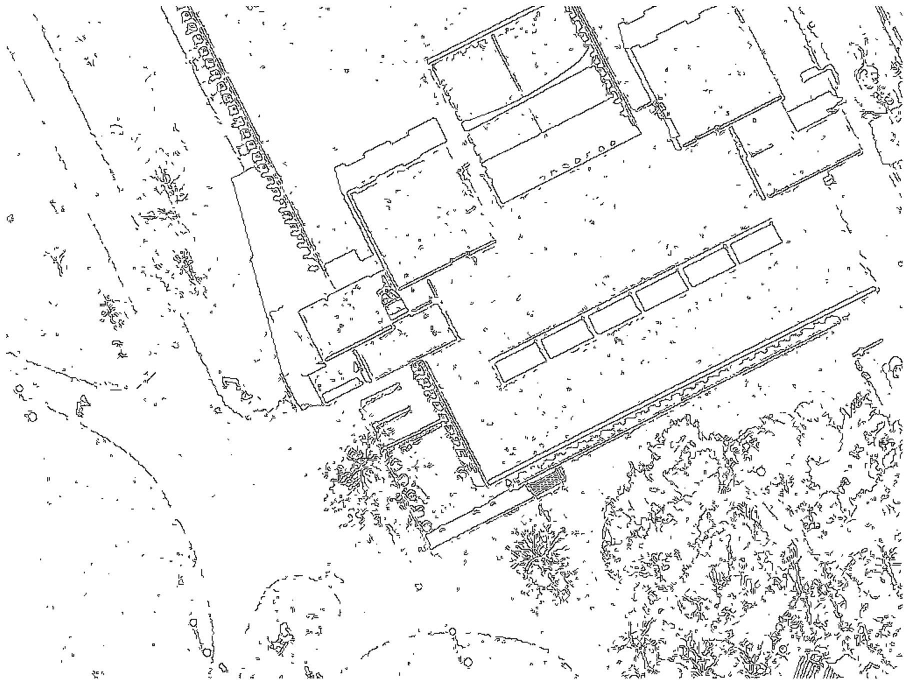
Figure 4.10: The Progressive Probabilistic Hough Transform (PPHT) Algorithm

features from aerial image edge maps. Besides having the benefit of being more efficient than the SHT, features returned by the PPHT are line segments. This is beneficial because not only is the PPHT able to indicate the pose of a line (which is the output from the SHT), the PPHT is also able to indicate its endpoints, which is important in defining the building boundary features on the map. The drawback in using any HT algorithms is that it can only detect shapes that it is programmed to find. Therefore, it is impossible to detect the boundary of a building if it is not a straight line and in a shape that can not be parameterized. Fortunately, most man made structures contain straight boundaries. Figure 4.11 is an example of an edge map that is processed by the PPHT transform. Three observations should be made from this figure; first, the edge maps were not filtered with the corner response mask and this has led to many line segments detected in vegetation and distraction areas. Hence the necessity to perform the corner response filtering is shown. The second observation to make is that on building boundaries, many line segments are extracted and this makes the feature map look messy or "hairy". Therefore, further processing of the feature map is required to reduce the many overlapping line segments. The third observation is not so obvious, and it is that the line segments extracted depends on quality of the input edge map. Since these edge maps are results of the Canny edge detection process, it implies that the parameters used in edge detection will affect the line segments extracted. It is difficult to say what edge detection settings work best with the HT process to extract the best feature map. Therefore, it would be appropriate to evaluate edge maps generated with different settings. This process, along with augmentations to the PPHT that will make the resulting feature map less "hairy", will be introduced in the next section.

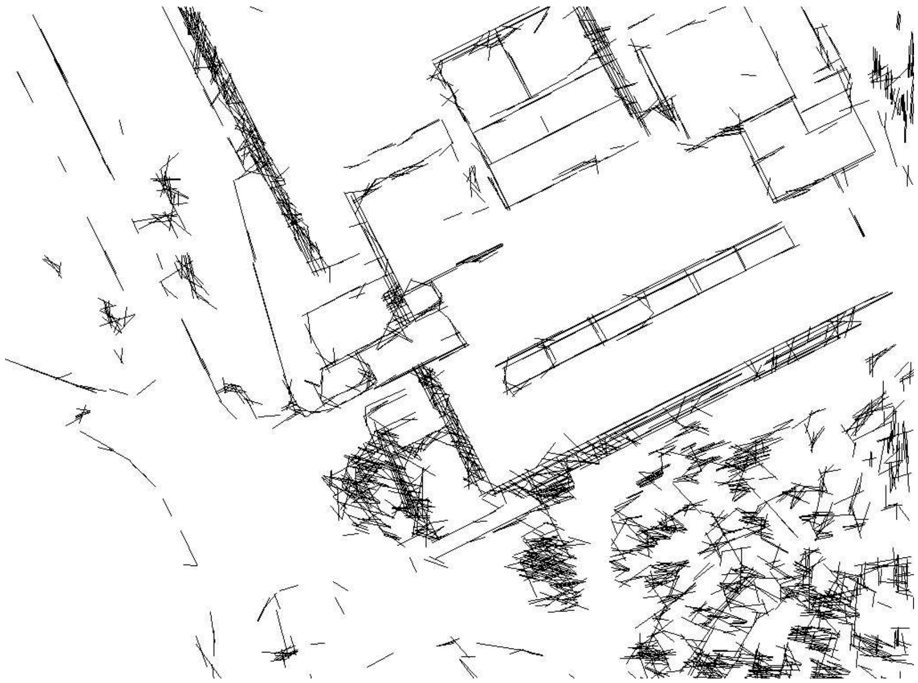
4.5 Modified Progressive Probabilistic Hough Transform

The PPHT was introduced as an effective algorithm for identifying line segments (or other simple geometric shapes) from a binary edge map. However, using this algorithm on an aerial image derived edge map (whether filtered by the corner response or in the unaltered state) yields a resulting feature map where many line segments overlap each other on a building boundary and causes the overall feature map to look messy.

The cause of this problem is partly due to the detail and resolution of the aerial image, and on the nature of the PPHT. High image detail allows details of physical objects near or on a building boundary to be shown. For instance, the ledge on the



(a) Input edge map



(b) Output feature (line segment) map

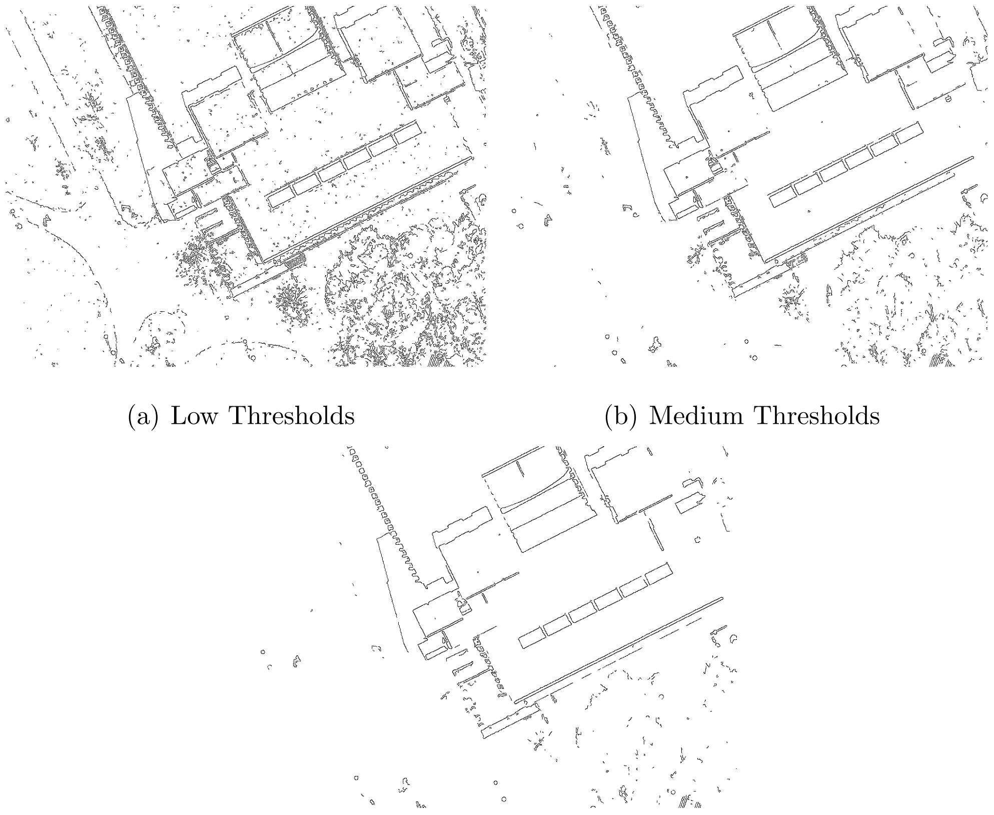
Figure 4.11: The corner response / distraction and vegetation mask

roof of a building may appear as two parallel lines. In addition, due to perspective, sides of buildings can often be seen and objects such as windows will contribute to the edge map in the case where the corner response filter fails to mask them. Image resolution reduction is an option but this is not favoured because it introduces additional errors into the position of orientation of extracted line segments.

The PPHT algorithm itself is a cause of problems because it stops searching the accumulator space when a threshold value is reached. This accumulator threshold, along with the minimum length threshold has to be kept low in order to detect all lengths of building boundaries. Unfortunately it also means that it is more likely to extract many short line segments instead of one long line segment from the highly detailed edge map. The feature map can be left at this state because the many overlapping line segments may be argued to sufficiently represent a building boundary, but it will cause great inefficiencies in the localization filter when the real measurements are compared to the perceived measurements (which come from the feature map).

Another problem highlighted from the previous section is that the resulting feature map is dependent on the parameters of the edge detection process and the parameters used. Consider figure 4.12, where edge maps of figure 4.3(a) are processed with the Canny edge detector using different threshold values. Using a high threshold level will yield less line segments and alleviate the messy feature map problem. However the resulting set of line segments may not represent a building boundary properly because the edge map is likely to have gaps in locations where a building boundary exists. Setting the threshold too low will ensure that the edge map fills pixels on building boundaries properly but will augment the messy feature map problem.

Some modifications to the PPHT are necessary to overcome the messy feature map problem and the edge detector parameter setting uncertainty. The modified


(a) Low Thresholds (b) Medium Thresholds

(c) High Thresholds

Figure 4.12: Canny edge detection threshold setting comparison

PPHT algorithm is shown in figure 4.13.

The first modification is to allow the PPHT algorithm to accept multiple edge maps. Three input edge maps with different Canny edge detection threshold settings are shown as inputs in this case with corresponding accumulator spaces. The accumulator space search will be performed sequentially starting from the high threshold edge map. Any lines that are detected will have its corresponding pixels removed from all input edge maps. Furthermore, the accumulator spaces will be updated according to the pixels removed.

The second modification is allow the thresholds to vary. The thresholds should be initially set at a high value representing the desire to search for long line segments

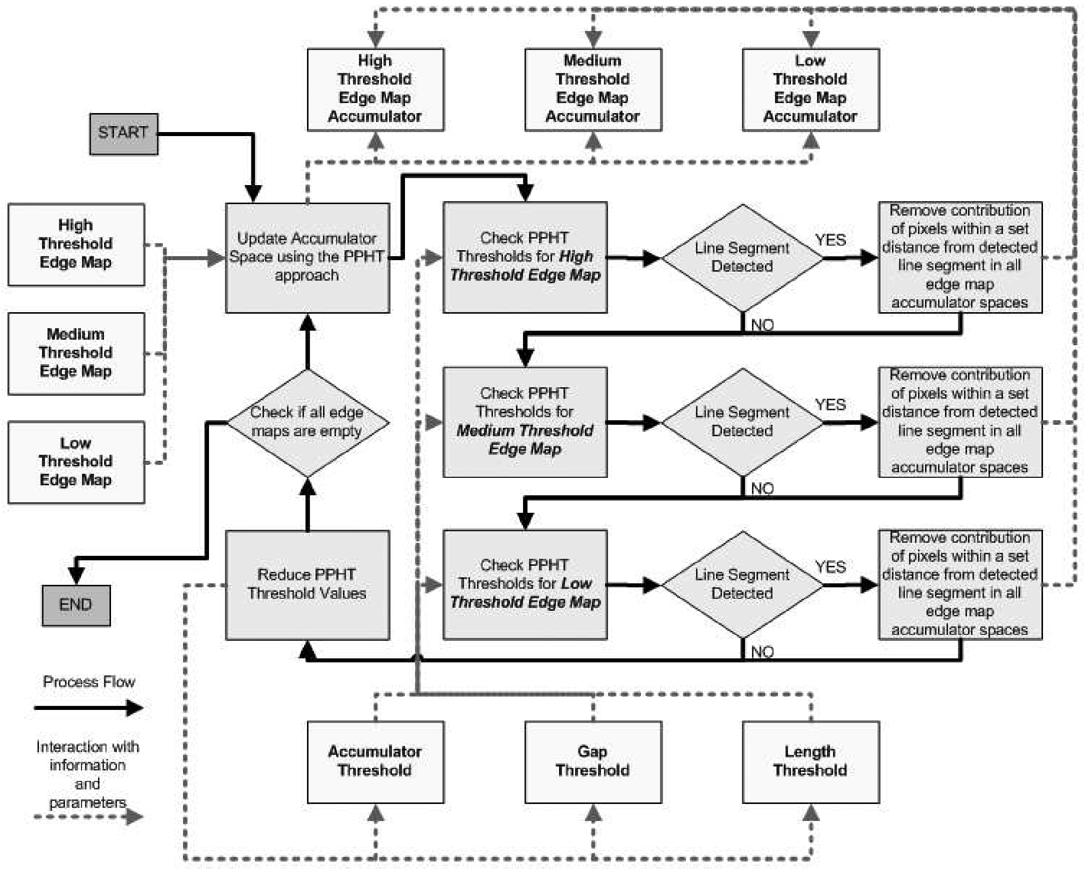

High Threshold Edge Map Accumulator
Medium Threshold Edge Map Accumulator
Low Threshold Edge Map Accumulator
START
High Threshold Edge Map
Medium Threshold Edge Map
Low Threshold Edge Map
Update Accumulator Space using the PPHT approach
Check PPHT Thresholds for High Threshold Edge Map
Line Segment Detected
YES
NO
Remove contribution of pixels within a set distance from detected line segment in all edge map accumulator spaces
Check PPHT Thresholds for Medium Threshold Edge Map
Check if all edge maps are empty
Check PPHT Thresholds for Low Threshold Edge Map
Reduce PPHT Threshold Values
END
Process Flow
Interaction with information and parameters
Accumulator Threshold
Gap Threshold
Length Threshold

Figure 4.13: The modified Progressive Probabilistic Hough Transform algorithm

first. When no line segments are detected in all the accumulator spaces at the current threshold values, they are decremented and the accumulator space search is repeated. This process continues until the thresholds reach a lower limit, which represents the shortest line segments that is desired to be detected.

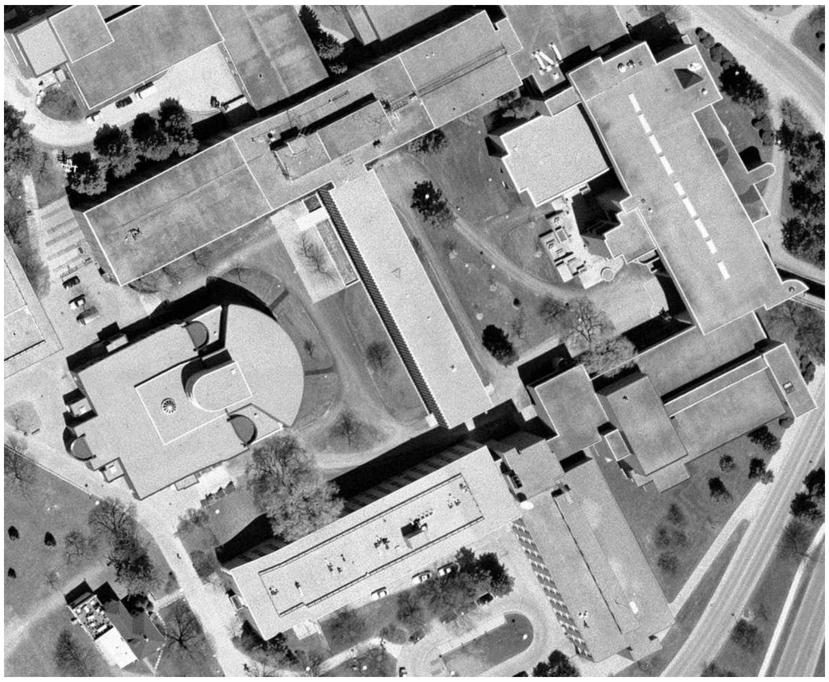
The third modification is in the pixel removal process from the edge maps when a line is detected. Instead of only deleting pixels that are coincident with the detected line segment, all pixels within a certain distance from the line segment (for instance, 3 pixel lengths) can be removed as well.

By allowing for multiple edge maps as inputs, there is no longer the difficulty and expectation of relying on a single edge detection threshold setting to produce an edge map with minimal noise while maximizing the response to true building edges (in other words, trying to maximize the SNR). This desired result is instead achieved by collaboratively using information from multiple edge maps. The accumulator space of the edge map with the highest threshold is searched first because it have the least noise and thus less likely to extract irrelevant line segments.

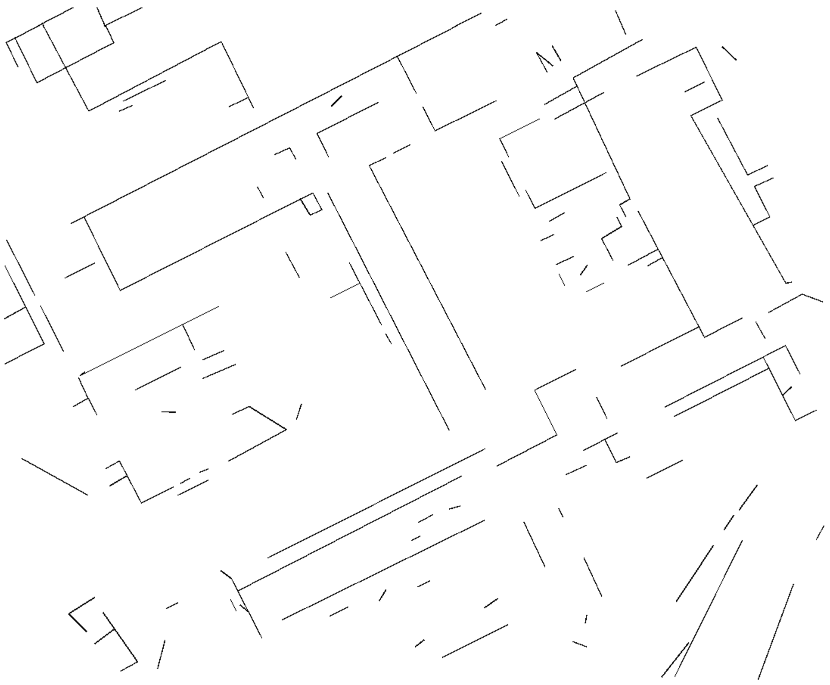
The PPHT algorithm uses fixed threshold inputs and they have to be kept at a value low enough to detect short building boundaries. By using varying thresholds, pixels making long line segments will be eliminated from the image first and will have a less likely chance to contribute to shorter line segments in the same location.

The change in the pixel removal process is placed to further reduce the problem or short line segments being detected and overlapping an existing longer segment.

Results of this modified PPHT process over the localization system testing area can be seen in figures 4.14. Compared to the previous unmodified PPHT result in figure 4.11(b), improvements can be seen around building boundaries where short overlapping line segments no longer exist and the number of features is significantly reduced without compromising the integrity of the feature map. Two other observations to make from this figure is that from all the filtering and masking that have been performed, a few short edges unrelated to building boundaries still scatter throughout the feature map. Also, there are unwanted line segments that are extracted because they come from objects that look like the edge of a building (like the side of a road for example). The final step in this feature map generation process will try to address these two problems. Additional examples of the modified PPHT process results can be found in Appendix A.



(a) Original image



(b) Result of the modified PPHT process

Figure 4.14: The result of the modified PPHT feature extraction process on the localization system testing area

4.6 Noise Removal from Feature Map

Any line segments extracted from the process covered in the previous sections that do not belong to a building boundary can be considered as noise in the feature map. These line segments exists because the shadow effect removal process or the corner response mask were not able to detect their source on the edge map. However, caution must be taken in removing these noisy features so that the good features are not disturbed. On other boundaries, the feature map may display discontinuities because the edge detector failed to respond at a certain location or perhaps it was mistakenly masked.

The building boundary gap problem can be easily corrected by searching the feature map for line segments that are coincident on the same line or close to parallel and close to each other in terms of distance. It is important to realize that in computer graphics, two lines that are meant to be coincident may actually not be when their orientations are calculated using image coordinates because an image is discretized into pixels. Therefore, it is necessary to relax the rules in determining if two lines are coincident. Here, line segments are considered coincident if their orientation is no more than the angular threshold of three degrees apart. The line segments are merged if the distance between them is less than a certain distance threshold. This threshold is set in proportion to the length of the line segments being examined. It is assumed that the length of a detected line segment is proportional to the probability of it being part of a true building boundary. Therefore longer line segments have more influence in the line merging process. the merged line segment is created by making the longest line segment possible from the four endpoints of the merging line segments. As a final check, the orientation of new newly merged line should be exceed the angle threshold when compared to the orientations of the merging line segments. Figure 4.15 is an illustration of the line merging procedure.

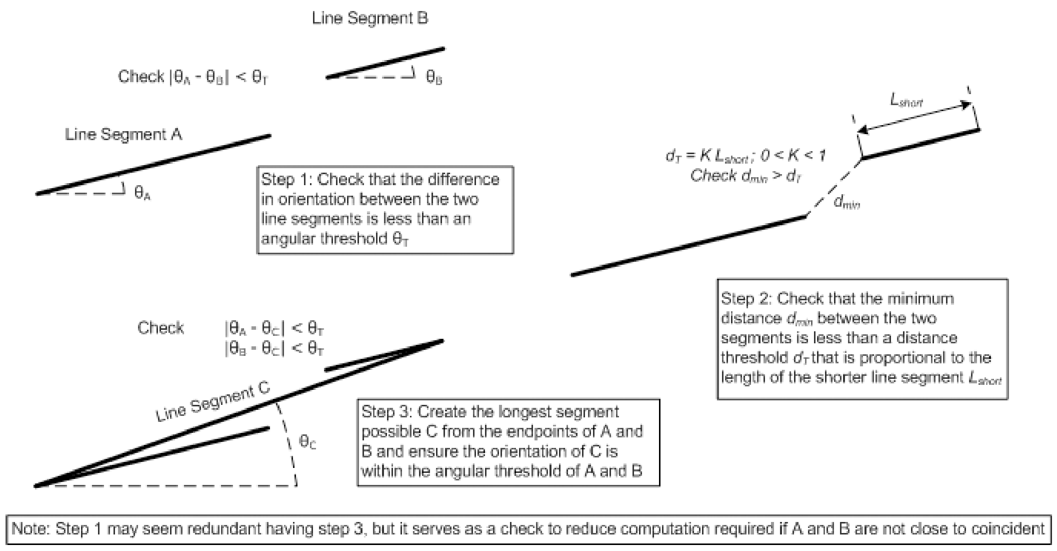

Line Segment B
Check $|\theta_A - \theta_B| < \theta_T$
θ_B
Line Segment A
θ_A
Step 1: Check that the difference in orientation between the two line segments is less than an angular threshold θ_T
L_{short}
$d_T = K\,L_{short}\,;\ 0 < K < 1$
Check $d_{min} > d_T$
d_{min}
Step 2: Check that the minimum distance d_{min} between the two segments is less than a distance threshold d_T that is proportional to the length of the shorter line segment L_{short}
Check $|\theta_A - \theta_C| < \theta_T$
$|\theta_B - \theta_C| < \theta_T$
Line Segment C
θ_C
Step 3: Create the longest segment possible C from the endpoints of A and B and ensure the orientation of C is within the angular threshold of A and B
Note: Step 1 may seem redundant having step 3, but it serves as a check to reduce computation required if A and B are not close to coincident

Figure 4.15: Coincidence check for line segments

Many short line segments scatter the edge map as shown previously in figure 4.14(b). To distinguish whether they are part of a real building boundary, consider a simple model of a building. A building should be enclosed and therefore the line segments making up its boundary should be close to each other. Theoretically, if the edge map used for feature extraction was perfect, both endpoints of a line segment should connect to another segment. However, since the edge map is not perfect, this requirement is relaxed in determining whether a line segment is part of a real building boundary. Instead of looking for coincidental endpoints, the distances from both endpoints of a segment to other line segments are considered. A minimum distance threshold proportional to the line segment is again used in determining the threshold. A line segment is removed from the feature map if it fails this test and figure 4.16 is an illustration of the test. A line segment which passes the test is unaltered.

At this point, the end of the feature map generation process has been reached, and the final result can be seen in figure 4.17. This map is now ready to be used to provide expected measurements in the particle filter localization system. The feature map is not perfect, meaning that a robot using this map will have errors in its interpretation of the real world. Fortunately, Bayes filters are known to remain robust even with such discrepancies. Having a map, it is now necessary to focus on the sensor measurements and extract information from there that can be compared with the feature map.

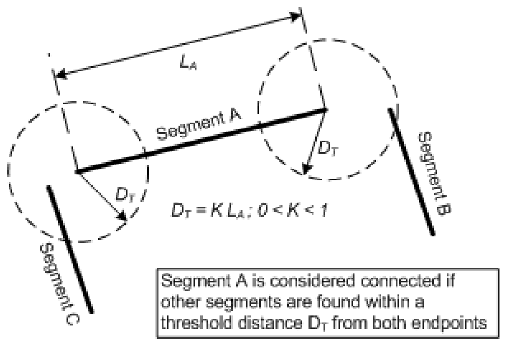
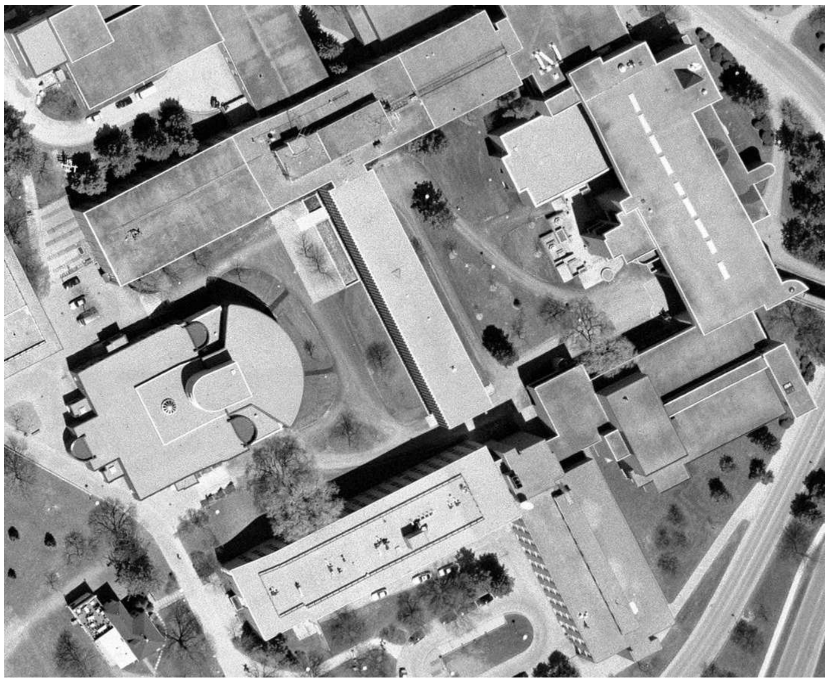

L_A
Segment A
D_T
D_T
Segment B
Segment C
$D_T = K\,L_A\,;\ 0 < K < 1$
Segment A is considered connected if other segments are found within a threshold distance D_T from both endpoints

Figure 4.16: Connectivity check to other line segments


(a) Original image

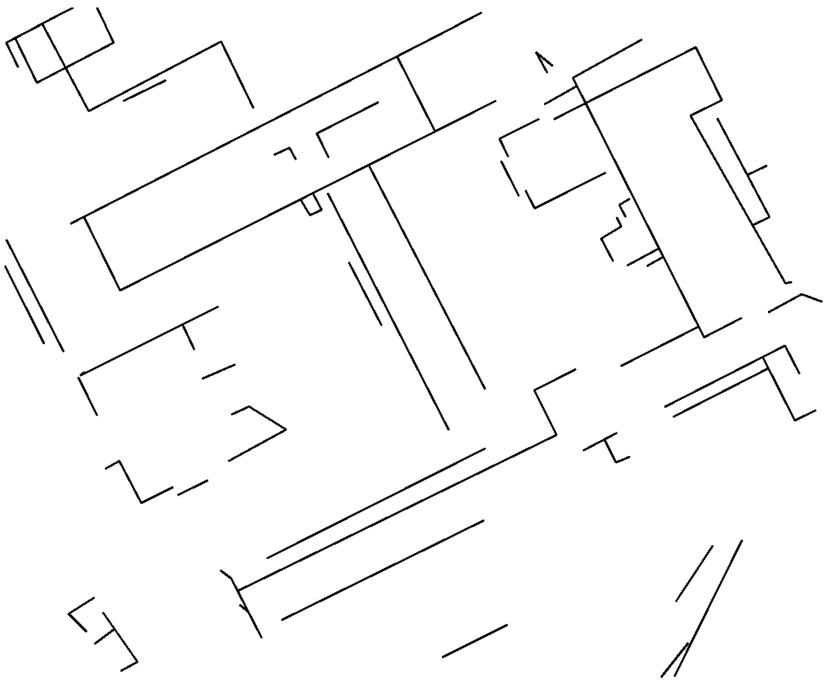

(b) The resulting feature map

Figure 4.17: The final result of the feature extraction process on the localization system testing area

Chapter 5

Camera Measurements

The processing of camera image frames for the vision based localization system is in many ways similar to the processing performed on an aerial image to extract features. The features that are of interest are the walls or building boundaries, which are obstacles that a mobile robot needs to navigate around to avoid a collision. The challenge again is to be able to identify features that indicate the presence of a building boundary while trying to minimize the responses from irrelevant items in the surrounding. This is especially difficult in an outdoor environment because of numerous objects in the surrounding area that may act as distractions. Trees are one example of distractions, as not only will they generate responses in processes such as edge detection, they may also occlude features that are of interest. Dynamic objects such as people walking around in the operating environment are also distractions. While filtering can be performed to reduce the responses from distractions, little can be done to retrieve information lost from occlusions.

Up to this point, it has not been revealed what feature properties will be used for the comparison between the observed measurement and the expected measurement in the localization system. As indicated earlier in section 2.4, in monocular vision information from the three dimensional world is projected onto a two dimensional image plane. In this process, information regarding the depth of objects in the scene is lost in the raw image data. Without the use of stereo vision, it may be possible to recover this information by assuming a fixed camera position (height) and using the base of a building to determine distance as depicted in figure 5.1. However, this method is susceptible to changes in orientation or more specifically the tilt of the chassis on which the camera is mounted. In outdoor environments, it would be very optimistic to assume that the ground is flat and leveled. In addition to this problem, the imaging system needs to be able to correctly identify where the base of a building is regardless of the building orientation with respect to the camera. This is a difficult object recognition problem, and to add to the difficulty, distractions in a scene are usually more prominent near the ground level and therefore the base line of a building may be occluded. Also a problem with operating outdoors is that buildings in the scene are usually a fair distance away from the camera and therefore the image resolution available may not be adequate to provide depth information with accuracy. Overall, while distance is theoretically a possible feature measurement, it is difficult to obtain practically and therefore it will not be used as a feature property for comparison in the localization system.

Although information regarding depth is lost in an image, the bearing of an object with respect to the camera can be used as a measure to restrict the possible location of an object as shown in figure 5.2. If distance measurements are obtainable, or the actual size of the object in view is known, then it would be possible to locate an object. Unfortunately these measures are unavailable and having only a bearing measurement is inadequate. There needs to be another measure indicating another useful property of objects in the field of view.

The orientation of a building wall with respect to the direction at which the camera

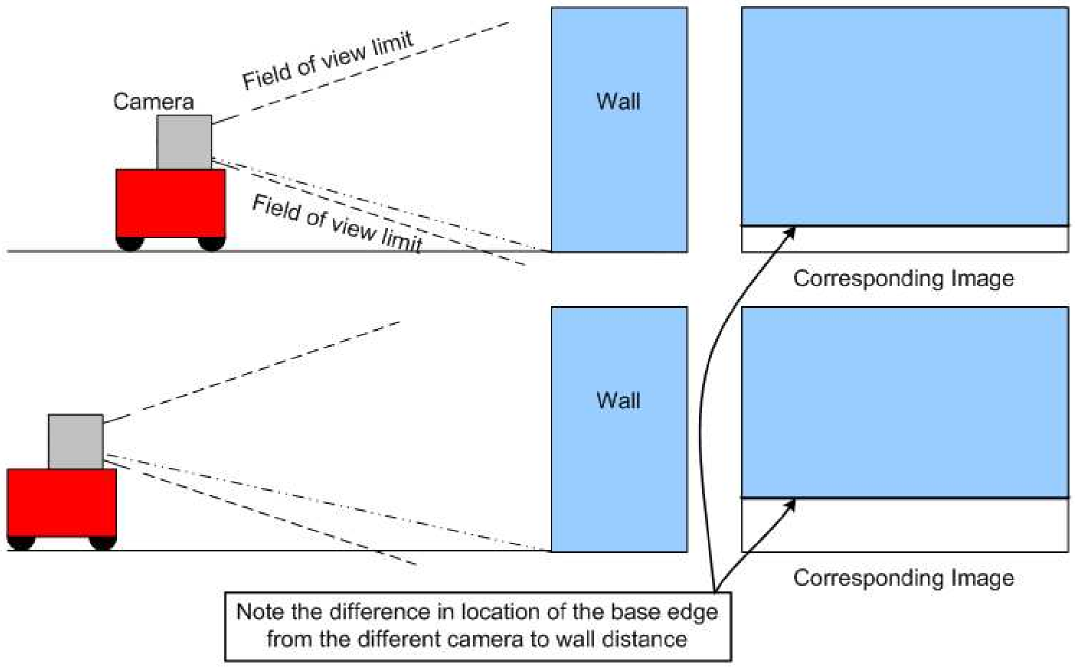

Field of view limit
Camera
Field of view limit
Wall
Corresponding Image
Wall
Corresponding Image
Note the difference in location of the base edge from the different camera to wall distance

Figure 5.1: Recovering depth information using building base lines in monocular vision

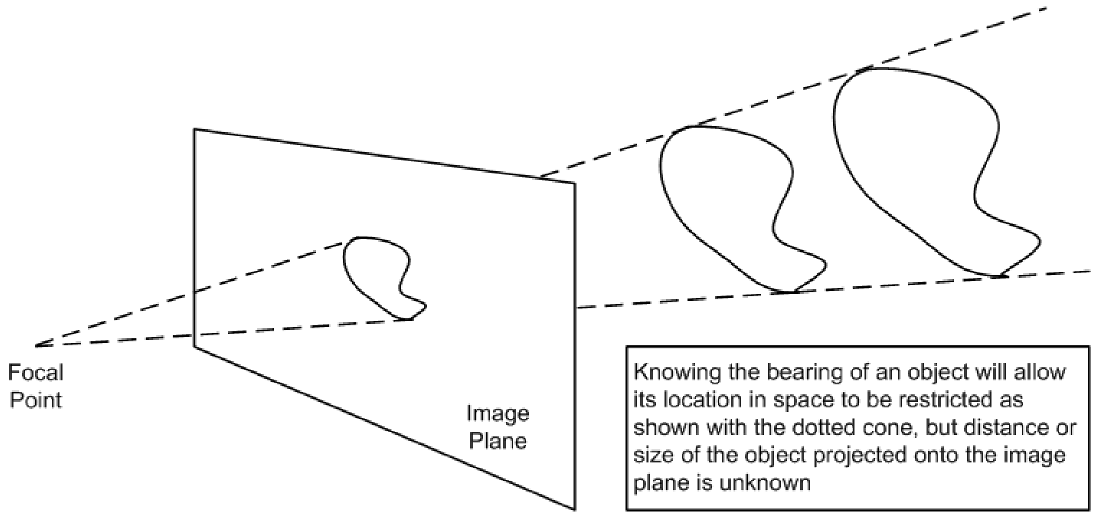

Focal Point
Image Plane
Knowing the bearing of an object will allow its location in space to be restricted as shown with the dotted cone, but distance or size of the object projected onto the image plane is unknown

Figure 5.2: Restricting object location based on bearing information

is pointed can potentially be another source of information. This information can be recovered from the effect of perspective (or how objects near-by appear larger in an image compared to a same sized object far away) using a technique known as vanishing point analysis. In order to use this method, it is necessary to extract line segments on the walls of buildings, so edge detection and the Hough transform (PPHT) will again be used in the process. In the next section, the distraction masking process for camera images is examined. Then the remainder of this chapter will look at at how the vanishing point analysis is used to retrieve information on the orientation of building boundaries.

5.1 Distraction Masking

The first step in processing a camera image for the localization system is to identify the edges in a scene. The Canny edge detector presented in section 4.1 will be used to generate an edge map from the scene. From this, the PPHT algorithm presented in section 4.4 will be used to extract line segments as features, which will in turn be used as inputs to the vanishing point analysis so that the orientation of building boundaries can be determined.

Shown in figure 5.3 is a typical example of an image captured by the camera for the localization system. The resulting edge map obtained using the Canny edge detector is shown in figure 5.4.

Many objects such as the tree shown in the figure respond to the edge detector because of the strong contrast the objects possess compared to other objects and the background of the image scene. It is necessary to remove the edge responses from such objects because features that do not belong to a building wall may cause incorrect and unpredictable results in the vanishing point analysis.

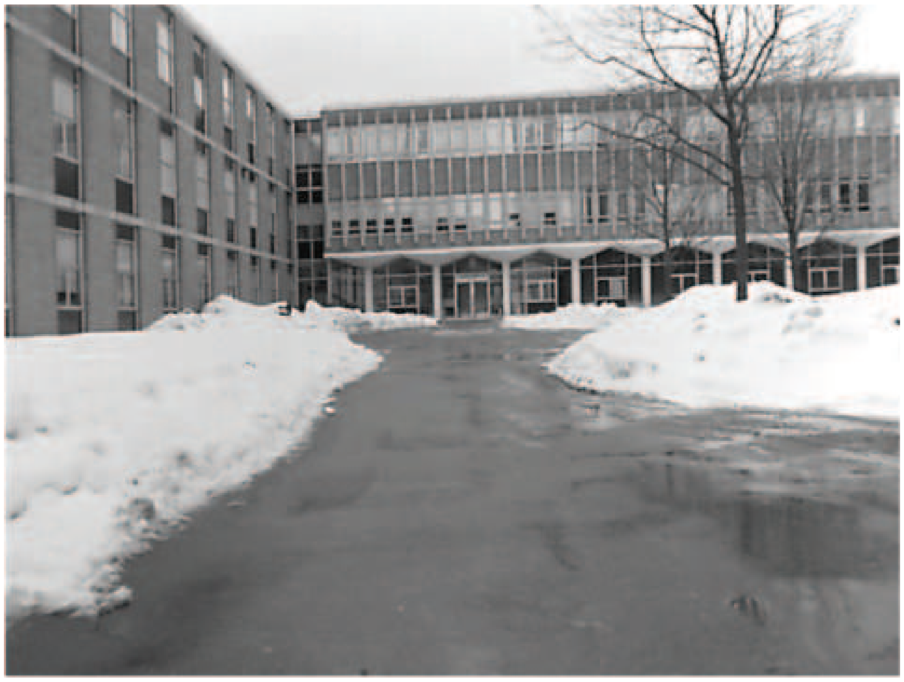


Figure 5.3: An image captured by the camera for localization

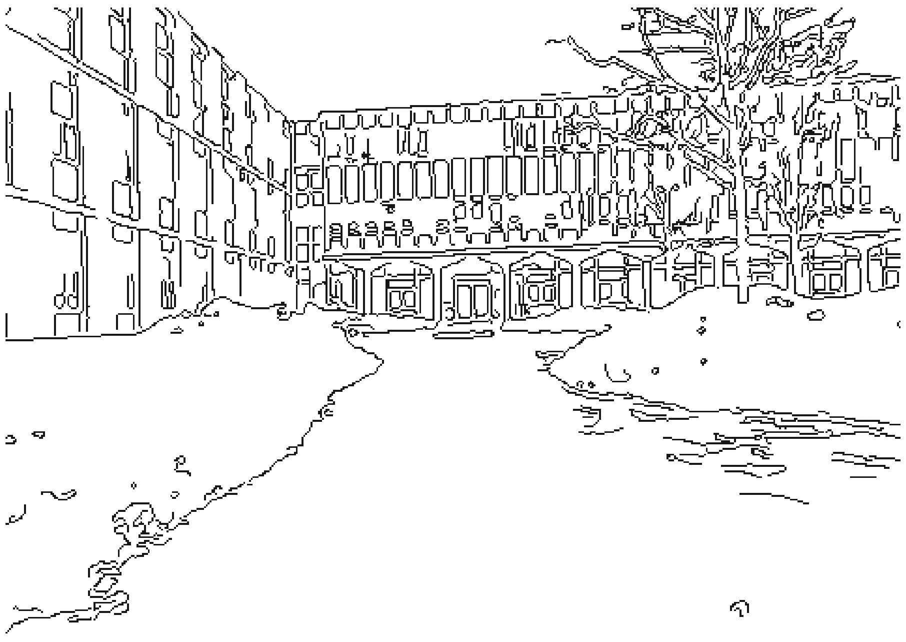


Figure 5.4: The edge map of figure 5.3 produced by the Canny edge detector

Inspection of numerous edge maps created from camera images suggests that edge responses from distractions in a scene are usually cluttered in certain areas of the edge map. The edge responses of interest from objects on building walls on the other hand are usually free from such cluttering or experience less of it. Therefore, creating a mask using a measure of edge pixel density seems feasible. To do this, the edge density at each pixel is measured using an $n \times n$ kernel, where n is an odd and positive integer, and where each element of the kernel is equal to $1/n^2$. As this kernel is convoluted over an image, the response at each pixel will represent the percentage of pixels in the neighbourhood that responded to the edge detector. For the images used in the localization system, the value of n is set to 5, and it was determined that a pixel should be masked if the edge density exceeds 60%. Using a morphological closing operation will help fill in small gaps in the mask. Figure 5.5 shows the density measure of figure 5.4, and figure 5.6 is the mask derived from the density measure. The resulting edge map after the density mask is applied is shown in figure 5.7.

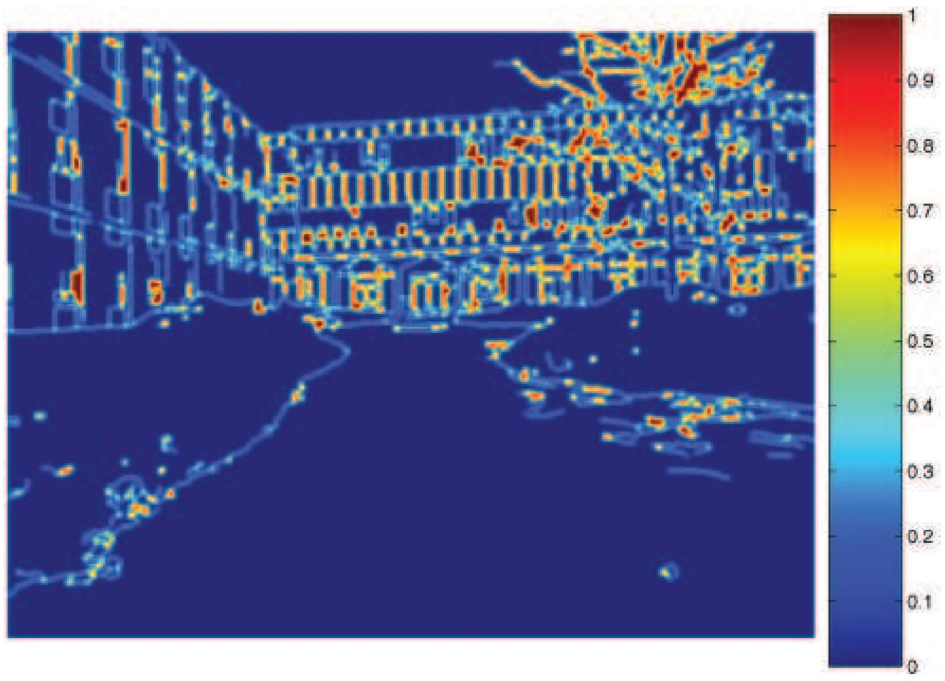


Figure 5.5: The edge map density measure for estimating areas of distractions

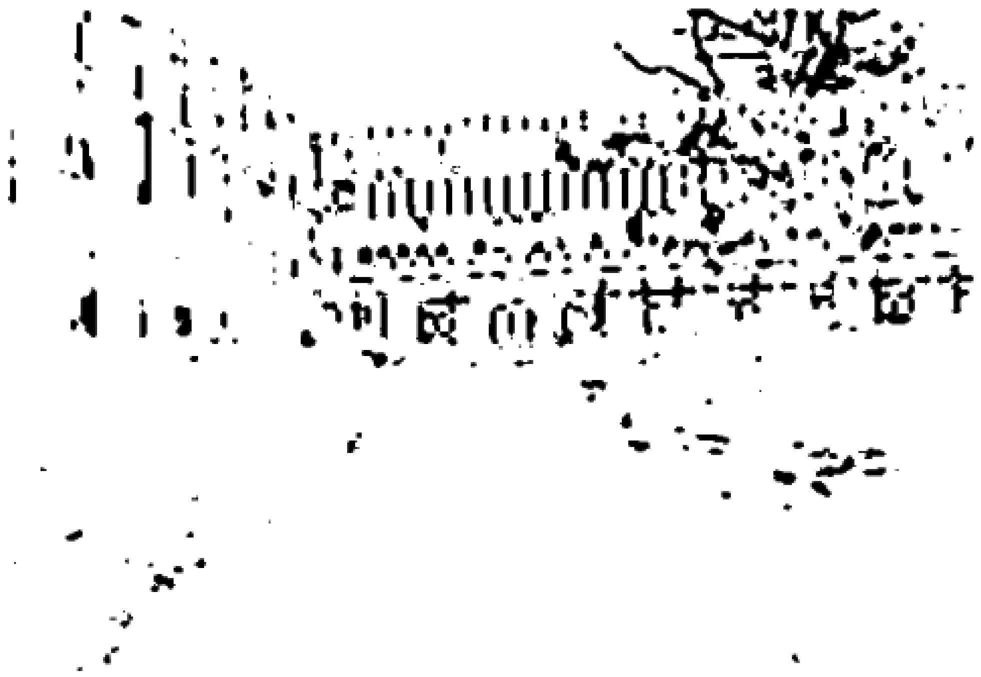


Figure 5.6: The edge response mask generated from the density measure

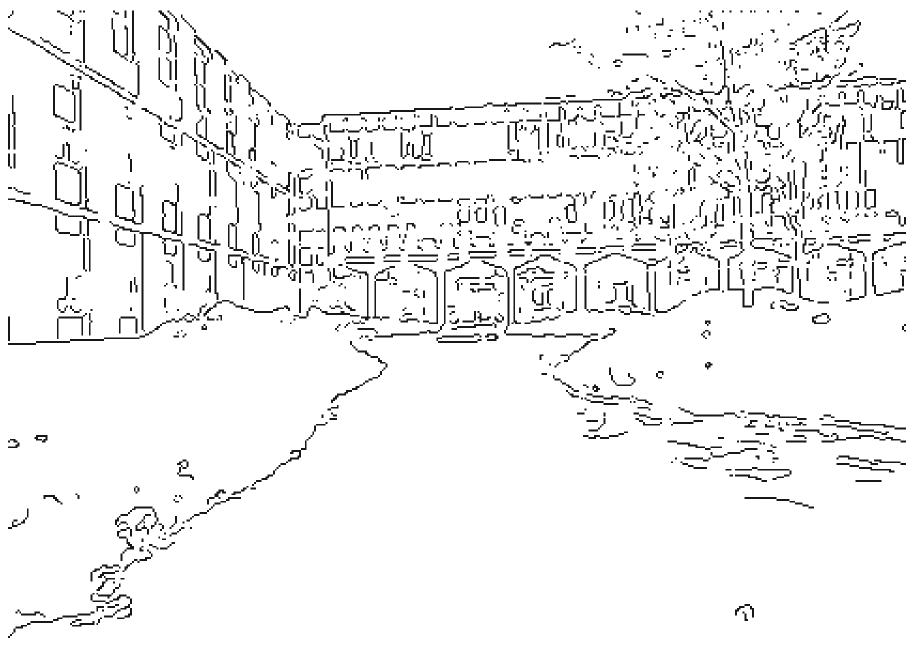


Figure 5.7: The resulting edge map after masking

Note that in the masked edge map, a large portion of the edge response produced by the tree has been eliminated. Although some distraction edge responses still remain, they are now less cluttered and therefore have a low chance of being detected in the Hough Transform. Another observation to make is that some edge responses that are of interest have been inevitably removed by the mask. This is a problem often faced in filtering where there is the need to retain the desired information while minimizing the noise. As will be shown later by the results, a good portion of the desired information is still present to complete the vanishing point analysis, suggesting that the density threshold used in generating the edge map filter is appropriately set. More examples of filtering using edge density measure are available in appendix B. The filtered edge map is now ready to be put through the Hough Transform to extract higher level information.

5.2 Vanishing Point Detection Overview

The concept of the vanishing point has been known for centuries. When parallel lines in 3d space (or object space) are projected onto an image plane using a central projection model, the lines on the image plane will intersect at a point known as the vanishing point [51, 52] as shown in figure 5.8. The central projection model forms an image by taking the light rays that pass through a common focal point. This model is consistent with human vision as well as cameras such as the one used with the localization system, and the 3d to 2d transformation governed by this model is the reason for perspective [53].

Note that in figure 5.8, a set of lines in object space may project onto the image plane as parallel lines. In this case, the vanishing point for this pair of lines is at infinity. The Gaussian sphere shown in figure 5.9 was introduced as a method

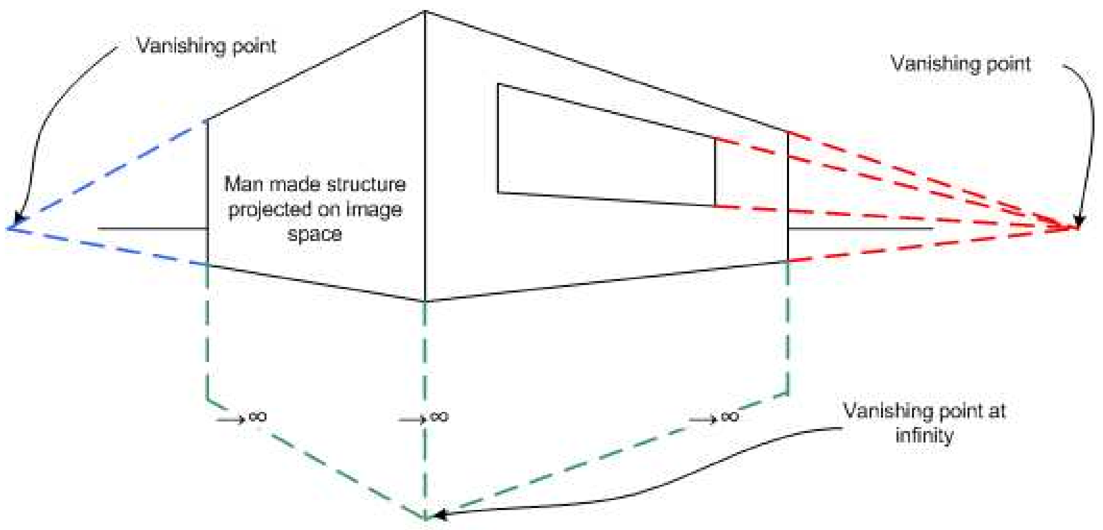

Vanishing point
Vanishing point
Man made structure projected on image space
→∞
→∞
→∞
Vanishing point at infinity

Figure 5.8: The vanishing points of an object in image space

of defining vanishing points that avoids such singularities [53, 51]. The Gaussian sphere is a unit sphere centered on the focal point of the vision system. Using the Gaussian sphere, a vanishing point can be defined by its projection (using the central projection model) on the sphere, where it has a unique coordinate (azimuth and elevation). Another way of finding a vanishing point on the Gaussian sphere is to look at the plane formed by a line in object or image space, and the focal point. This plane is known as the interpretation plane, and it will intersect the Gaussian sphere to produce a great circle. Multiple parallel lines in object space correspond to multiple great circles on the Gaussian sphere that will intersect at a common point. This point is equivalent to the projection of the vanishing point in image space onto the Gaussian sphere.

When a vanishing point has been identified in an image, it is possible to infer the orientation of 3d objects from an image [55]. For an image captured with the on board camera of the localization system, finding the vanishing point for a group of

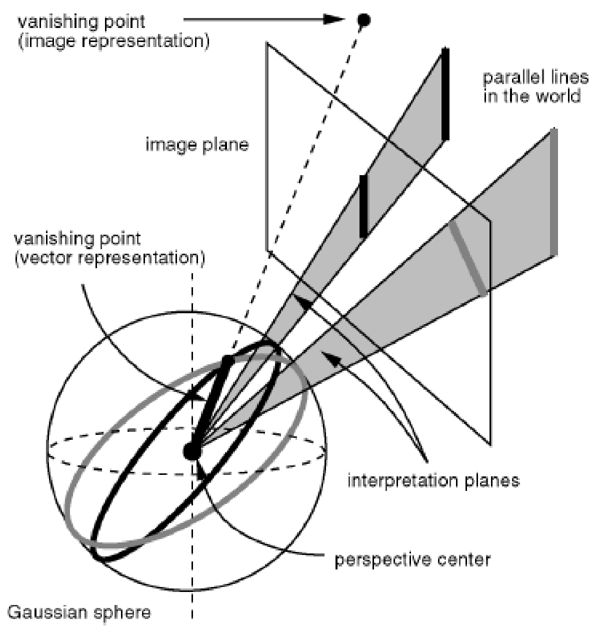


Figure 5.9: The Gaussian Sphere [54]

line features that belong to the same building wall will allow the orientation of the wall to be predicted.

In this problem, there is difficulty in determining which lines on an image has to be grouped together (in other words it has to identify the lines which are parallel in space). A historically well known approach introduced in [53] adopts the Hough Transform method and discretizes the Gaussian sphere as an accumulator plane. Great circles projected by lines on the image plane will cast votes in the accumulator space as in the Hough Transform, and vanishing points are found by looking for local maxima. This method is identified as the Gaussian sphere approach in many publications, but was realized to have several problems. The first is as with any Hough Transform approach, the size of accumulator cells is arbitrary. A more serious problem is that a regular quantization in the parameters (azimuth and elevation) does not lead to equal sized accumulator areas on the Gaussian sphere [51, 56, 57, 58], which can bias the voting scheme. Although this is the case, the Gaussian sphere approach is still well known and popular, and in realization of the existent problems, many methods have been developed to enhance the Gaussian sphere approach such as the ones presented in [54, 59].

A slightly different approach which still uses the Gaussian sphere as an accumulator space looks at the intersection between all line pairs in an image [60, 57]. These intersections are then used to cast votes in the accumulator space. To avoid the problems of the Hough Transform approach, [56] proposed to find vanishing points by clustering intersection points. Furthermore, in the presented approach, an intersection point is weighted by properties of the contributing lines to represent the likelihood of the intersection point being a true vanishing point.

Besides the methods described above, many other implementations exists. One approach chooses to look at more than two line features from the image plane at the same time [51]. Another approach assumes that the faces of objects in an image are orthogonal and searches for three vanishing points in mutually orthogonal directions at the same time. While good results can be seen from these publications, these approaches have been acknowledged to be computationally expensive and unsuitable for real time applications [61]. Other vanishing point methods that exists in the literature include [62, 63, 58, 64].

The use of vanishing points in mobile robotics has been used in the past in [65]. However the navigation in this case is behaviour based, meaning that the vanishing point observations are directly linked to motion control. The urban localization system in development on the other hand is map based, and the method taken for vanishing point detection is based on the intersection clustering approach. Some modifications and assumptions are made however to better suit the localization system.

5.3 Vanishing Point Identification

An intersection clustering based approach is taken as the vanishing point detection method because it avoids problems that exist with the Hough Transform based Gaussian sphere approach. However, the Gaussian sphere is still used as a tool in this method. The vanishing point detection problem is essentially a pattern recognition problem where it is necessary to identify the existing classes (vanishing points). When this is accomplished, features (line segments from building walls) can be classified accordingly, with the azimuth of the vanishing point corresponding to the orientation of a building wall in space.

Given a line segment identified by using the Hough Transform on the image plane, its corresponding interpretation plane is formed by itself and the center (focal point and origin) of the Gaussian sphere. This plane can be represented by a normal vector ϕ as shown in figure 5.10.

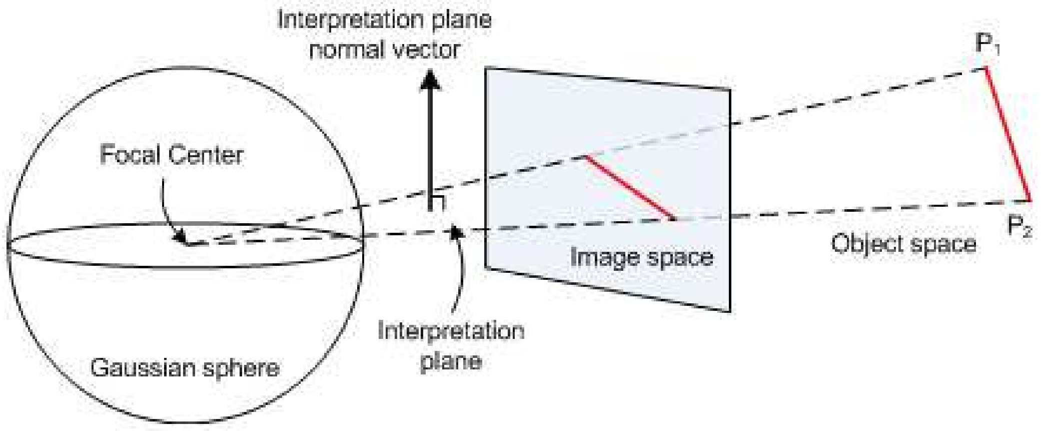


Figure 5.10: The interpretation plane normal vector

If vectors P_1 and P_2 go from the origin to the end points of the line segment, the normal vector can be calculated using equation 5.1 [53].

$$\phi = (\phi_x, \phi_y, \phi_z) = \frac{P_1 \times P_2}{|P_1||P_2|} \tag{5.1}$$

A vector to an arbitrary point on the surface of the Gaussian sphere g can be expressed in Cartesian coordinates using the Gaussian sphere azimuth α and elevation β parameters according to equation 5.2 [53].

$$g = (\sin\alpha\cos\beta, \sin\beta, \cos\alpha\cos\beta) \tag{5.2}$$

When an interpretation plane intersects with the Gaussian sphere, the great circle formed is formulated when the condition expressed in equation 5.3 [53].

$$g \cdot \phi = 0 \tag{5.3}$$

By manipulating equations 5.1, 5.2, and 5.3, the great circle can be defined by the elevation as a function of azimuth as shown in equation 5.4 [53]

$$\beta = \arctan\left(\frac{-\phi_x \sin\alpha - \phi_z \cos\alpha}{\phi_y}\right) \tag{5.4}$$

The point of intersection between two great circles is potentially a vanishing point of building walls in the camera image. However, this can not be determined until all intersection points are found and analyzed through clustering. The intersection between two great circles (identified by subscripts 1 and 2) occurs when equation 5.5 holds true.

$$\frac{-\phi_{x1}\sin\alpha - \phi_{z1}\cos\alpha}{\phi_{y1}} = \frac{-\phi_{x2}\sin\alpha - \phi_{z2}\cos\alpha}{\phi_{y2}} \tag{5.5}$$

The above equation can be solved for in terms of the azimuth parameter, which leads to equation 5.6

$$\alpha = \arctan\left(\frac{\phi_{z2}\phi_{y1} - \phi_{z1}\phi_{y2}}{\phi_{x1}\phi_{y2} - \phi_{x2}\phi_{y1}}\right) \tag{5.6}$$

Equation 5.6 can then be back substituted into equation 5.4 to solve for the corresponding elevation of the intersection point.

Up the this point, the approach to finding intersections is largely the same as the methods presented in [60] and [56]. One enhancement is used to reduce the number of intersection points and also to eliminate the ones that are extremely unlikely to be a true vanishing point. To identify these points, it is necessary to examine the corresponding intersection of lines on the image plane. As illustrated in figure 5.11, a true vanishing point is always found beyond the endpoints of the corresponding line segments. Therefore, by performing a simple check on where an intersection between two line segments is located on the image plane, irrelevant candidates can be eliminated.

Consider a line segment (a) with endpoints (x_1, y_1) and (x_2, y_2), and another line segment (b) with endpoints (x_3, y_3) and (x_4, y_4) on the image plane. Taking the first line segment as an example, it can be expressed according to equation 5.7, where $0 \leq u_a \leq 1$.

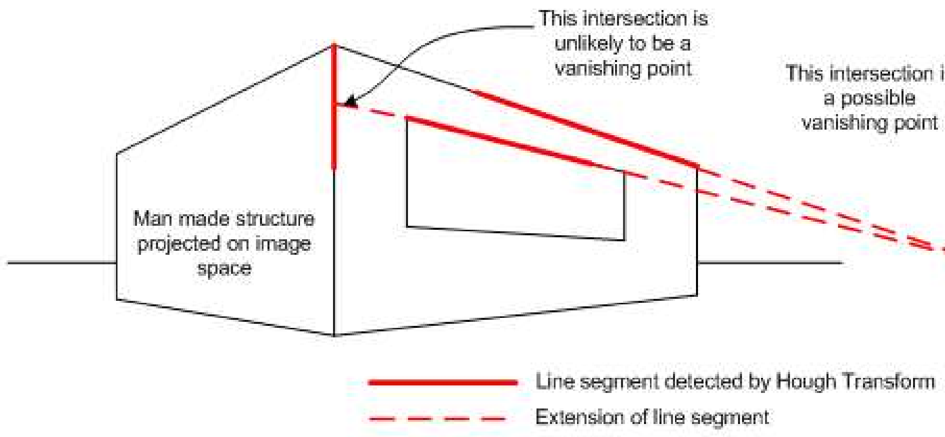

This intersection is unlikely to be a vanishing point
This intersection is a possible vanishing point
Man made structure projected on image space
Line segment detected by Hough Transform
Extension of line segment

Figure 5.11: Identifying irrelevant intersection points

$$
\begin{aligned}
x_1 &= u_a(x_2 - x_1) \\
y_1 &= u_a(y_2 - y_1)
\end{aligned} \tag{5.7}
$$

The intersection between two lines extended from the two line segments occurs at a particular value of u_a and u_b as indicated in equations 5.8 and 5.9.

$$
u_a = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_2) - (x_4 - x_3)(y_2 - y_1)} \tag{5.8}
$$

$$
u_b = \frac{(x_4 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_2) - (x_4 - x_3)(y_2 - y_1)} \tag{5.9}
$$

If either $0 \leq u_a \leq 1$ or $0 \leq u_b \leq 1$, it implies that the intersection point is coincident with one of the line segments and therefore it should be ruled out as a possible vanishing point. With this implementation, it was found that errors at the end of the vanishing point analysis were greatly reduced.

In the intersection clustering approach taken by [56], intersection points were weighted to signify their likelihood of being a true vanishing point. A similar approach is taken here, and a weight is assigned to an intersection point according to the lengths of the contributing lines. A longer line segment is less likely to be the product of noise and therefore an intersection point resulting from a long ling segment should be more likely to exist as a vanishing point. Mathematically, given line segments a and b of lengths L_a and L_b respectively, the weight of the intersection point W_i is determined by equation 5.10, where k is a constant.

$$W_i = min(kL_a, kL_b) \tag{5.10}$$

In the field of pattern recognition, there are numerous approaches to clustering, which are used for data exploration and to provide prototypes for classifiers [66]. In this case, the data is the set of intersection points, and the prototypes are the vanishing points being seeked. Once found, these will be used to classify the line segments observed in the image. The weighted k-means method was used in [56], but this clustering method requires the number of clusters to be known beforehand. Here, the subtractive clustering algorithm [67, 68]is used instead as it does not require the number of clusters to be known.

The subtractive clustering algorithm is iterative. The first step is to find the density measure D at each data (intersection) point x_i by looking at its distance to n other data points according to equation 5.11. In this equation, $\delta()$ is the distance measuring function and r_a is a scaling constant.

$$D_i = \sum_{j=1}^{n} \exp\left(-\frac{\delta(x_i, x_j)}{(r_a/2)^2}\right) \tag{5.11}$$

Traditionally, the Euclidean distance is taken between data points to determine the density because feature dimensions for data points are usually placed in Euclidean space. However, since a Gaussian sphere is used and the intersection points are defined by the azimuth and elevation, the arc distance on the sphere is used as a substitute. This distance can be calculated according to equation 5.12 [60].

$$\delta(x_i, x_j) = \arccos\left(\cos\left(\frac{\pi}{2} - \beta_i\right)\cos\left(\frac{\pi}{2} - \beta_j\right) + \sin\left(\frac{\pi}{2} - \beta_i\right)\sin\left(\frac{\pi}{2} - \beta_j\right)\cos\left(\alpha_i - \alpha_j\right)\right) \tag{5.12}$$

Once the density D_i at all data points has been calculated, they are multiplied by the weighting factor W_i calculated previously. The point with the largest density D_c is chosen as a cluster center x_c. The density influence of this cluster center is then subtracted from all the data points according to equation 5.13, and all the density measures are updated. In this equation D' is the updated density measure and r_b is a scaling factor similar to r_a. The magnitude of r_b will determine the extent to which other data points are influenced by the subtraction. Literature on subtractive clustering suggests setting $r_b = 1.5r_a$.

$$D'_i = D_i - D_c \exp\left(-\frac{\delta(x_i, x_c)}{(r_b/2)^2}\right) \tag{5.13}$$

The clustering process then looks for the next highest density measure and density subtraction is repeated. This will continue iteratively until an end condition is satisfied. For the localization system, the end condition is a threshold T which will end the clustering process when $T > D_c$.

To further aid the clustering process, it is assumed that the onboard camera will normally remain leveled and not experience severe tilting and rolling. With this assumption, it follows that vanishing points will likely appear near the pole (from vertical lines in object space) and the equator (from horizontal lines in object space) of the Gaussian sphere. Therefore, the clustering problem can be divided into two parts; one that searches above a certain angular elevation for a vanishing point near the pole (for instance at greater than 70 degrees), and one that searches between two elevations slightly above and below the horizon (at less than 20 degrees and greater than -20 degrees for example). This implementation has been tested and confirmed to generate better and more accurate vanishing point detection results.

The number of cluster centers (vanishing points) that the clustering process will find is dependent on the scene of the corresponding image. In a scene where building walls are heavily occluded by trees and other distractions, it is possible to find no vanishing points at all. In general however, usually three vanishing points can be detected from a scene captured by the onboard camera of the localization system. These three points correspond to the vertical edges and usually two orthogonal building walls, where the azimuth of the vanishing points are approximately 90 degrees apart. At this point the prototypes for classification have been identified, and it is now possible classify the line segments found in an image to associate them with a particular vanishing point.

5.4 Feature Classification

Classification in pattern recognition involves identifying the membership of a data point given information regarding the possible classes that the point can belong to. Previously, vanishing points have been identified through subtractive clustering and they are to serve as the classes in the classification process. The objects that need to be classified are the line segments in image space. In [56], the approach taken was to classify the intersection points and backtrack to label the line segments from which the intersection point was created. A different approach is taken here with the argument being that some intersection points may not belong to any class at all because they may be the product of lines that are not parallel in object space, or noise. The method taken proposes to classify line segments directly using distance to vanishing point measures on the Gaussian sphere.

Line segments in image space project onto the Gaussian sphere as great circles. Often in pattern recognition, a data point being classified has the same features as the classification prototypes. Vanishing points (a point on the Gaussian sphere) and great circles (a set of points on the Gaussian sphere) are different entities but classification is still possible by considering the point on the great circle closest to the a given vanishing point. In other words, the great circle distance between a vanishing point and the great circle that originated from a feature in the image space will be used to classify the same feature.

Membership of a class is won by the shortest distance measure. The distance calculation can be accomplish using vector mathematics, so the first step to take is to convert the vanishing points V_i into Cartesian coordinates using equation 5.2. The interpretation plane normal vector ϕ is a unit vector perpendicular to the great circle formed by the projection of a line segment onto the Gaussian sphere. Using this, the distance to the vanishing point d_i can be calculated using equation 5.14.

$$d_i = \frac{\pi}{2} - \arccos\left(\phi \cdot V_i\right) \tag{5.14}$$

In order to avoid mistakenly classifying line segments, a maximum threshold distance is set. A vanishing point class C_i will only win the membership of a line segment λ_j if the distance measure is less than the distance measure to all other vanishing points, and if the distance measure is less than the threshold d_{max}. Mathematically, the classification criterion is expressed in equation 5.15.

$$\lambda_j \in \begin{cases} C_i & \text{if } d_{ij} \leq d_{max}; \\ \emptyset & \text{if } d_{ij} > d_{max} \end{cases} \tag{5.15}$$

Another precaution taken is to ignore all features below a certain elevation because of their proximity to the ground where the chance of a feature being a distraction is higher.

The result for line segment classification can be seen in figure 5.12. In this figure, the thick line segments are the ones detected from using the PPHT on figure 5.7. These line segments are colour coded according to the vanishing point that they belong to. Red lines belong to the first vanishing point, green lines belong to the second vanishing point, and blue lines belong to the third. A black line segment is one which does not belong to any class so its orientation is unknown. The location of the corresponding vanishing points on the Gaussian sphere has also been identified in the figure by their azimuth and elevation in units of degrees. The azimuth is of greater interest as it indicates the orientation of line segments in object space. When the azimuth is at zero, it implies that the line segments are parallel with the direction in which the camera is pointed. When the azimuth is at 90 degrees or -90 degrees, it means that the line segment is perpendicular to the viewing direction. Additional results can be found in appendix B.

It turns out that the vanishing point analysis is quite capable of enduring unwanted rolling and tilting of the camera without causing much deviation to the estimate of building wall orientations. To prove this, consider a camera that has endured some degree of rolling (rotation about the direction of view) and pitching (rotating that causes the camera to tilt up and down). It is more convenient in this case to use the

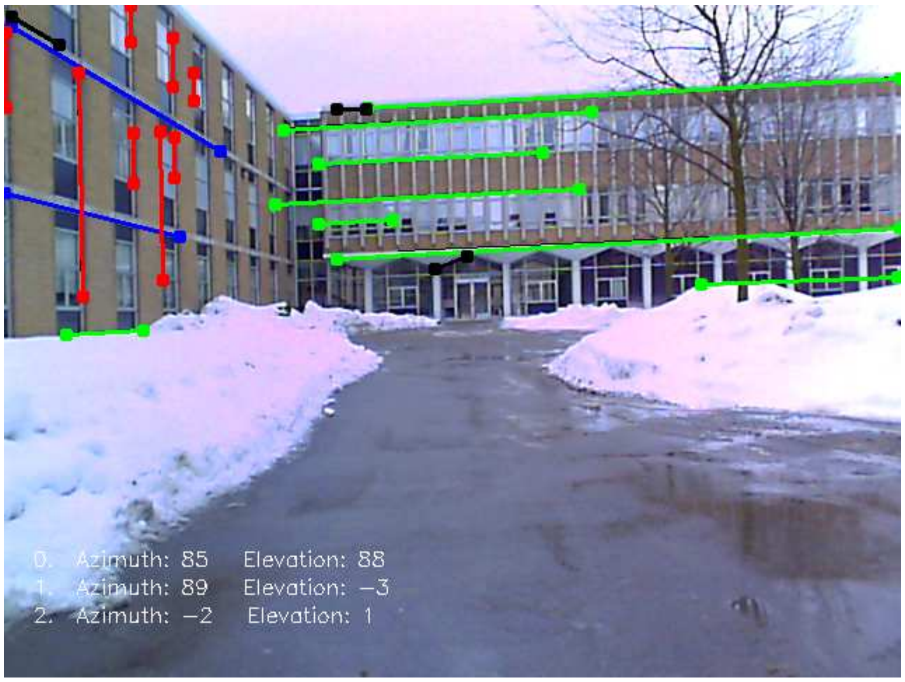

0. Azimuth: 85 Elevation: 88
1. Azimuth: 89 Elevation: −3
2. Azimuth: −2 Elevation: 1

Figure 5.12: Result of the vanishing point analysis

vanishing point for vertical lines as an indication of tilt and roll. Camera movement will be offset the vanishing point for vertical lines from the pole of the Gaussian sphere. Correcting this offset requires two mathematical transformations in Cartesian coordinates as shown in equation 5.16. Here, the angles γ and θ are required to move the offset vanishing point back to the pole of the Gaussian sphere.

$$\begin{aligned} R_\gamma R_\theta &= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \\ &= \begin{bmatrix} \cos\gamma & -\sin\gamma\cos\theta & \sin\gamma\sin\theta \\ \sin\gamma & \cos\gamma\cos\theta & -\cos\gamma\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \end{aligned} \tag{5.16}$$

Applying this transformation to other vanishing points reveals that their azimuths are almost unchanged. Since tilting and rolling causes the most change in azimuth near the zero azimuth mark, figure 5.13 is generated to show how little effects camera movement (quantified by the offset of the vanishing point at the pole of the Gaussian sphere) has on the azimuth measure at this location. It is not until severe camera tilting and rolling occurs that more significant error is experienced. For a simplified explanation of why the azimuth measure has such tolerance to camera movement, consider how the cosine of a small angle (representing some rotation) remains close to the value of one, and apply this concept to the Gaussian sphere.

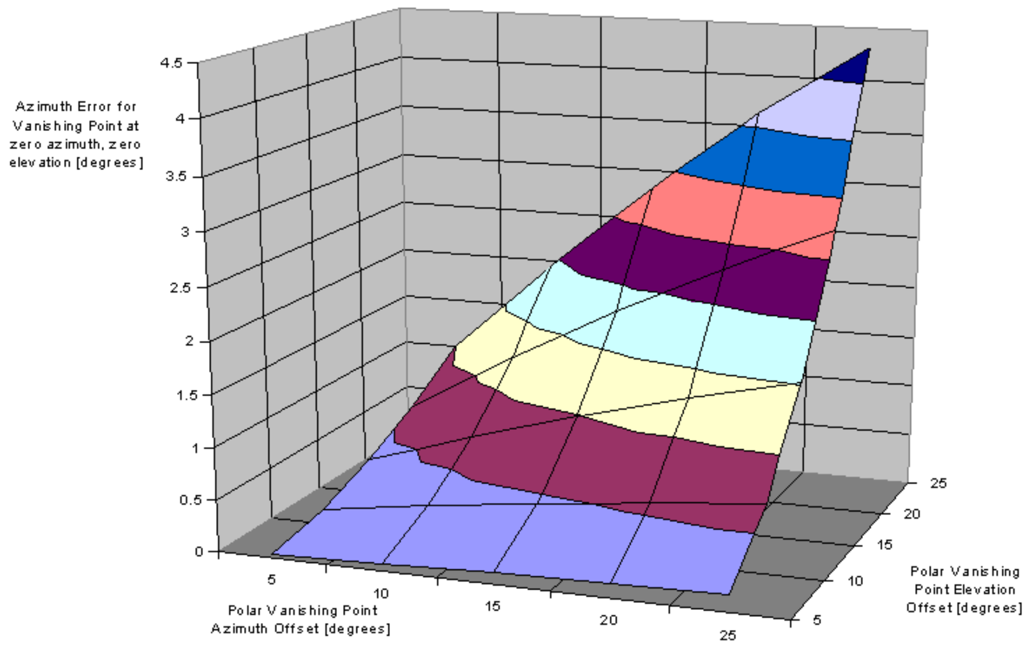


Figure 5.13: Azimuth error due to camera tilt and roll

5.5 Data Representation

It would be useful to organize all the classified line segments into a structure that is convenient to use in the particle filter. This is done by first identifying the azimuth of all the line segments that have an associated non-vertical vanishing point (hence the line segment is not a vertical line in object space). These points along the azimuth signal a change in the orientation or the number of observable objects. A data structure list is created and each element of it holds information regarding the orientation of the objects in view at each section. In other words, each element contains the vanishing point azimuth of objects visible in the corresponding view section. This data representation scheme is illustrated in figure 5.14. Here, the detected features of the building in view are already associated with vanishing points and hence each of their orientation has already been estimated. The corresponding data structure list is illustrated beneath the scene, and the contents of selected elements are shown as examples. The top-down view of the building in the scene is also drawn in the figure to better illustrate the configuration of the building in the scene.

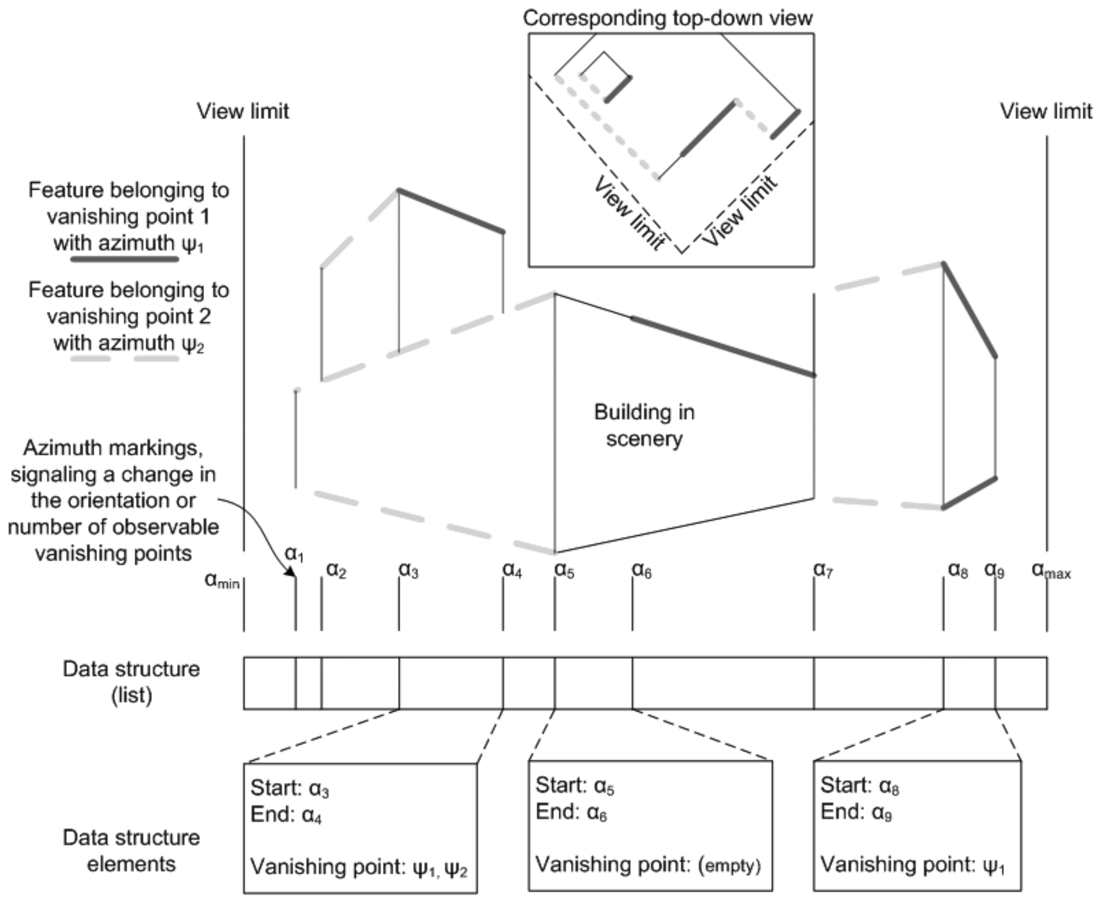

Corresponding top-down view
View limit
View limit
View limit
View limit
Feature belonging to vanishing point 1 with azimuth ψ_1
Feature belonging to vanishing point 2 with azimuth ψ_2
Building in scenery
Azimuth markings, signaling a change in the orientation or number of observable vanishing points
α_{min} α_1 α_2 α_3 α_4 α_5 α_6 α_7 α_8 α_9 α_{max}
Data structure (list)
Data structure elements
Start: α_3 End: α_4 Vanishing point: ψ_1, ψ_2
Start: α_5 End: α_6 Vanishing point: (empty)
Start: α_8 End: α_9 Vanishing point: ψ_1

Figure 5.14: Data representation for information extracted from on board camera images

Chapter 6

Particle Filter Implementation

In the previous chapters, methods for feature map generation and camera image processing were described. These two processes provide the map and measurement inputs essential for the implementation of a particle filter localization algorithm, or any Bayes filter in general. Also required is the ability to predict state transition between measurements. This can be accomplished using control inputs and a state transition model, both of which are assumed to be available.

In this chapter, the implementation of the vision based urban localization system is discussed. First, the state vector $\underline{x}$ which the particle filter attempts to estimate is defined by two robot position variables (x,y) and a robot heading variable (ϑ) as expressed in equation 6.1.

$$\underline{x} = \begin{bmatrix} x \\ y \\ \vartheta \end{bmatrix} \tag{6.1}$$

To review, the particle filter algorithm contains the following steps. First particles are distributed in state space according to the initial belief state. Each particle is then moved according to the state transition model and motion control inputs, the details of which will be discussed in the following section. After state transition, each particle will compare the expected and observed sensor measurements. The expected measurement is different for particles at different states, and it is necessary to refer to the processed aerial image to determine the expected measurement for each particle. Observed, or real measurements come from the on board camera, and were extracted using the image processing techniques and vanishing point analysis described in the previous chapter. The comparison of expected and real measurements results in an importance factor for each particle. This factor is used in the resampling of particles, or deciding which particles will survive onto the next iteration of the particle filter.

Validation of the particle filter is done over a course set in an urban environment that covers an area of approximately $220m \times 180m$. The aerial map of this area has already been presented in figure 4.14. In the duration of this course, the maximum velocity that will be reached is $2.8\frac{m}{s}$, the lowest is $1.0\frac{m}{s}$, and the average velocity is $1.9\frac{m}{s}$ The on board camera is used to capture and save images in the duration of the course and the images are then processed offline.

Results of the localization system will be presented for the kidnapped robot situation, where the initial state is unknown. Additionally, the sensitivity of the localization system to changes in the control input covariance is examined. Finally, process timing information will be presented.

6.1 State Transition

Transition of the state estimate between measurements is accounted for by applying the state transition to all the particles of the particle filter. During this operation, the spread of the particles in state space goes from a sample of the prior probability to a sample of the probability density function representing the transitioned state. This was presented in a mathematical form in equation 3.10.

It is assumed that information regarding the forward velocity v and the yaw $\Delta\vartheta$ is available for the localization system. The availability of motion control inputs is generally dependent on the design and the equipment on board a specific robot. Robots equipped with encoders on the wheels may be able to provide information on velocity and displacement. An inertial measurement unit is able to provide accelerations and angular velocities, and a compass may indicate rotation. The motion control input vector u for this implementation is expressed in equation 6.2.

$$u = \begin{bmatrix} v \\ \Delta\vartheta \end{bmatrix} \tag{6.2}$$

The fact that yaw (angular displacement) instead of yaw rate (angular velocity) is used is of little consequence because it is assumed that yaw rate remains constant in a time step of state transition, and yaw is simply the integral of the yaw rate over the elapsed time.

It is necessary to develop a state transition model based on the motion control input vector. For the current implementation of the particle filter, a velocity motion model will be used. The state transition model between the prior state vector $\underline{x}_{t-1}$ and the new state vector $\underline{x}'$ is shown in equation 6.3 [9], where Δt is the time elapsed in the state transition. It is assumed that the control vector remains constant in this duration.

$$\begin{bmatrix} x' \\ y' \\ \vartheta' \end{bmatrix} = \begin{bmatrix} v\Delta t \cos(\vartheta + \Delta\vartheta/2) \\ v\Delta t \sin(\vartheta + \Delta\vartheta/2) \\ \Delta\vartheta \end{bmatrix} + \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \vartheta_{t-1} \end{bmatrix} \tag{6.3}$$

This model is actually a simplification of a more precise model shown in equation 6.4. However, the prior model is simpler to compute and the difference between the two can be considered almost negligible because large changes in heading is not expected over a time step. Furthermore, random noise that will be added to the control input will also make the difference between the two models insignificant.

$$\begin{bmatrix} x' \\ y' \\ \vartheta' \end{bmatrix} = \begin{bmatrix} -\frac{v}{\Delta\vartheta/\Delta t} \sin\vartheta' + \frac{v}{\Delta\vartheta/\Delta t} \sin(\vartheta' + \Delta\vartheta) \\ \frac{v}{\Delta\vartheta/\Delta t} \cos\vartheta' - \frac{v}{\Delta\vartheta/\Delta t} \cos(\vartheta' + \Delta\vartheta) \\ \Delta\vartheta \end{bmatrix} + \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \vartheta_{t-1} \end{bmatrix} \tag{6.4}$$

In order to incorporate uncertainty in the control inputs, zero mean Gaussian noise is added to the control input variables. In other words, the control inputs are modeled as a Gaussian distribution with a mean at the specified control input values. The spread of this distribution is governed by the covariance. It is assumed that the two input variables have no correlation with each other, and that the covariance is constant. The resulting covariance matrix Σis shown in equation 6.5.

$$\Sigma = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\vartheta^2 \end{bmatrix} \tag{6.5}$$

To add random noise to the controls for particle propagations according to a Gaussian distribution, a random number generator that draws randomly from a normal distribution is required. Common random number generators usually generate numbers based on a uniform distribution. The Box-Muller transform [69] can be used to convert a set of uniformly distributed points to follow a normal distribution. Another algorithm for approximate sampling from the normal distribution is shown in equation 6.6 [1]. Again, a uniform distributed random number generator $rand(-b, b)$ is used, where it generates a number between b and $-b$. The value b should be set equal to the standard deviation of the desired normal distribution. This can be obtained by taking the square root of a variance σ_v^2 or σ_ϑ^2. One should find that the output of this method (w) will approximately follow the specified normal distribution with a standard deviation of b.

$$w = \frac{1}{2}\sum_{i=1}^{12} rand(-b, b) \tag{6.6}$$

In figure 6.1, the path taken for testing of the localization system is dotted, and the direction of travel along the dotted path is labeled with arrows. These dots are also representative of the state transition experienced by a particle if it starts at the true initial state, and if the above transition model is used. The time elapsed between each dot is one second, and the entire course lasts 3 minutes and 20 seconds.

Aside from providing expected measurements, the feature map is also used to indicate whether the state transition of a particle has caused it to run into a boundary. A particle is considered to have crashed if the a straight line representing its change in (x, y) position intersects with a building boundary feature on the map. Such a particle will be removed and regenerated with a new random state.

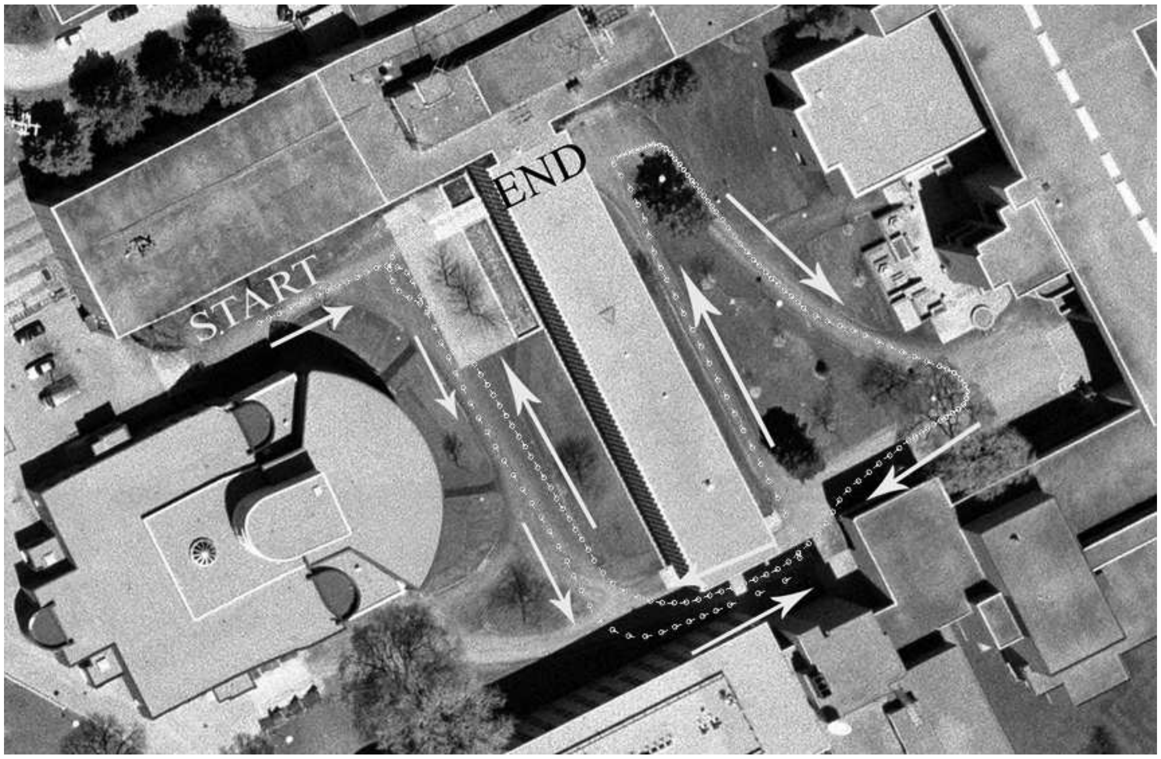


Figure 6.1: The localization test track

6.2 The Importance Factor

The importance factor is a weighting for particles that influence the propagation of the particles into the next iteration of the particle filter. This factor is based on the measurement model described in equation 3.11, and it is proportional to the similarity between the expected measurement (ζ) and the real measurement (z). The expected measurement (ζ) is derived from the current state estimate $\underline{\text{x}}$', after accounting for state transition due to control inputs, and also the feature map. The feature map is created through image processing techniques which have already been presented. The real measurements (z) are obtained from camera images. Using image processing techniques and the vanishing point analysis presented in the previous chapter, information regarding the orientation (vanishing point azimuth) of building boundaries

(ψ) and the bearings (azimuths) of their endpoints relative to the on board camera ($\alpha_{start}, \alpha_{end}$) have been extracted to serve as the real measurement. Equation 6.7 is the mathematical representation of the real measurement. Here, index e is used to identify a section of the field of view. Note again that the measurement for any view section could contain multiple numbers $(1 \ldots r)$ of building orientations, and this was illustrated in figure 5.14.

$$z_e = \begin{bmatrix} \psi_{1 \ldots r} \\ \alpha_{e,start} \\ \alpha_{e,end} \end{bmatrix} \tag{6.7}$$

It is necessary to compute what each particle sees in its field of view according to the feature map to obtain the expected measurement (ζ), as this has not yet been computed. This calculation is performed by examining the orientation of each particle (ϑ) and the orientation of boundaries in a global reference frame. The orientation of a boundary (ϑ_b) can be determined by its endpoints (x_1, x_2), (y_1, y_2) according to equation 6.8.

$$\vartheta_b = \arctan \frac{y_2 - y_1}{x_2 - x_1} \tag{6.8}$$

The expected orientation of a building boundary with respect to a particle (ψ) can then be determined using equation 6.9. This is also the azimuth of the corresponding vanishing point if the boundary is seen from the onboard camera view. For convenience, the calculated value should be expressed between 180^o and -180^o.

$$\psi = \vartheta_b - \vartheta \tag{6.9}$$

The mathematical representation of the expected measurement shown in equation 6.10 is similar to that of the real measurement, which should be of no surprise since the real and expected measurements are to be compared. Here, s is used to index the observable vanishing point azimuths in a view section.

$$\zeta_e = \begin{bmatrix} \psi_{1\ldots s} \\ \alpha_{e,start} \\ \alpha_{e,end} \end{bmatrix} \tag{6.10}$$

The field of view for each particle ($\alpha_{\max} - \alpha_{\min}$) is equivalent to that of the on board camera, which is 48 degrees. To compute the expected visible features, an angular sweepline algorithm is used. The features on the map are first sorted by the bearing (azimuth, α) of its endpoints in ascending order. Endpoints are used because they represent a possible change in the observability of features. Features that are outside the field of view are not included in this list. Starting with the endpoint with the lowest relative bearing to a particle, the corresponding feature is checked to see whether the particle has an unobstructed view of it. The entire length of a feature is considered visible even if a portion of it is occluded by another feature. This is done because information about building height is not available on an aerial map, and the visibility of a tall building behind a shorter building needs to be considered. Features that are fully occluded will not be entered to the list, otherwise the list is updated and the process moves onto the next endpoint. Figure 6.2 is an illustration of this process and the resulting feature list data structure. This data structure is almost identical to the one used for organizing feature data extracted from on board camera images.

The data structure containing visible features from the map and the data structure containing visible features from the on board camera are merged together as shown in figure 6.3. This is done for the convenience of interpreting the feature orientations

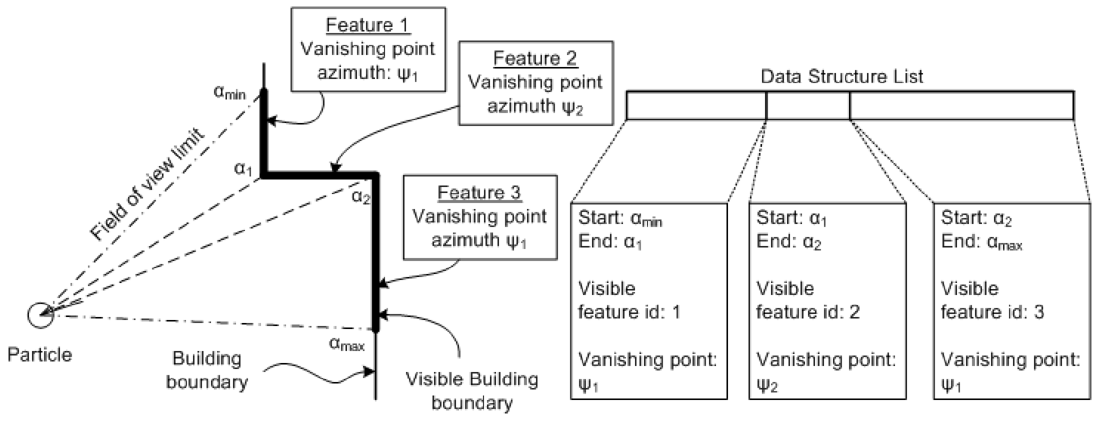

Feature 1
Vanishing point azimuth: ψ1
Feature 2
Vanishing point azimuth ψ2
Feature 3
Vanishing point azimuth ψ1
αmin
α1
α2
αmax
Field of view limit
Particle
Building boundary
Visible Building boundary
Data Structure List
Start: αmin End: α1 Visible feature id: 1 Vanishing point: ψ1
Start: α1 End: α2 Visible feature id: 2 Vanishing point: ψ2
Start: α2 End: αmax Visible feature id: 3 Vanishing point: ψ1

Figure 6.2: Data representation example for map features visible to a particle

in determining the importance factor for each particle. In this combined list, each element still represents a section of the field of view $(\alpha_{e+1} - \alpha_e)$ and contains the expected orientation and the perceived orientation of visible building boundaries in that section. Note that since the objects visible to each particle is different, it follows that the data list will be different for each particle.

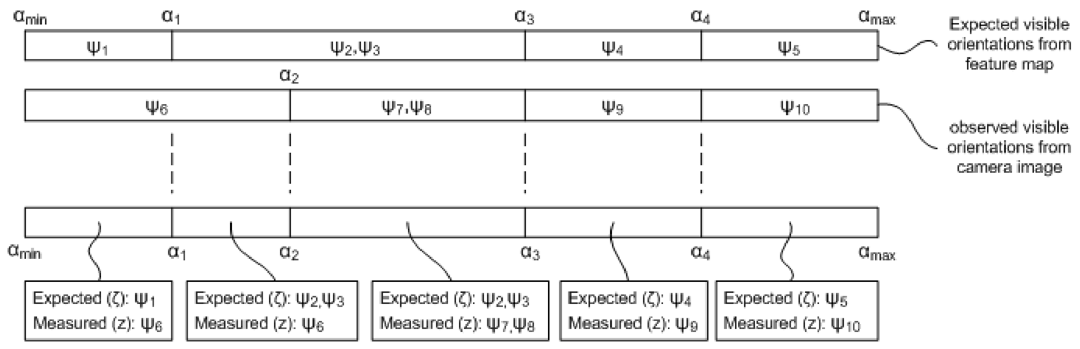

αmin
α1
α2
α3
α4
αmax
ψ1
ψ2,ψ3
ψ4
ψ5
Expected visible orientations from feature map
ψ6
ψ7,ψ8
ψ9
ψ10
observed visible orientations from camera image
Expected (ζ): ψ1 Measured (z): ψ6
Expected (ζ): ψ2,ψ3 Measured (z): ψ6
Expected (ζ): ψ2,ψ3 Measured (z): ψ7,ψ8
Expected (ζ): ψ4 Measured (z): ψ9
Expected (ζ): ψ5 Measured (z): ψ10

Figure 6.3: A combined data representation of expected and real measurements

The importance factor for each particle W_m is evaluated by considering each element of the combined feature data list. For each element e (from 1 to $e_{\max}$), there may be a number of expected orientation measurements ψ_s (expressed in degrees, and indexed by s), and a number true orientation measurements ψ_r (also expressed in degrees, but indexed by r). A matching factor η is determined by considering the closest pair of expected orientation measurement and true orientation measurement. This closeness is quantified using a sigmoid function, as expressed in equation 6.11. A sigmoid function is used so that slight differences in expected and observed measurements can still result in a high importance factor, but the penalty in the difference will become increasingly severe if measurement differences increase.

$$\eta_e = \max_{r,s} \left[1 - \frac{1}{1 + \exp\left(\frac{20 - |\psi_s - \psi_r|}{2}\right)} \right] \tag{6.11}$$

A weighting factor ρ_e is determined for each data list element. This factor is proportional to the view span section ($\alpha_{e,end} - \alpha_{e,start}$) it represents in the complete field of view ($\alpha_{\max} - \alpha_{\min}$), and is calculated according to equation 6.12. This factor will ensure that the feature comparison result for a view section that spans across a large proportion of the field of view will receive more weighting compared to the feature comparison result of a section that covers a relatively smaller proportion of the field of view.

$$\rho_e = \frac{\alpha_{e,end} - \alpha_{e,start}}{\alpha_{\max} - \alpha_{\min}} \tag{6.12}$$

The importance factor for a particle is then calculated using equation 6.13

$$W_m = \sum_{e=1}^{e_{\max}} \rho_e \eta_e \tag{6.13}$$

6.3 Particle Resampling

In a particle filter, the sampling variance concerning the state of each particle compared to the true state can increase during the resampling process, even though the variance of the particle set itself appears to decrease. This occurs because of the reduction in particle diversity during resampling, and the random nature of the resampling process [1].

To reduce the sampling error, the low variance sampling method is used, where each selection of particles in resampling is not an independent process [1]. This implies that for M particles, selection is not based on generating M random numbers. An illustration of how the low variance sampling method works is shown in figure 6.4. In this figure, each cell represents a particle and the importance factor determines the width of each cell. A random number R is generated between zero and M^{-1}, upon which the value M^{-1} is repeatedly added for (M-1) iterations. The value of the sum during each iteration is used to select the particles that will make the next iteration of the particle filter.

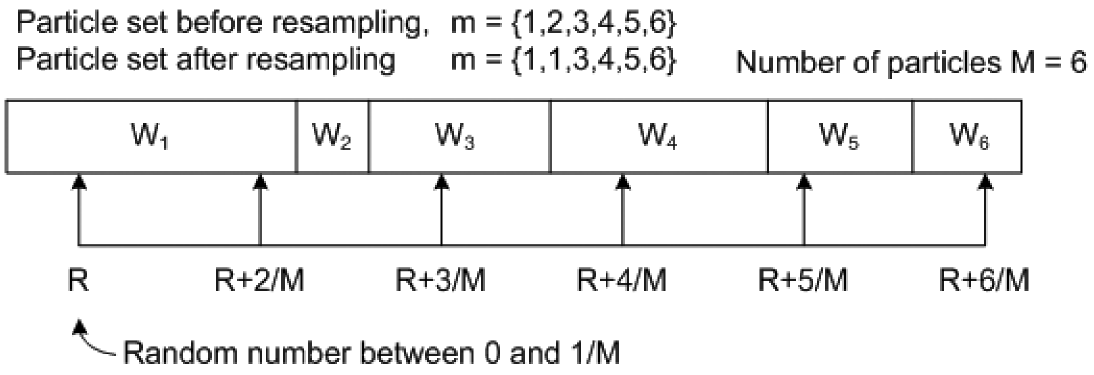

Particle set before resampling, m = {1,2,3,4,5,6}
Particle set after resampling m = {1,1,3,4,5,6}
Number of particles M = 6
W1
W2
W3
W4
W5
W6
R
R+2/M
R+3/M
R+4/M
R+5/M
R+6/M
Random number between 0 and 1/M

Figure 6.4: Low variance sampling [1]

The low variance sampling method is computationally more efficient compared to generating M number of random numbers. Additionally, in the case where all particles have the same importance factor, all particles will be resampled and used in the next iteration of the particle filter. The same result is not achieved by generating M independent random numbers.

6.4 Localization Results

The result of localization for an unknown initial state is greatly influenced by the number of particles used. As discussed when the particle filter was first presented in section 3.3, the particle deprivation problem can exist even when a large number of particles is used because the random nature of the particle filter may not generate a particle close enough to the true state. However, the problem is usually more pronounced with a low number of particles. Therefore, the selection of the particle set size should ensure that this problem does not occur frequently. Determining the number of particles to use is also dependent on the size of the state space (which in this case has three dimensions). Through numerous trials with different sizes of particle sets, localization has been achieved with as little as 300 particles. However, to assure a high likelihood of convergence, it is determined that 2000 particles should be used for the workspace defined by the aerial feature map. It should be noted that the variance of the control inputs also have an influence on the ability to localize, but this will be discussed in the next section. For the results presented in this section, the goal is to illustrate that localization can be achieved and to quantify the performance.

Over 100 test runs have been conducted on the particle filter to ensure that localization was not achieved by pure chance. However, since it is impractical to show time frame by time frame results of every test run. The results of a typical test run will be shown here in a series of figures depicting the result at various points along the test run. Additional results can be found in appendix C to provide further proof of the localization system's validity.

Figure 6.5 illustrates the start of a test run for which the standard deviation of the forward velocity is set at $10\frac{cm}{s}$, and 5 degrees for yaw. Since the initial true state is unknown, particles are spread randomly throughout the workspace. The direction in which a particle is pointed is indicated by a short line protruding from the center of each particle. The true state is also shown.

Figures 6.6 through figures 6.16 shows the evolution of particles as the test track is traversed. At first, particles are seen to start converging at several locations although there is only one true state. This is occuring because of sensor aliasing, where the expected measurement at different states appear similar to the real measurement. As state transitions occur, particles move into different areas of the workspace where they can further determine if corresponding expected measurements match real measurements. Hence the visible clusters where particles converge begin to reduce until only one remains. As evident from figure 6.15, the particle cluster overlaps the true state when localization is achieved.

In looking at the many test runs performed, the time at which particles converge onto a single cluster near the true state is unpredictable due to the random nature of the particle filter. Additionally, in some instances, particles are still found grouped into several clusters at the end of a test run, while at other times, a single cluster is formed at an incorrect location due to the particle deprivation problem. Although both cases are infrequent with the number of particles used, it is believed that the particles will eventually converge around the true state given enough time. The reason for this is because a particle at an incorrect state will likely to eventually run across a building boundary during state transition. When this occurs, the design of the particle filter causes the particle to regenerate at a random state, which by chance

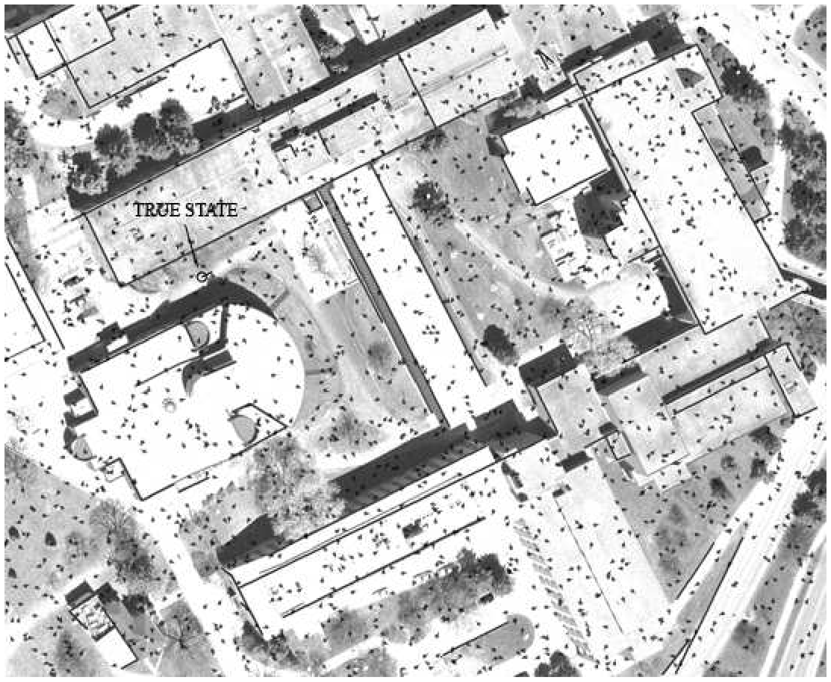


Figure 6.5: Particle filter localization result - sequence 1 of 12 - elapsed time: 0s

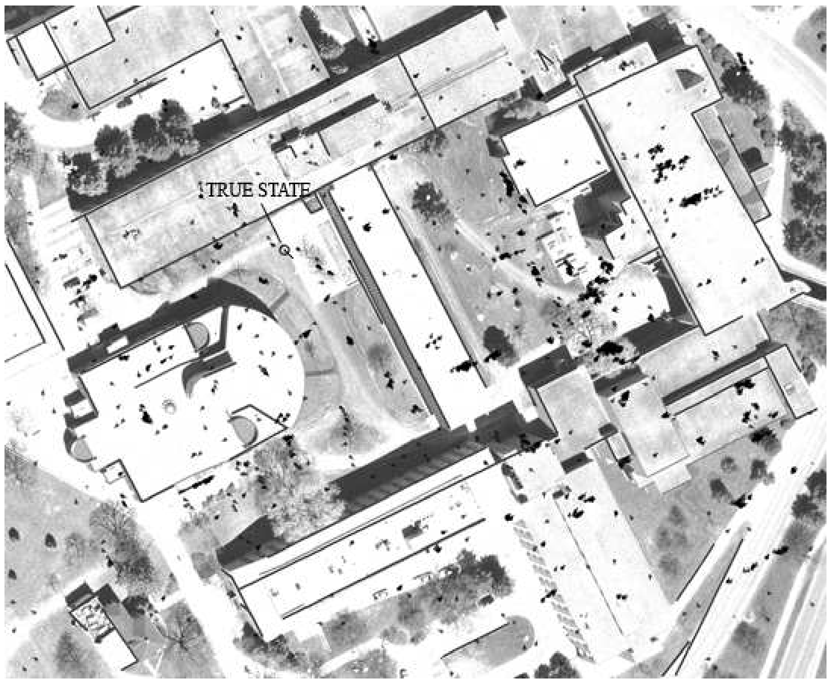


Figure 6.6: Particle filter localization result - sequence 2 of 12 - elapsed time: 15s

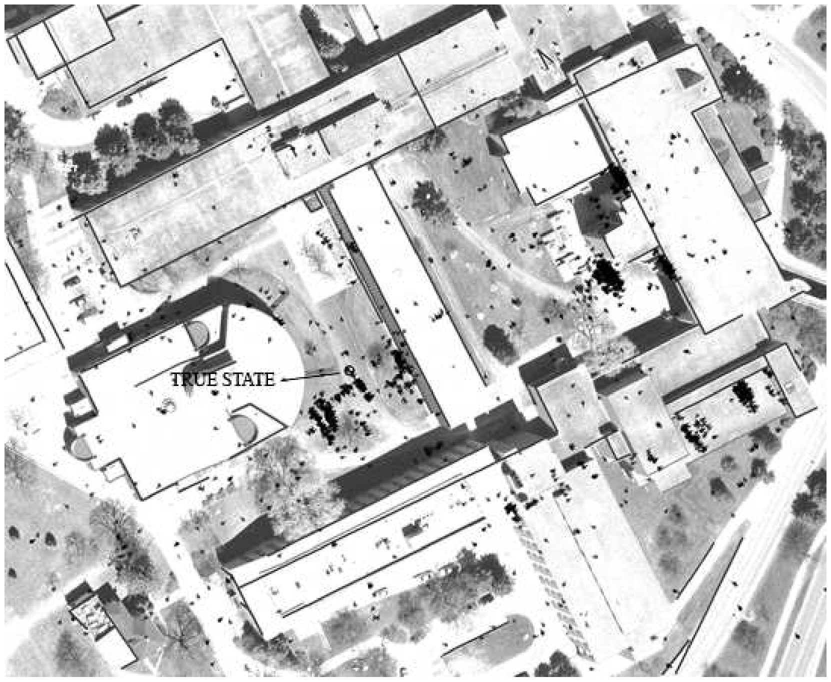


Figure 6.7: Particle filter localization result - sequence 3 of 12 - elapsed time: 30s

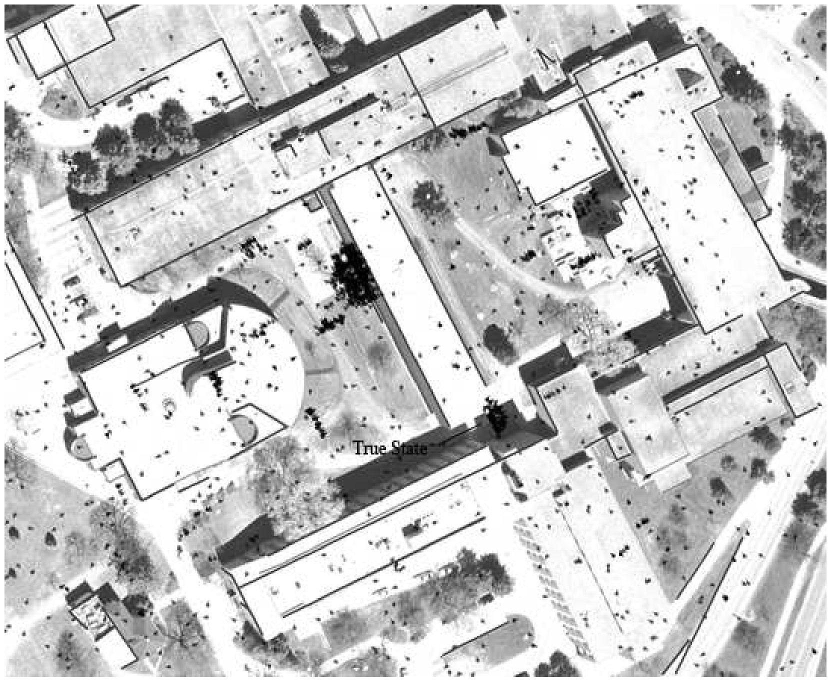


Figure 6.8: Particle filter localization result - sequence 4 of 12 - elapsed time: 45s

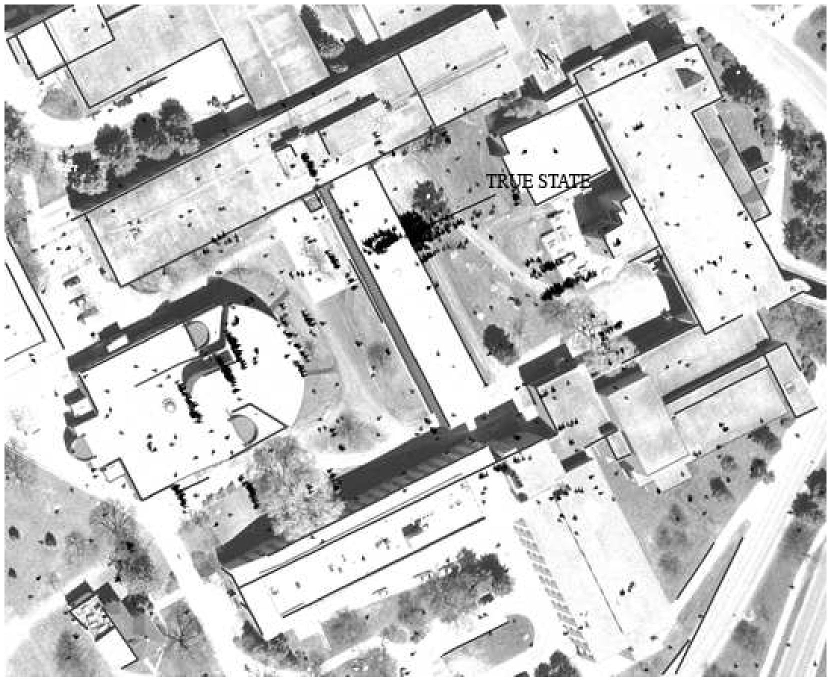


Figure 6.9: Particle filter localization result - sequence 5 of 12 - elapsed time: 60s

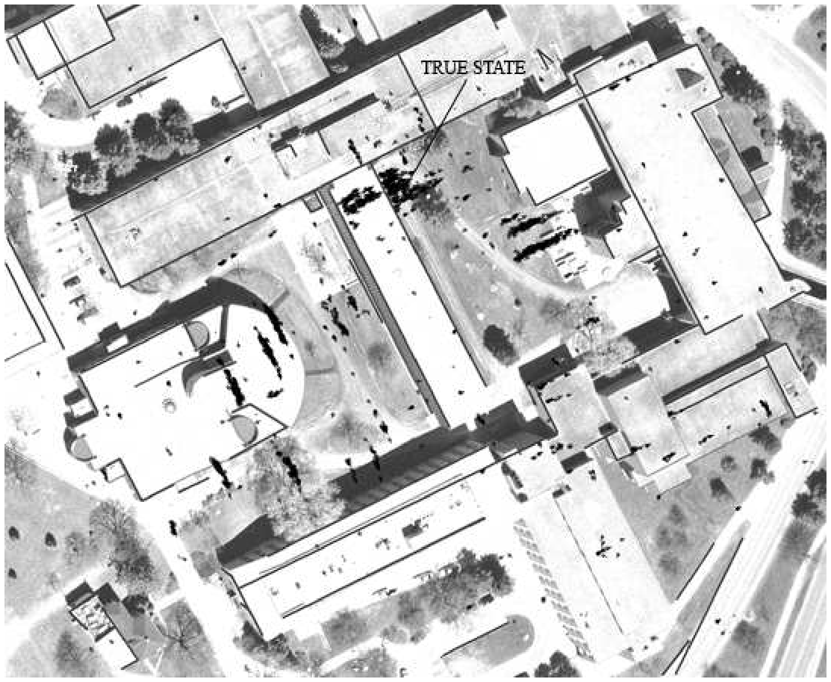


Figure 6.10: Particle filter localization result - sequence 6 of 12 - elapsed time: 75s

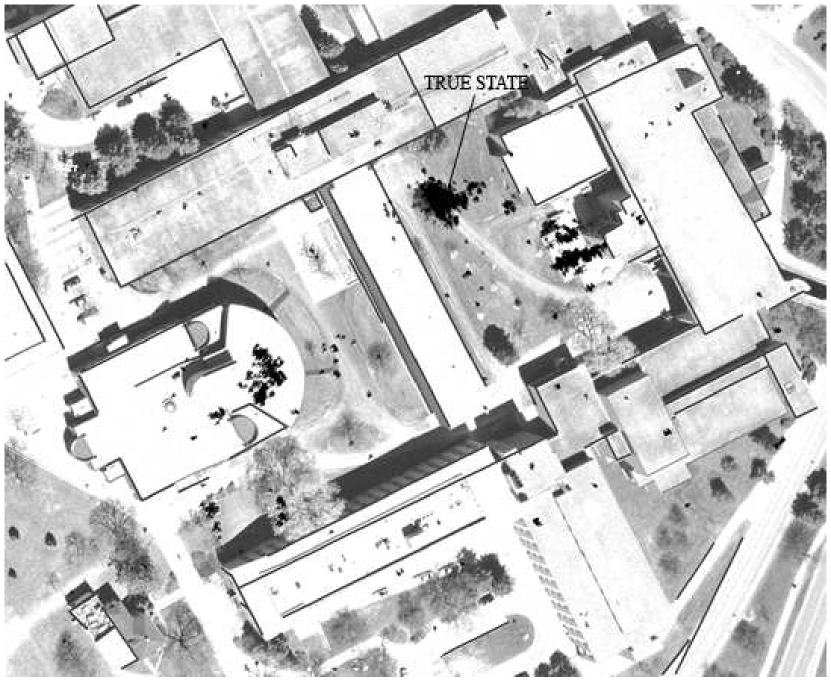


Figure 6.11: Particle filter localization result - sequence 7 of 12 - elapsed time: 90s

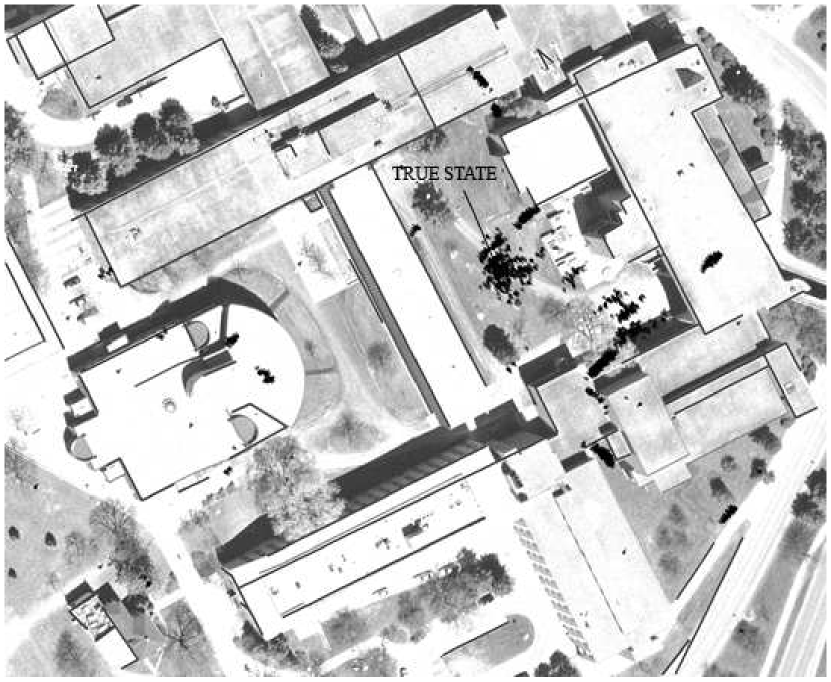


Figure 6.12: Particle filter localization result - sequence 8 of 12 - elapsed time: 105s

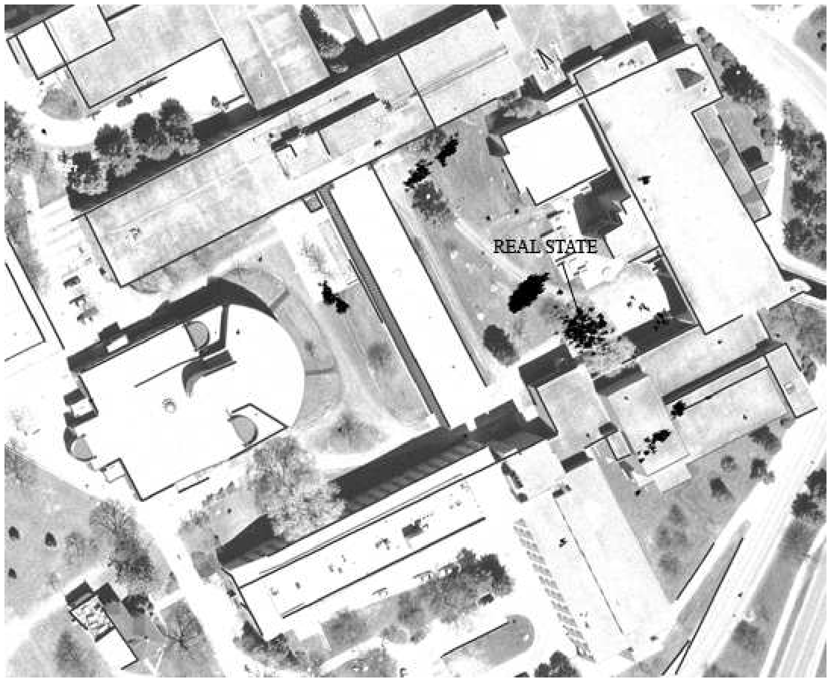


Figure 6.13: Particle filter localization result - sequence 9 of 12 - elapsed time: 120s

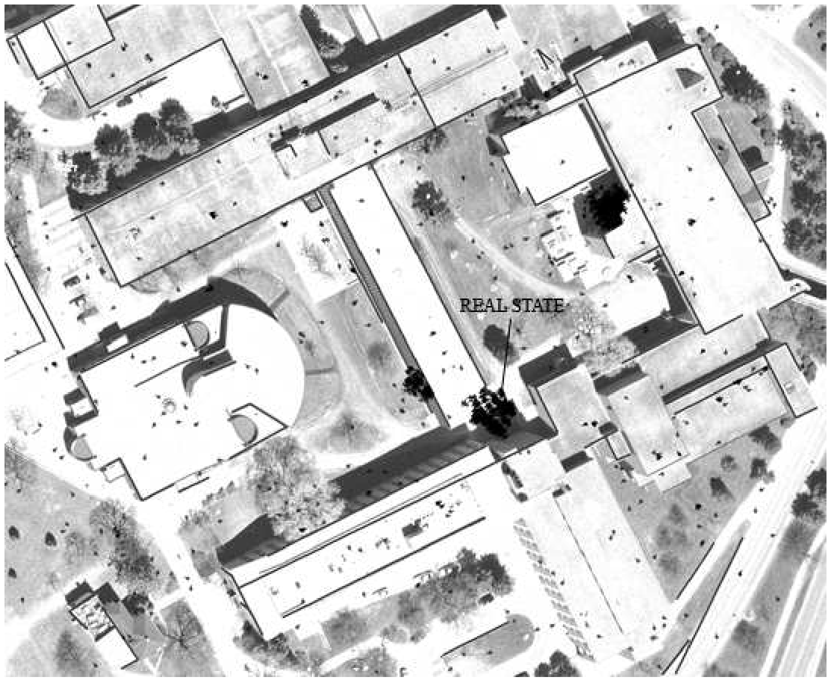


Figure 6.14: Particle filter localization result - sequence 10 of 12 - elapsed time: 135s

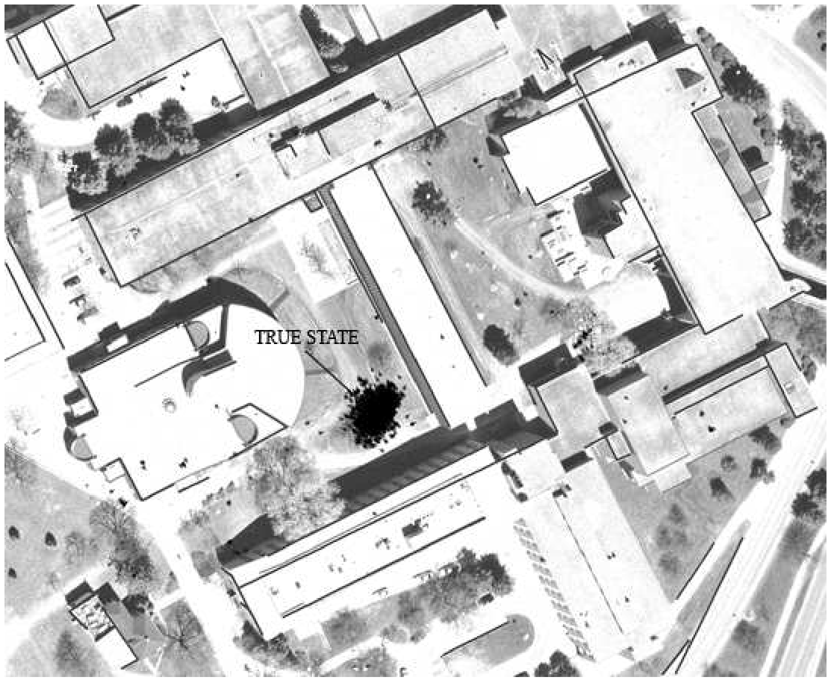


Figure 6.15: Particle filter localization result - sequence 11 of 12 - elapsed time: 150s

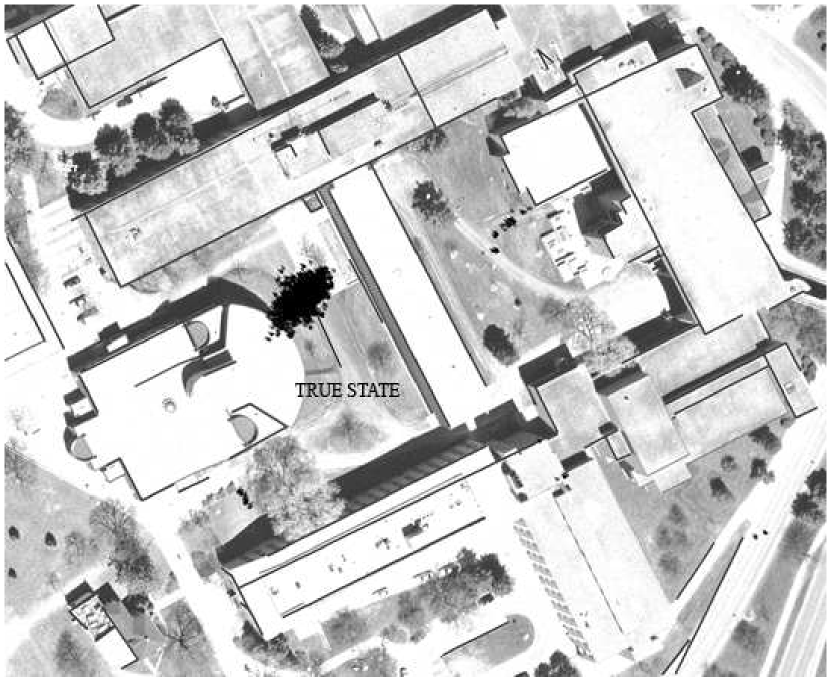


Figure 6.16: Particle filter localization result - sequence 12 of 12 - elapsed time: 175s

may be near the true state. This design can be considered as a counter measure for the particle deprivation problem.

To quantify the performance of the particle filter, the difference in (x, y) position between the true state and the estimated state is measured. However, it is only fair to apply this distance measure when the particles have converged at a single cluster, signifying that that system thinks it has localized. Visually determining this point may induce some degree of bias. Therefore, a measure of particle spread will be used to indicate when the system thinks it has achieved convergence. The error is then averaged over time, and also over randomly chosen test runs.

The measure of particle spread is performed by assuming a Gaussian distribution and determining the covariance of particle locations. The covariance matrix is then diagonalized by finding its eigenvalues [43], as in a principal axes analysis. When the square root of the larger eigenvalues is below the threshold of $10m$ as shown in figure 6.17, localization is considered to be achieved.

Error measurements averaged over 10 test runs indicate that the average positioning error of the localization system is $5.3m$, with 95% of the error measurements being below $9.3m$. This result is comparible to that of SPS for GPS, the performance of which was presented in section 2.3. It was stated that the positioning accuracy of $22m$ can be achieved with SPS. But realistically, the accuracy is at about $8m$ with the absence of SA. GPS positioning accuracy can also be increased due to the availability of WAAS, but at the same time, multipath in urban environments bring unpredictable effects. The DOP readings (used as an estimate of the effect of multipath) taken from a GPS receiver around the the area of the test course indicates an average value of approximately 2.5. Using this multiplicative factor on the UERE, it is estimated that on average, the GPS positioning error for the test site is within $20m$ for 95% of the position readings. Therefore, strictly based on positioning accuracy,

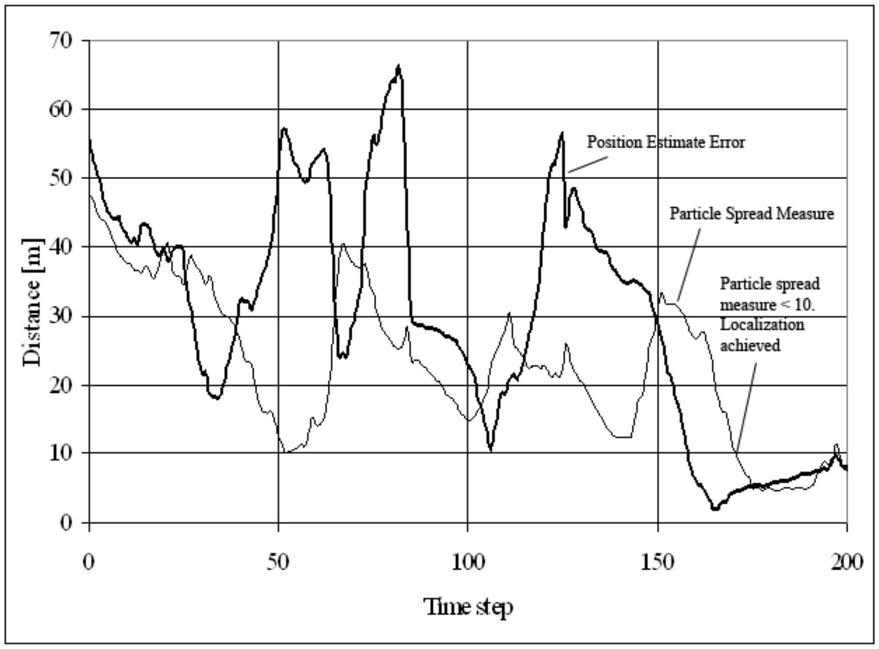


Figure 6.17: Indication of particle convergence in a typical test run

the performance of the vision based localization system is slightly better than the performance of GPS. Aside from positioning measures, another consideration is the time required to achieve localization. With GPS receivers, if the unit has recently been used, it is possible to obtain positioning measurements the instant the receiver is activated. This is of course provided that the surrounding environment allows for a positional fix. With the particle filter, localization requires some degree of movement in the environment, and the time it takes to localize is not deterministic.

The results presented so far indicate that the particle filter is able to perform state estimation with unknown initial state. Additional tests were performed to see how the particle filter handles the robot tracking problem where the initial state is known. This should be an easier problem and less particles should be required because it is no longer required to cover the entire state space with particles in the beginning. The purpose of performing this test is to ensure that once localization has been achieved (indicated by the convergence of particles to one cluster), localization can be maintained (or the single particle cluster does not dissipate into multiple clusters). This test was performed in several trials with different sizes of particle sets, the lowest of which is 10, the highest of which is 300. In all cases, localization was maintained throughout the test course.

6.5 Sensitivity to Input Uncertainties

It is necessary to examine how the particle filter reacts to different settings of the control input covariance to ensure that it can still perform with realistic amounts of errors in state transition. Additionally, it is of interest to find the performance limits of the particle filter design.

Since it was assumed that the two control inputs have Gaussian noise and are independent, the control input covariance matrix is diagonal and therefore the only concern is the two values representing the variance (the square root of which is the standard deviation) for forward velocity and yaw. To interpret the magnitude of the standard deviations in a statistical sense, it means that approximately 95% of the time, one can expect the control input error to be within two standard deviations. Furthermore, the errors will be within three standard deviations for approximately 99% of the time.

Different combinations of control input standard deviations (σ_v,σ_ϑ) were tested with the particle filter, and the resulting success rate of localization is summarized in table 6.1. For the combinations that were tested, five tests for each case were repeated and the number of successful localization attempts within the duration of the test track are recorded. The same definition of when the system thinks it has localized will be used as in the previous section.

	$\sigma_\vartheta = 2^o$	$\sigma_\vartheta = 5^o$	$\sigma_\vartheta = 7^o$
$\sigma_v = 0.1\frac{m}{s}$	5/5	5/5	2/5
$\sigma_v = 0.2\frac{m}{s}$	5/5	3/5	1/5
$\sigma_v = 0.5\frac{m}{s}$	4/5	3/5	2/5
$\sigma_v = 0.7\frac{m}{s}$	1/5	2/5	0/5

Table 6.1: The effects of control input variance on the localization performance (success rate) of the particle filter

	$\sigma_\vartheta = 2^o$	$\sigma_\vartheta = 5^o$	$\sigma_\vartheta = 7^o$
$\sigma_v = 0.1\frac{m}{s}$	$\varepsilon = 2.1m, \sigma = 5.7m$	$\varepsilon = 3.2m, \sigma = 7.7m$	$\varepsilon = 4.0m, \sigma = 12.1m$
$\sigma_v = 0.2\frac{m}{s}$	$\varepsilon = 2.8m, \sigma = 6.4m$	$\varepsilon = 4.7m, \sigma = 11.3m$	$\varepsilon = 6.5m, \sigma = 15.8m$
$\sigma_v = 0.5\frac{m}{s}$	$\varepsilon = 2.6m, \sigma = 9.3m$	$\varepsilon = 3.0m, \sigma = 15.7m$	$\varepsilon = 6.8m, \sigma = 17.3m$
$\sigma_v = 0.7\frac{m}{s}$	$\varepsilon = 1.8m, \sigma = 13.7m$	$\varepsilon = 8.4m, \sigma = 15.0m$	$\varepsilon = 8.5m, \sigma = 23.8m$

Table 6.2: The effects of control input variance on the localization performance (positional error and particle spread) of the particle filter

It is important to note that in some of the cases described in the table above, such as the case where $\sigma_v = 0.7\frac{m}{s}$ and $\sigma_\vartheta = 2^o$, a single cluster of particles were observed to have formed in all instances, but the spread of the clusters observed were too great for the system to considered itself to be localized. This same observation can be said for some of the test runs for the cases where $\sigma_v = 0.1\frac{m}{s}$, $\sigma_\vartheta = 7^o$. For this reason, table 6.2 shows the average minimum position error ε (in meters) and the minimum spread expressed as standard deviation σ (in meters) for each case. The minimum values obtained in a trial are used in the table instead of a timed average to avoid having to visually judge and determine when the system seems to have localized, so that bias will not be introduced into the results.

Overall, it can be seen how performance is degraded with increasing control input variances. With greater variances, particles that find themselves near the true state are more likely to move away from the intended course, which results in lower importance value, and the greater cluster spread observed.

Another interesting point to note from the table is that changes in the variance for yaw seem to have more effects than that of forward velocity. This can be explained from the design of the particle filter. In the particle filter, the measurement being used to evaluate the importance factor for particles is based on the orientation of building walls and their relative bearing to the particles. When a particle moves in a straight line, the expected orientation of the building wall will not change, but there might be slight changes in the bearings of feature endpoints. On the other hand when a particle rotates, both the apparent orientation of building walls and their bearings change. Therefore, this has a more pronounced effect on the expected measurements and the calculation of importance factor.

Overall, it is concluded that the performance of the particle filter can be maintained when the standard deviation for forward velocity is below $0.1\frac{m}{s}$ and when that of yaw is below 5^o.

6.6 Timing analysis

The testing of the particle filter was performed on a computer with an Intel Pentium M processor rated at $1.60GHz$. Process timing information for an iteration of the particle filter has been gathered over many test runs (using 2000 particles) and statistically summarized in table 6.3. Since images are acquired from the on board camera every one second, it implies that an equivalent cycle processing time required for real time implementation. In the table, the localization process has been broken down into several key steps to give an indication of which sub-processes are more

	Average time [s]	Standard deviation [s]
State transition	0.165	0.070
Camera image processing	1.236	1.736
Importance factor calculation	7.751	5.865
Particle resampling	0.001	0.004

Table 6.3: Particle filter localization process timing summary

computationally demanding, and a graphical representation on the average timing information is also presented in figure 6.18.

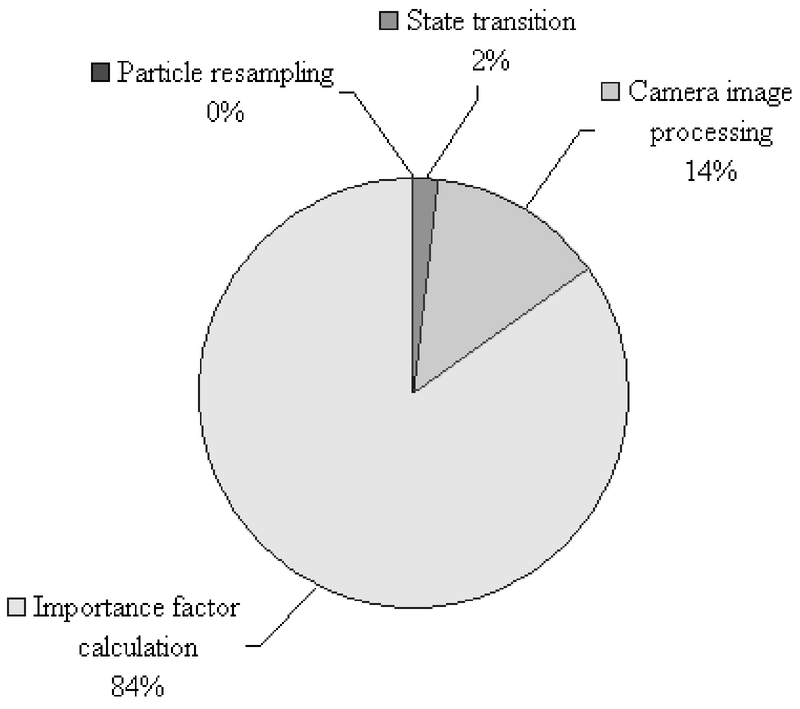


Figure 6.18: A comparison of computational demand for sub-processes of the particle filter based on mean timing information

The computation for state transition takes a relatively small share in an iteration of the particle filter. Within this sub-process, the majority of the time is spent on determining if particles have collided with a boundary, while the time required to compute state transitions alone is almost negligible.

Processing of camera images take up a noticeable portion of time in the particle filter algorithm. The actual time spent is dependent on the complexity of a scene and the number of features extracted. It is expected that the image processing processes steps of edge detection, filtering, and Hough transform require little computation time in comparison with the pattern recognition process of finding line segment feature intersections as well as clustering.

The step that takes the most time is the calculation of importance factor. This is due to the need for each particle to determine which features are within its field of view. Therefore, the time required in this step is dependent on the complexity of the feature map, and also the number of particles used, both of which have a linear relationship with the computation time.

The time required to resample particles takes the least amount of time and is almost negligible compared to the three other steps.

Overall, the results show that a faster processing unit, or improvements in the efficiency in the processing of aerial images as well as determination of the importance factor is required to achieve real time implementation.

Chapter 7

Conclusions

In this thesis, a design for an urban environment autonomous localization system has been proposed. This system is unique in that it uses monocular vision as the only form of sensing, which makes the system simpler in terms of hardware management and thus less costly. An aerial orthoimage is used as a map of the localization system to increase the degree of autonomy.

One of the main challenges was extracting accurate information from limited visual cues. Environmental information was extracted from aerial images through the use of image processing and computer vision techniques. While a feature map defined by a human user may be more accurate, the proposed design introduces automation to the process, which can drastically reduce the time involved in building a map that is comprehensible by a computer. Overall, this makes the localization system more flexible, as it can be implemented quickly in many places as long as an aerial map is available.

Image processing and computer vision techniques were once again utilized to extract information from on board camera images, with an additional process of vanishing point analysis. This enabled the orientation of building boundaries in the field of view to be estimated. Using the vanishing point analysis also helped reduce potential problems due to camera movement (tilting and rolling).

Another challenge of this localization system design was in placing the information from the aerial image and on board camera images to good use. The particle filter was implemented to update state estimates by comparing observed building boundary orientations to expected values, while using a velocity motion model to calculate state transitions. Testing of this system was performed off line with a set of previously saved on board camera images.

The numerous test results show that localization can be achieved using the proposed design for the problem where the initial state is unknown. The state space in which localization was performed corresponded to an actual large area. Furthermore, besides estimating the two coordinates for position, the localization system was required to estimate heading, a third dimension in the state space. All this was achieved using only 2000 particles and with the incorporation of realistic control input noise. Furthermore, the particle filter also works for the robot tracking problem. Performance analysis revealed that the designed system is comparable to GPS in terms of estimation error and can outperform it when multipath becomes a problem. However, the particle filter has the drawback of taking a longer time to localize, and the biggest drawback is the currently achievable computational time using an inexpensive portable computer.

With the successful implementation of the particle filter, it is confirmed that localization can be achieved with an imperfect map, which reconfirms that a Bayes filter approach is robust. Also, it is confirmed that localization is possible using purely monocular vision. In addition, the results of the particle filter indicate that the design is able to tolerate the uncertainties of working in an outdoor environment and can be used as a bench mark for future vision based localization systems.

7.1 Future Work

Using a single camera as the only sensor brings the benefit of simplicity in hardware management. This can be further extended by using vision to estimate the control inputs for state transition. This is known as the vision based ego-motion estimation problem, and may have the potential to eliminate the need for proprioceptive sensors. Many vision based ego-motion estimation system will first compare two images by calculating the optical flow, or the distribution of apparent displacement of intensity patterns between two images [70]. Based on the vectors of optical flow, an ego-motion estimator will try to determine the motion that has caused the displacements between two scenes. For additional information and algorithms, refer to [71, 72, 73, 74, 75].

The performance of GPS was briefly discussed in this thesis. GPS remains a popular tool in localization, regardless of the problems which it might encounter in urban settings. For instance, the degrading of GPS performance in urban canyons has not discouraged their use in automobiles. With decreased accuracy, it may still be possible to incorporate GPS readings to reduce the size of the state space where particles should be seeded. To elaborate on this idea, a GPS position reading can be used to increase the importance factor of particles near the perceived position fix. The amount of increase will depend on the confidence of the GPS measurement. A high confidence will only retain particles close to the position fix, while a low confidence will have less influence on the particles. From this, the implication is that fewer particles will have to be used even for a large workspace, and the approach should become increasingly beneficial as the size of workspace increases.

Although the particle filter implementation was successful, there is room for further improvements. One way to increase the robustness of the particle filter is to allow the importance factor to accumulate for several iterations before resampling. This approach is effective against random noise that may appear in measurements [1]. In terms of the designed localization system, this noise may come from errors in the vanishing point analysis and from errors on the feature map.

The localization timing analysis revealed that real time implementation is not possible without a faster computer processor. Besides better hardware, perhaps changes can be made to increase the efficiency of the particle filter algorithm.

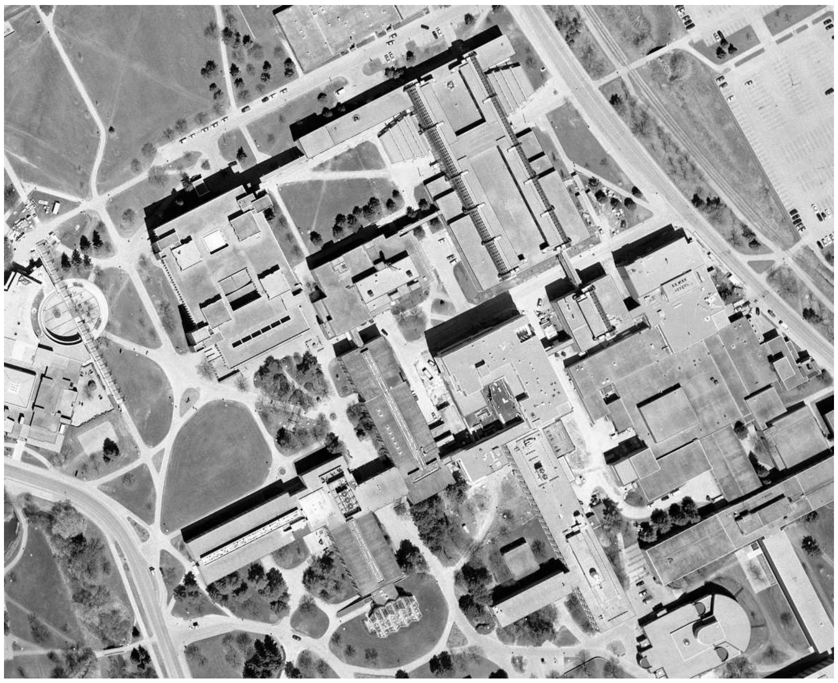
Depending on the application, it may be possible to reduce the measurement update frequency, or in other words, reduce the frequency at which on board camera images are captured. Another possibility is to first evaluate the usefulness of a captured image before using it to update state estimates. For instance, in a previously captured camera image, a building wall feature may be indicated as being perpendicular to the camera heading. The same information may be present in the current camera frame, which is unlikely to cause any large changes in the distribution of particles. Therefore, to save computational time, the image may be disregarded. A scene may also be considered to be discarded if it contains too little information. The time saved from not updating particles will mainly come from not having to reference the feature map to determine expected measurements.

Since the size of the particle set heavily influences the efficiency of a particle filter, it may be a good idea to use an adaptive particle set size. The idea behind an adaptive particle set size is to reduce the number of particles once they begin to converge upon a certain state since the particles are no longer required to cover a large area in state space. One implementation of this is known as KLD (Kullback-Leibler Divergence) sampling [1].

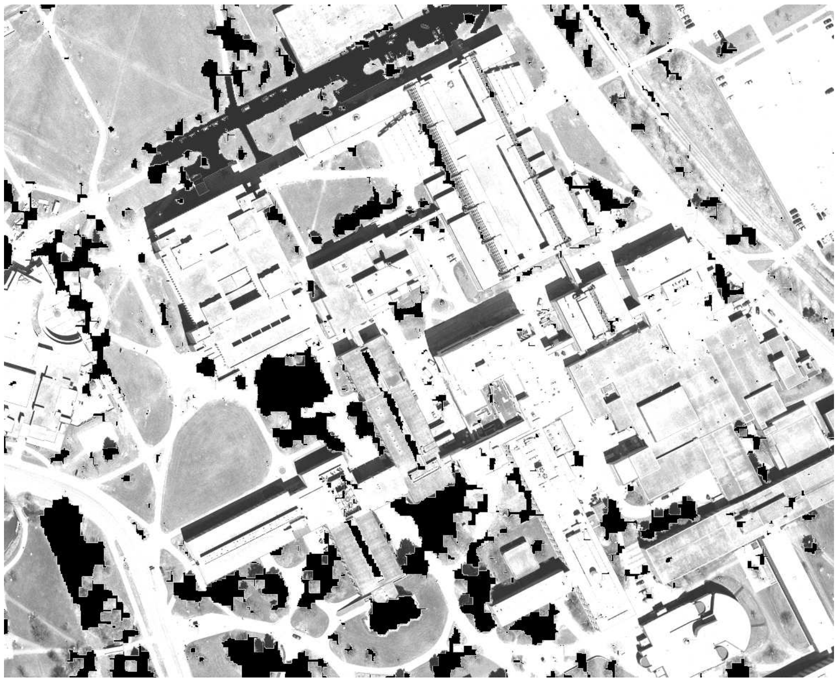
Appendix A

Feature Map Processing Examples

This Appendix section contains additional figures showing the results of corner response masking process and the modified PPHT process. Its intent is to indicate that the process described for feature map generation will work on other aerial images other than the one of the localization testing area.

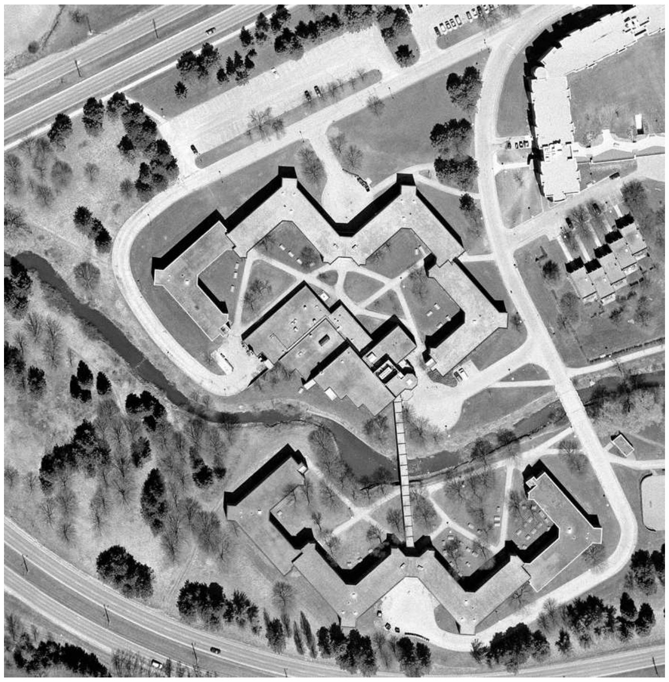


(a) Original image

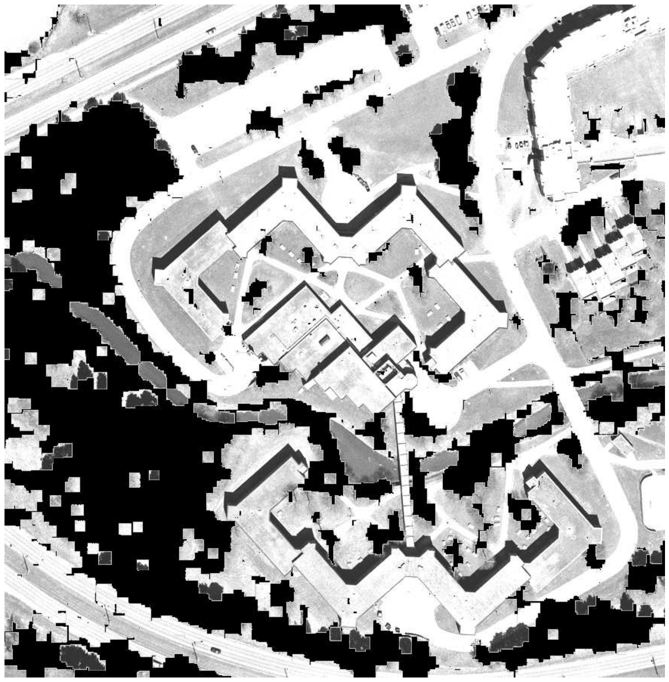


(b) Masked image

Figure A.1: An additional example on corner response / distraction / vegetation masking

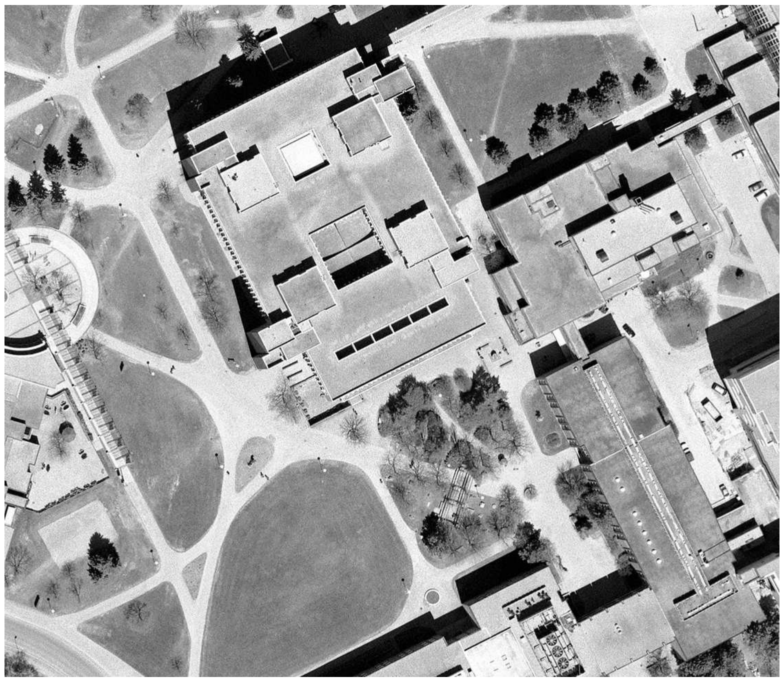


(a) Original image



(b) Masked image

Figure A.2: An additional example on corner response / distraction / vegetation masking


(a) Original image

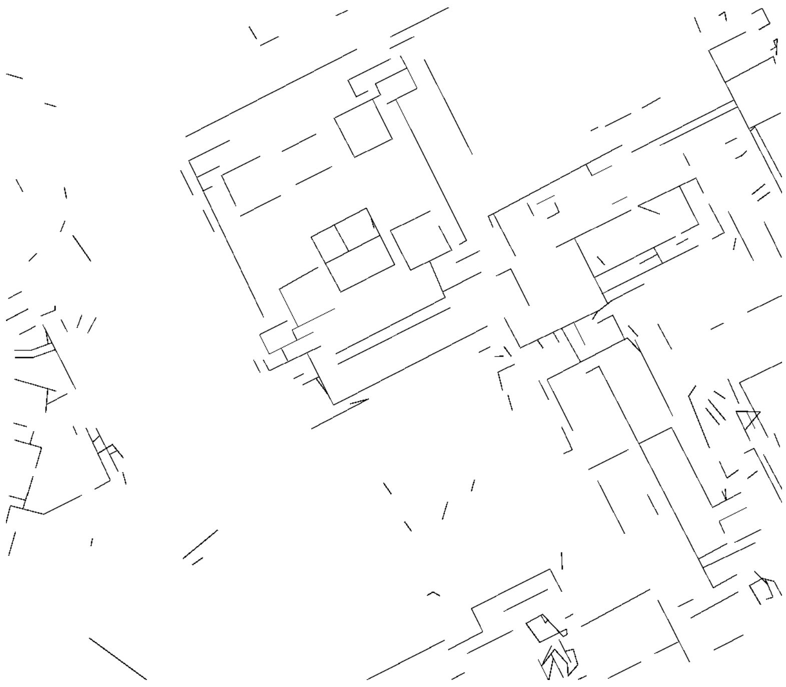

(b) Processed image

Figure A.3: An additional example on the modified PPHT process

Appendix B

Camera Image Processing Examples

Additional examples and figures are presented here to supplement the results shown in the Camera Measurements chapter. For each of the original camera image, several corresponding figures will be shown including: the edge map, the density measure, the edge map distraction mask, the filtered edge map, and the vanishing point analysis results. In the vanishing point analysis results, the features on the image associated with a particular vanishing point is colour coded. The first is in red, the second is in green, and the third is in blue. Furthermore, the azimuth and the elevation of these vanishing points are also shown in the figures. Features that are coloured black are ones that failed to be associated with any vanishing point.

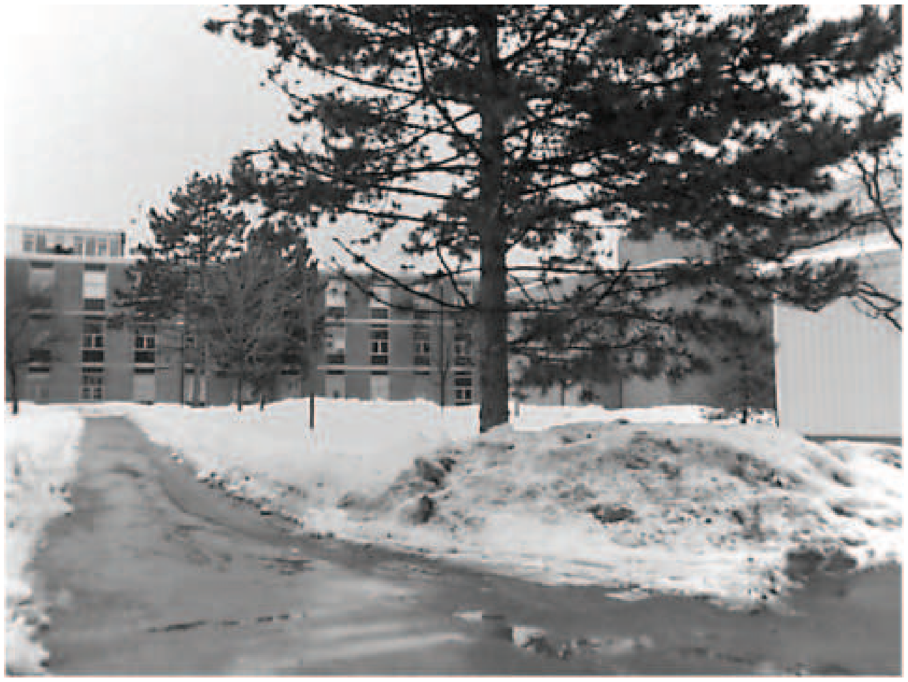


Figure B.1: An image captured by the camera for localization

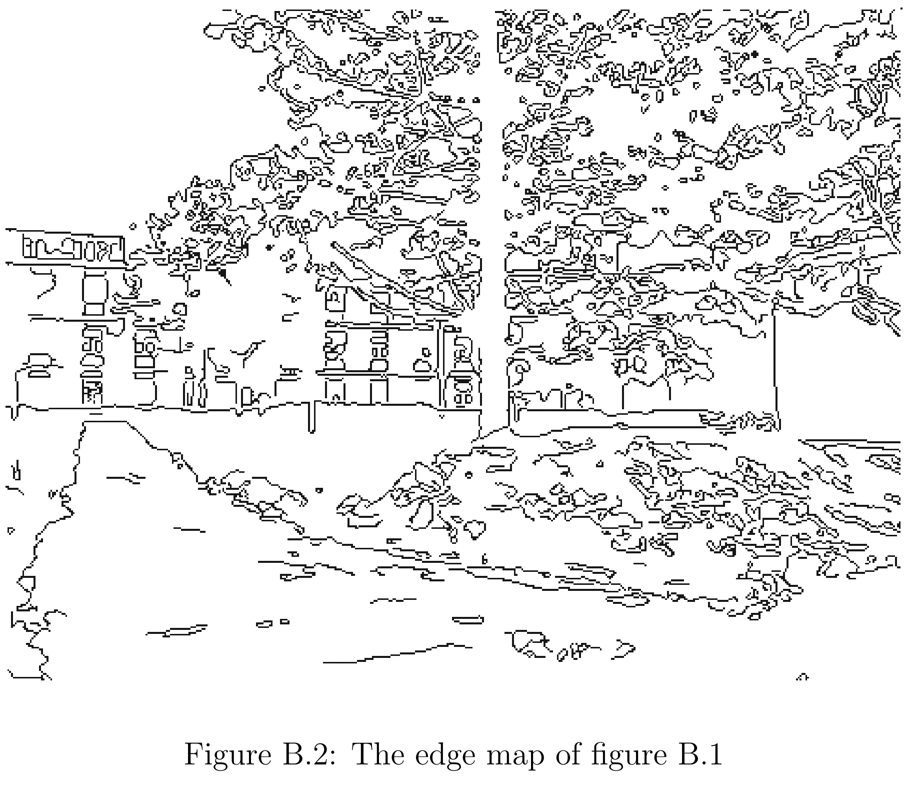


Figure B.2: The edge map of figure B.1

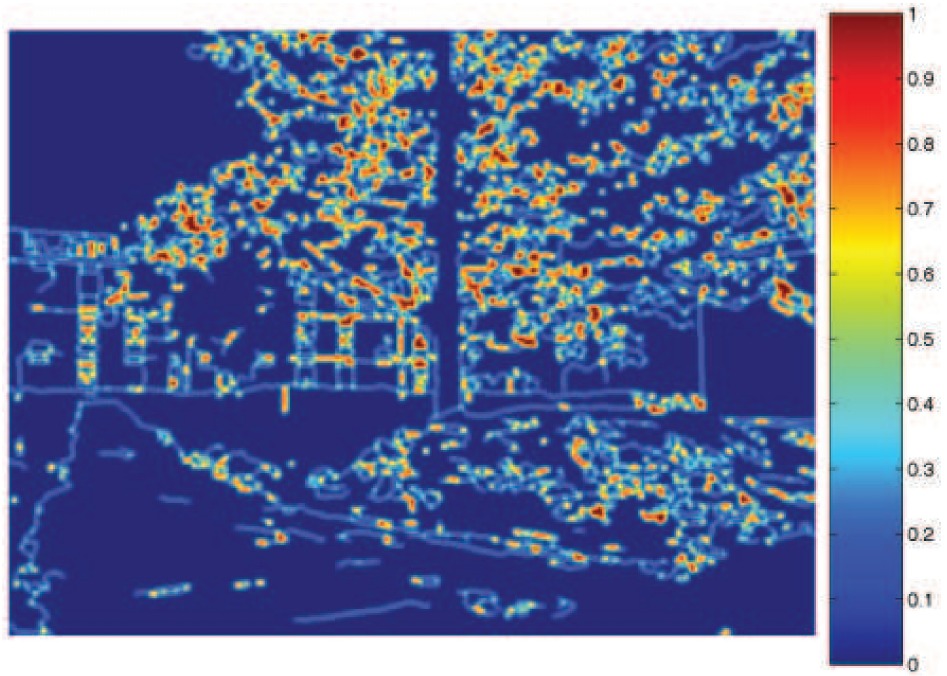


Figure B.3: The edge map density measure for figure B.1

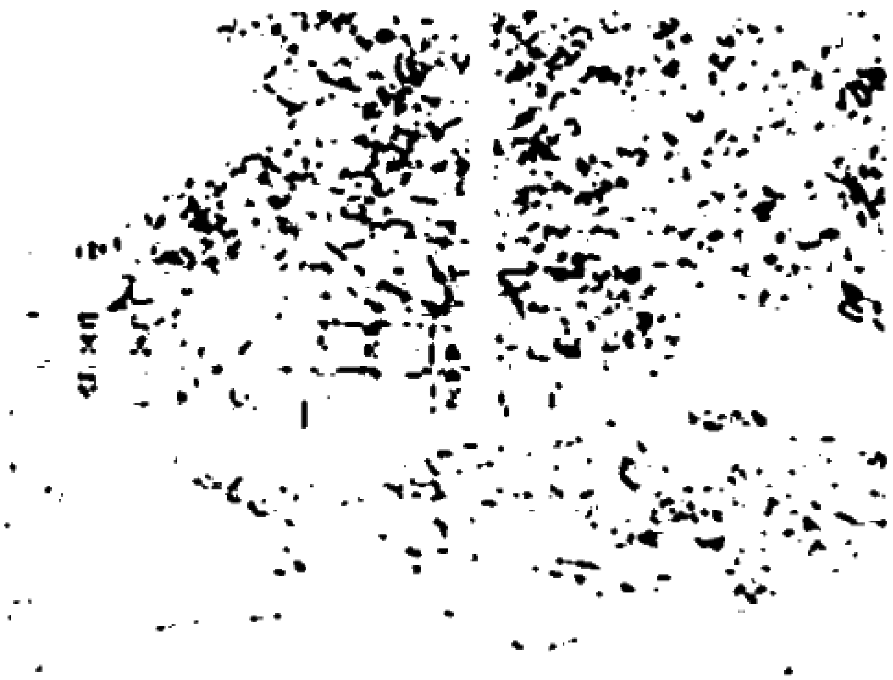


Figure B.4: The edge response mask for figure B.1

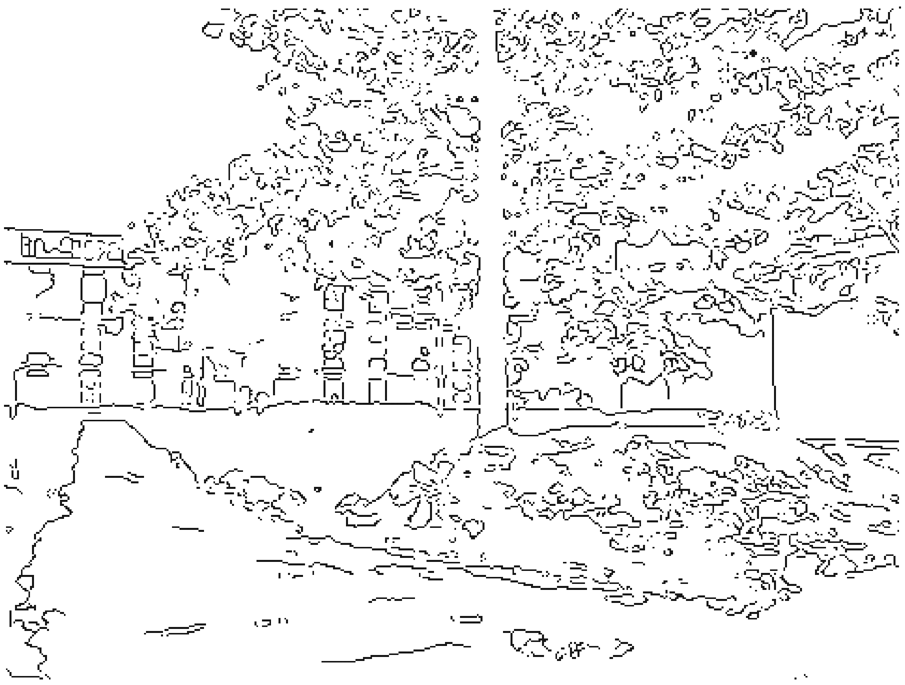


Figure B.5: The resulting edge map of figure B.1 after masking

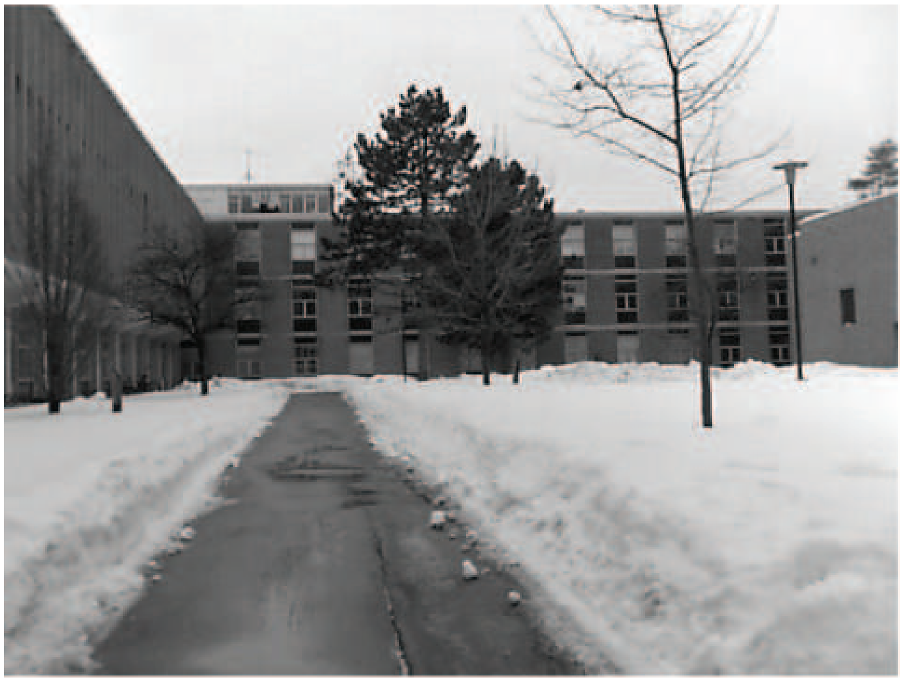


Figure B.6: An image captured by the camera for localization

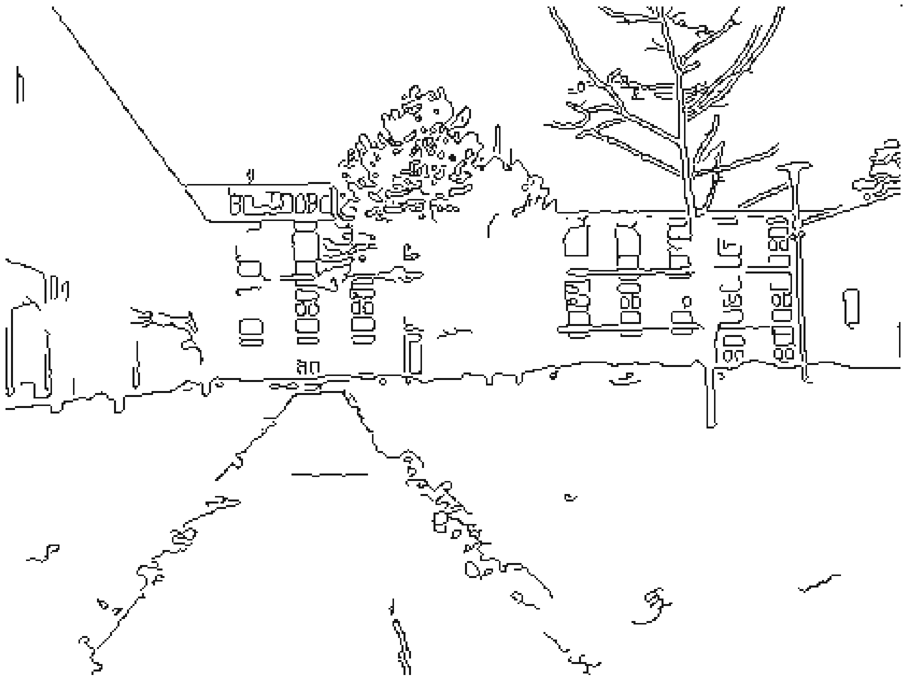


Figure B.7: The edge map of figure B.6

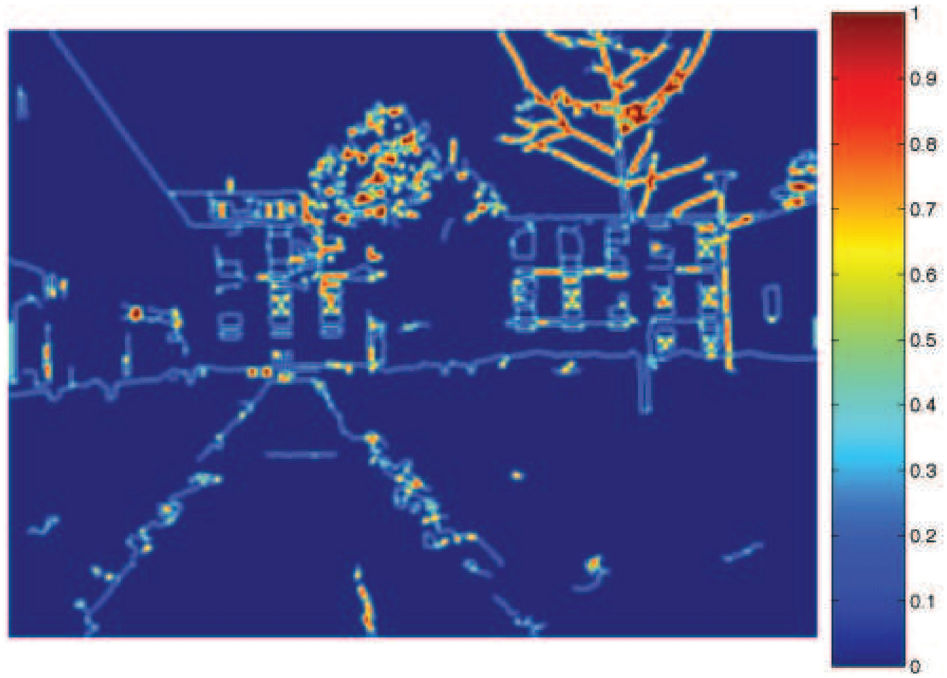


Figure B.8: The edge map density measure for figure B.6

Figure B.9: The edge response mask for figure B.6

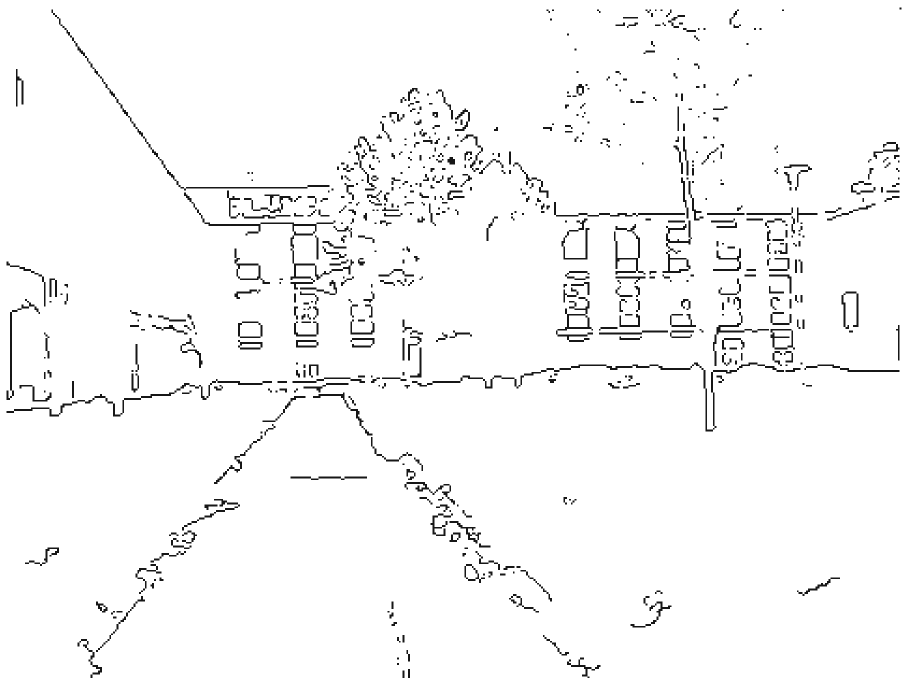


Figure B.10: The resulting edge map of figure B.6 after masking

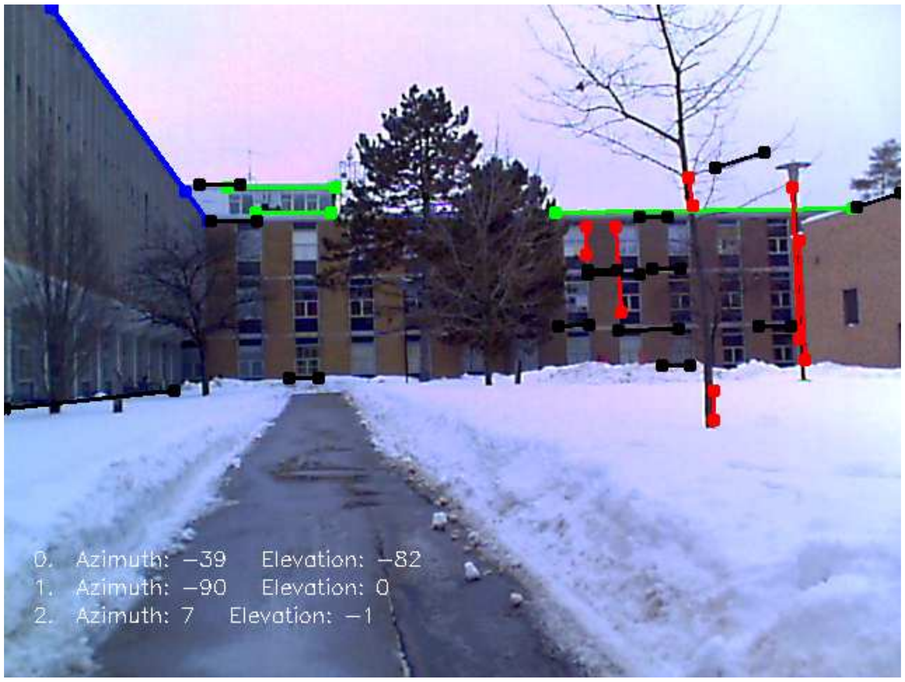

0. Azimuth: −39 Elevation: −82
1. Azimuth: −90 Elevation: 0
2. Azimuth: 7 Elevation: −1

Figure B.11: The vanishing point analysis results for figure B.6

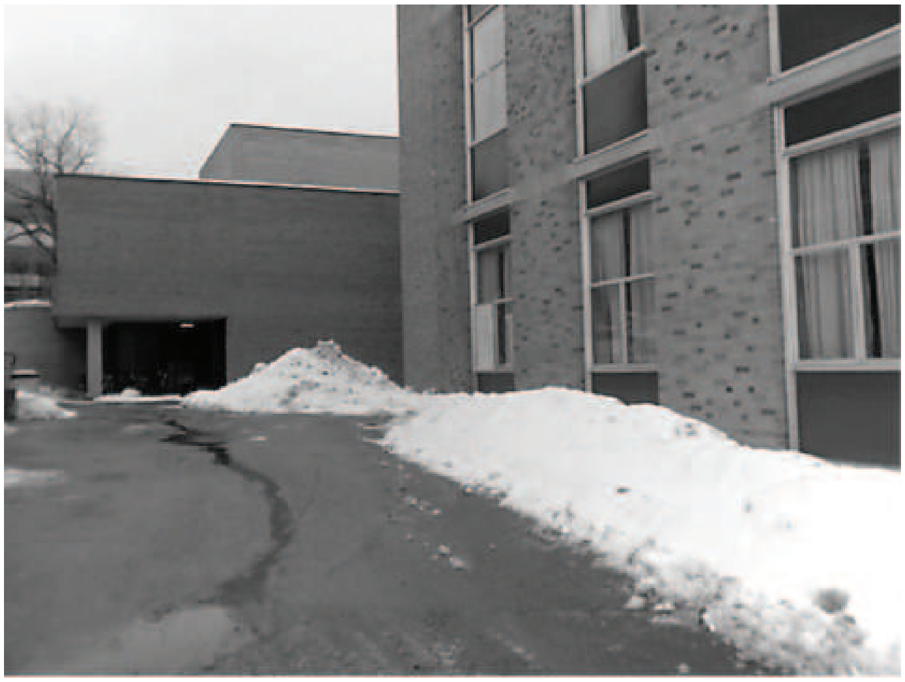


Figure B.12: An image captured by the camera for localization

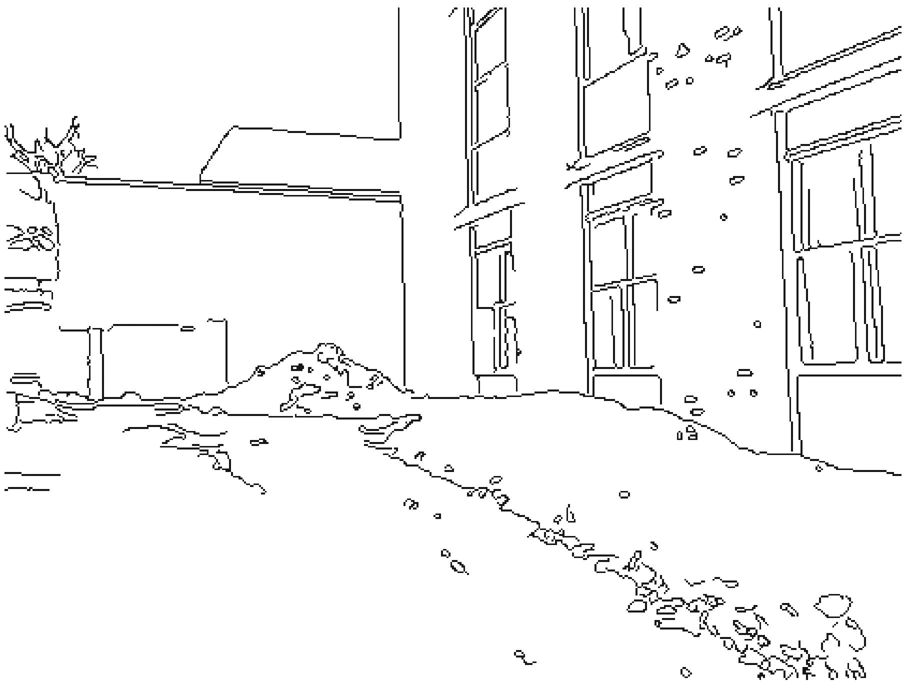


Figure B.13: The edge map of figure B.12

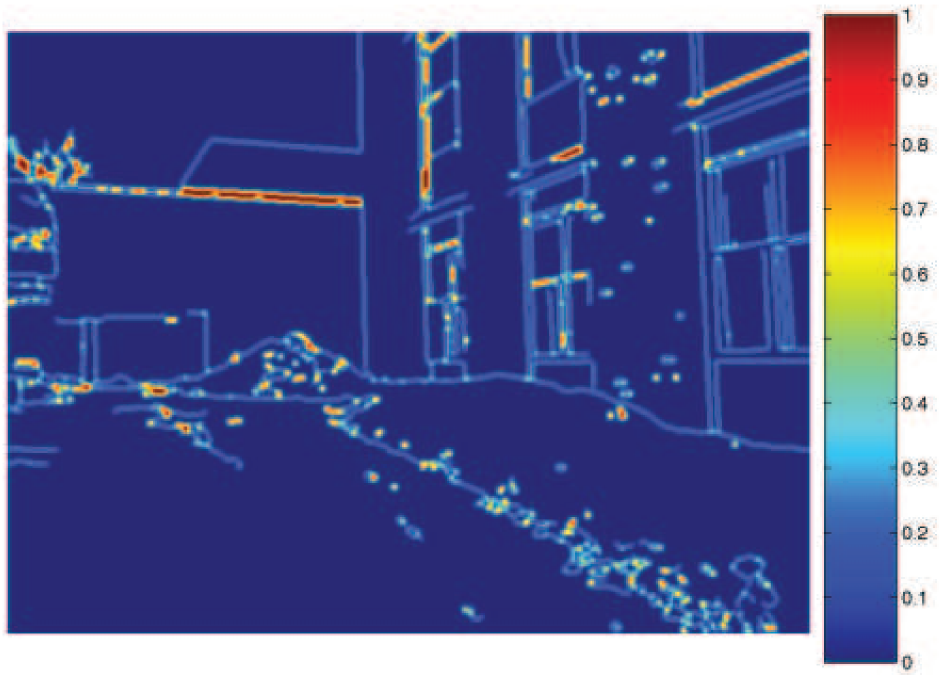


Figure B.14: The edge map density measure for figure B.12

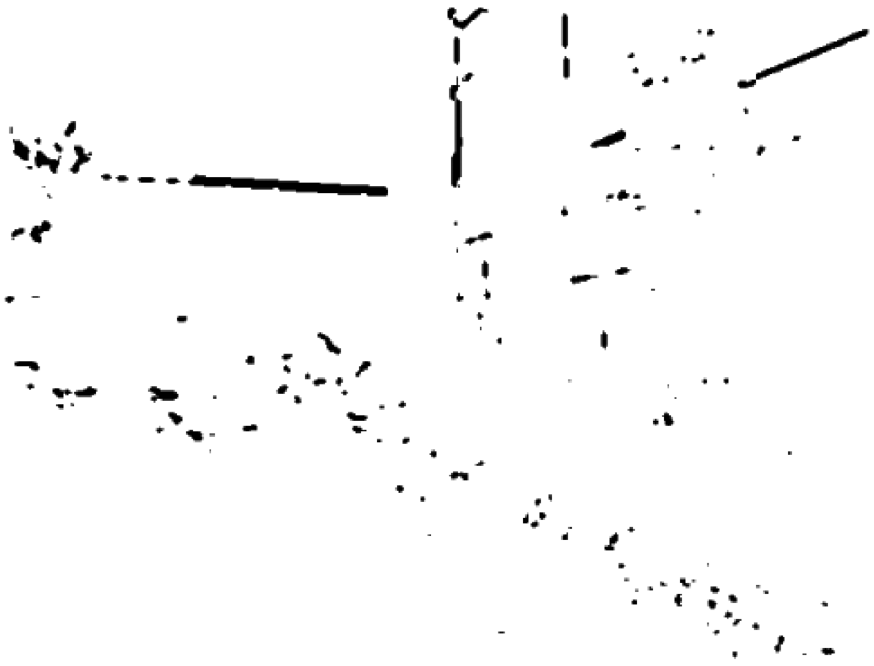


Figure B.15: The edge response mask for figure B.12

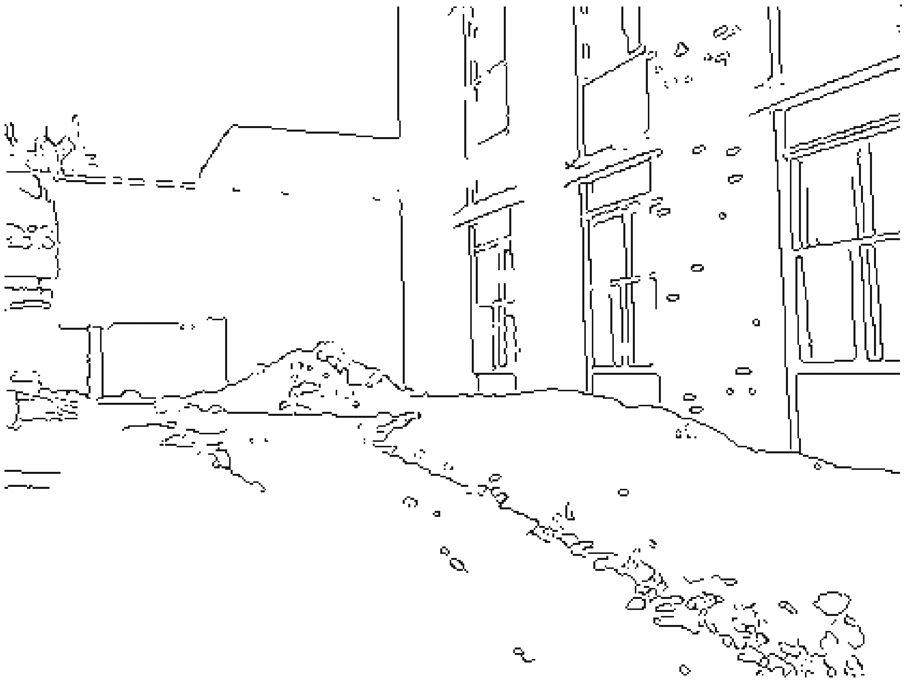


Figure B.16: The resulting edge map of figure B.12 after masking

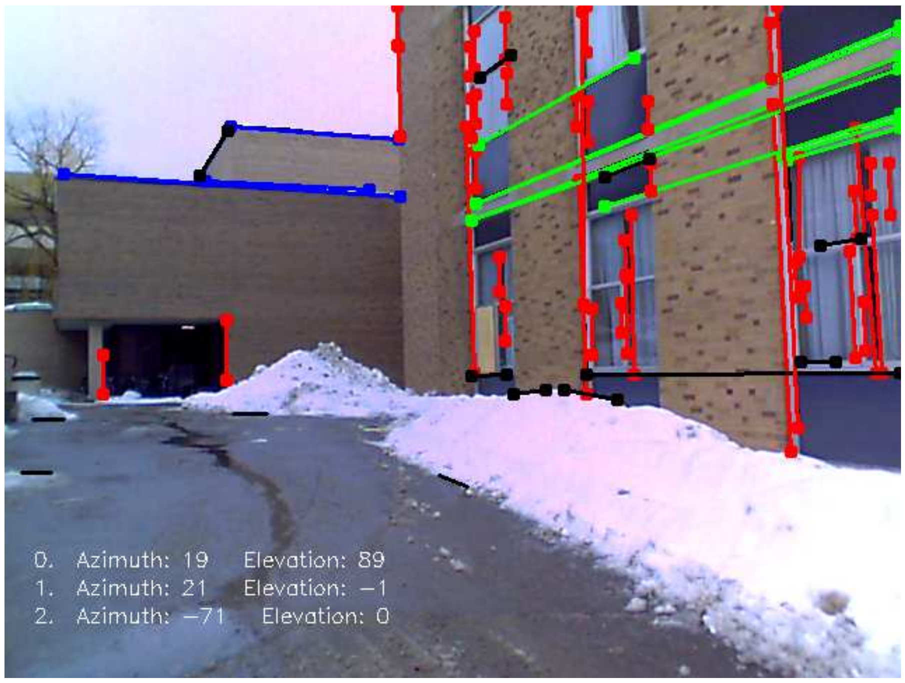

0. Azimuth: 19 Elevation: 89
1. Azimuth: 21 Elevation: −1
2. Azimuth: −71 Elevation: 0

Figure B.17: The vanishing point analysis results for figure B.12

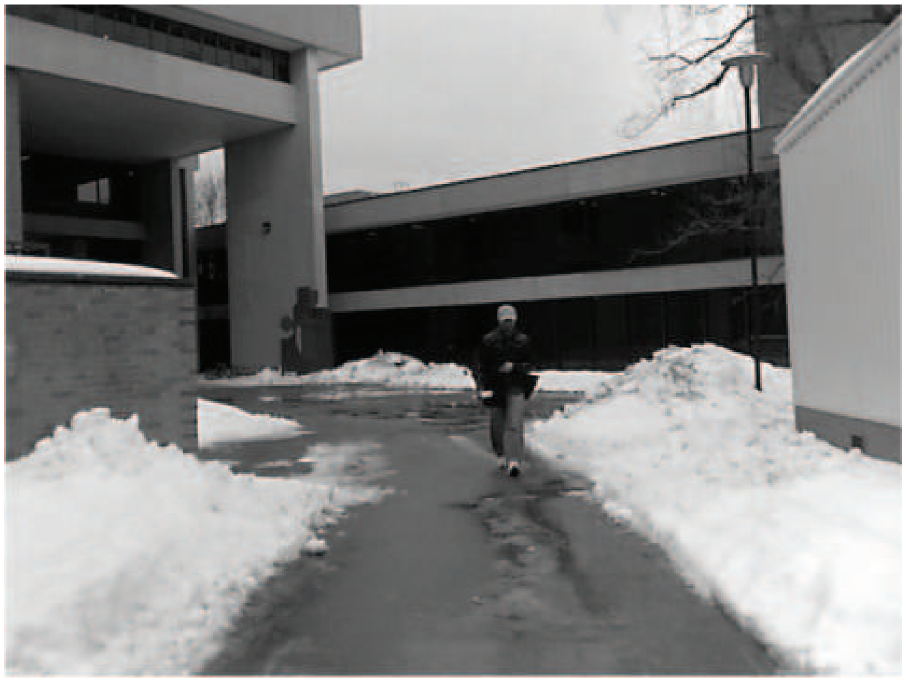


Figure B.18: An image captured by the camera for localization

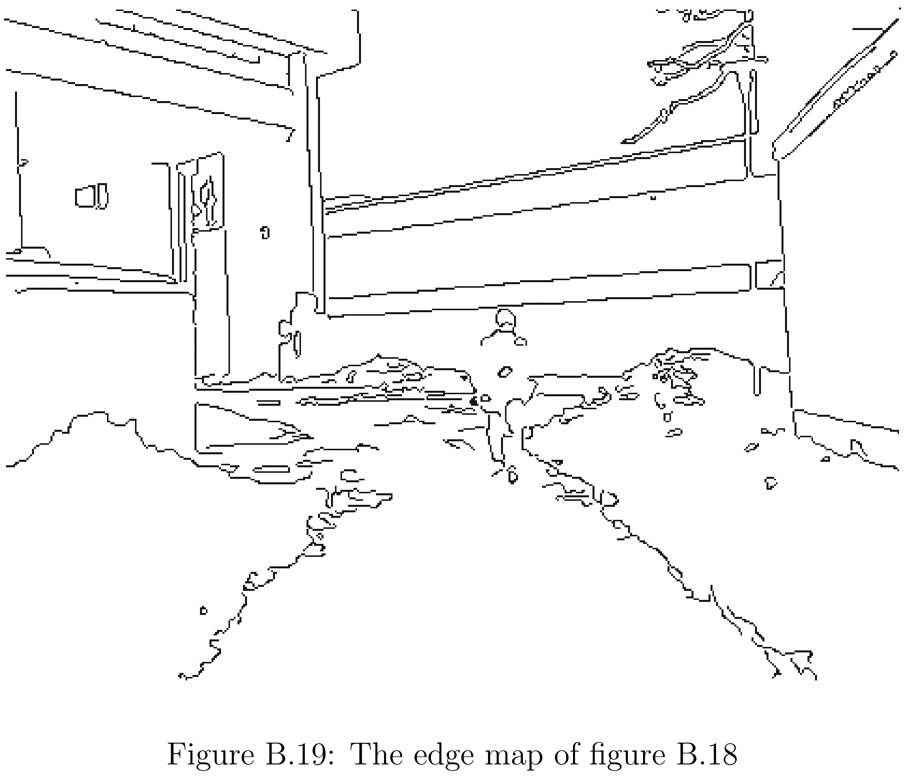


Figure B.19: The edge map of figure B.18

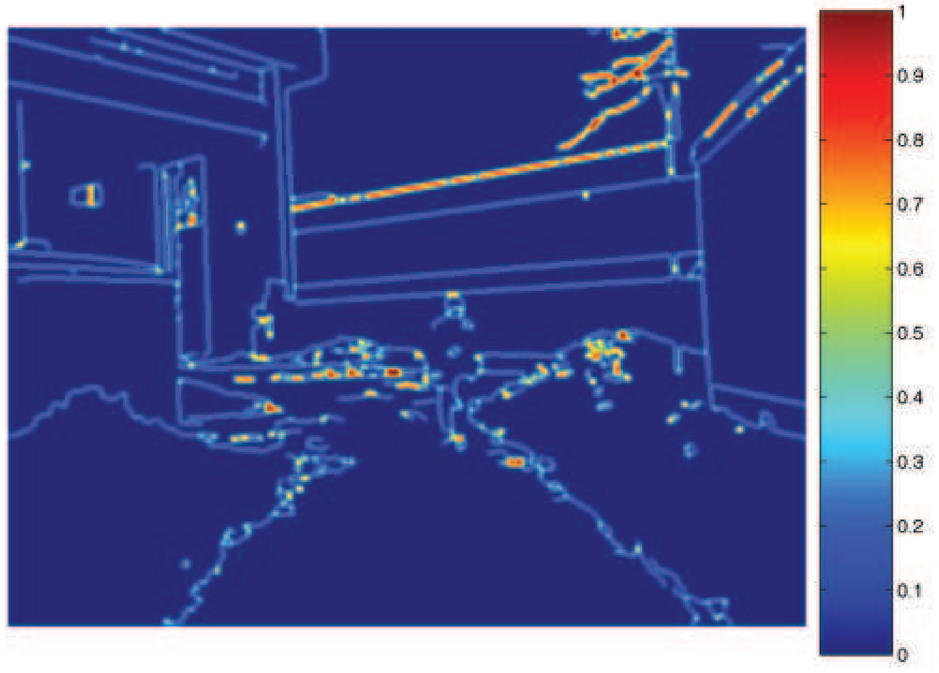


Figure B.20: The edge map density measure for figure B.18

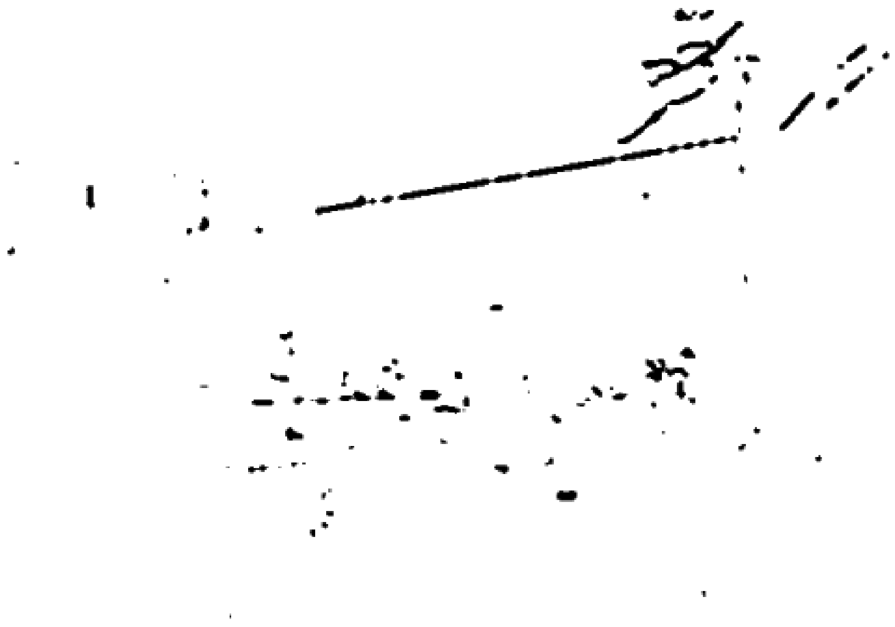


Figure B.21: The edge response mask for figure B.18

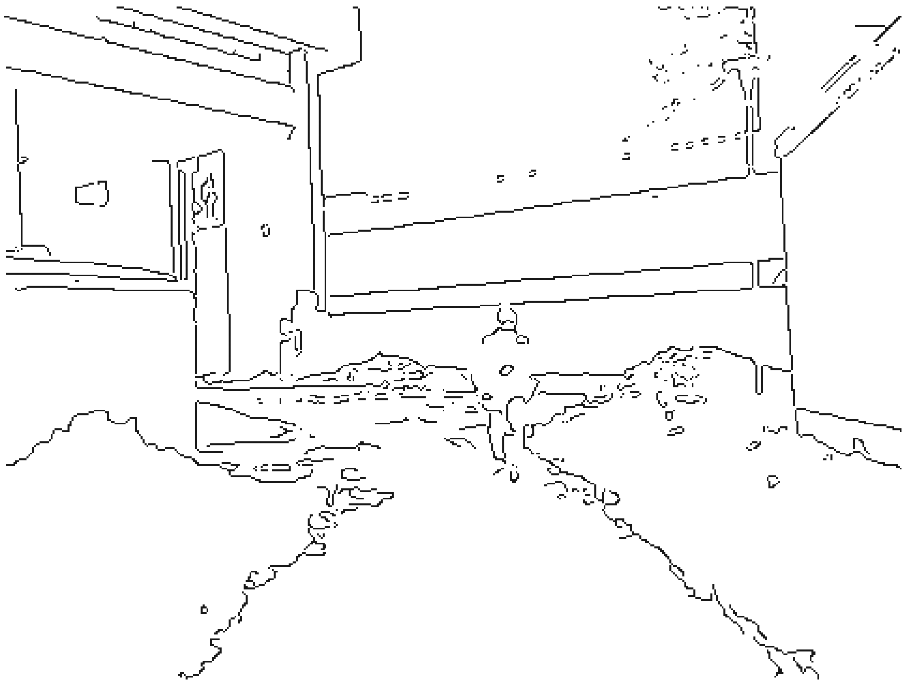


Figure B.22: The resulting edge map of figure B.18 after masking

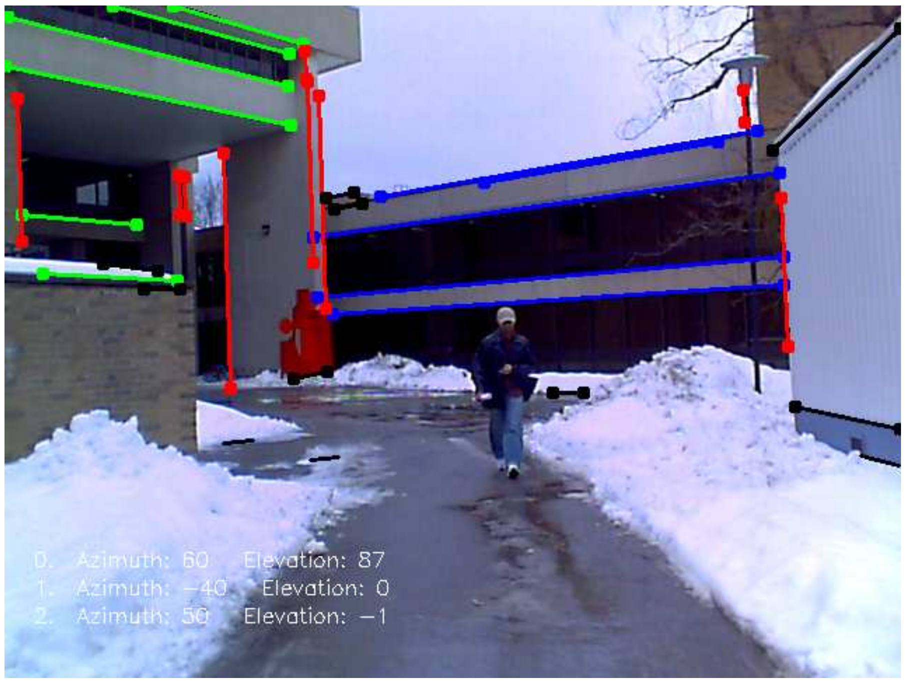

0. Azimuth: 60 Elevation: 87
1. Azimuth: −40 Elevation: 0
2. Azimuth: 50 Elevation: −1

Figure B.23: The vanishing point analysis results for figure B.18

Appendix C

Additional Localization Results

The figures presented in this appendix section are additional results of the particle filter localization algorithm. State estimates for two test runs are shown, where 2000 particles are used in both cases. The control input has a standard deviation of $10\frac{cm}{s}$ for forward velocity and 5 degrees for yaw. Smaller circles in each figure show the location of the particles. A larger circle show the true state, which is also uniquely labeled. The line protruding from each particle indicates its heading.

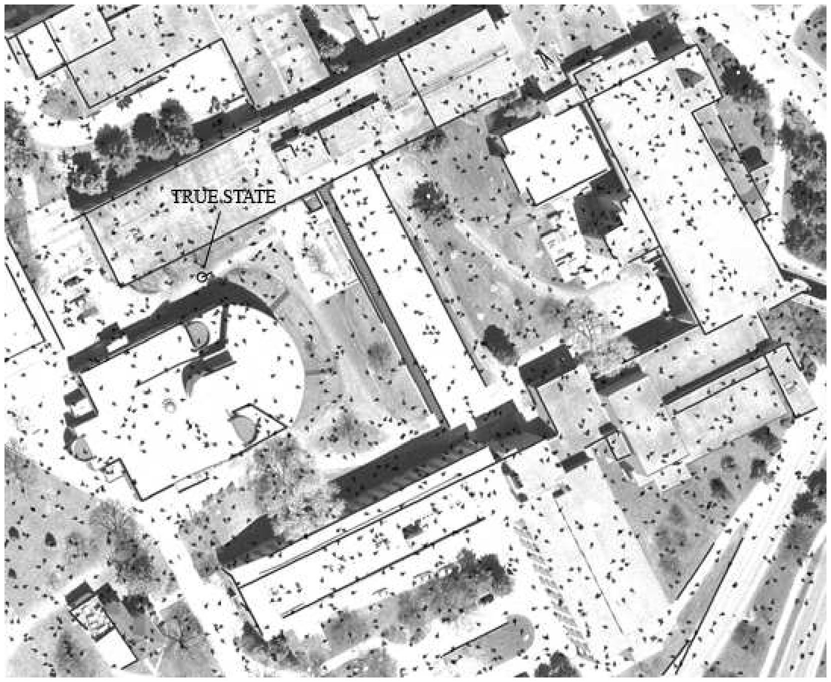


Figure C.1: particle filter localization result, set 1 - sequence 1 of 12

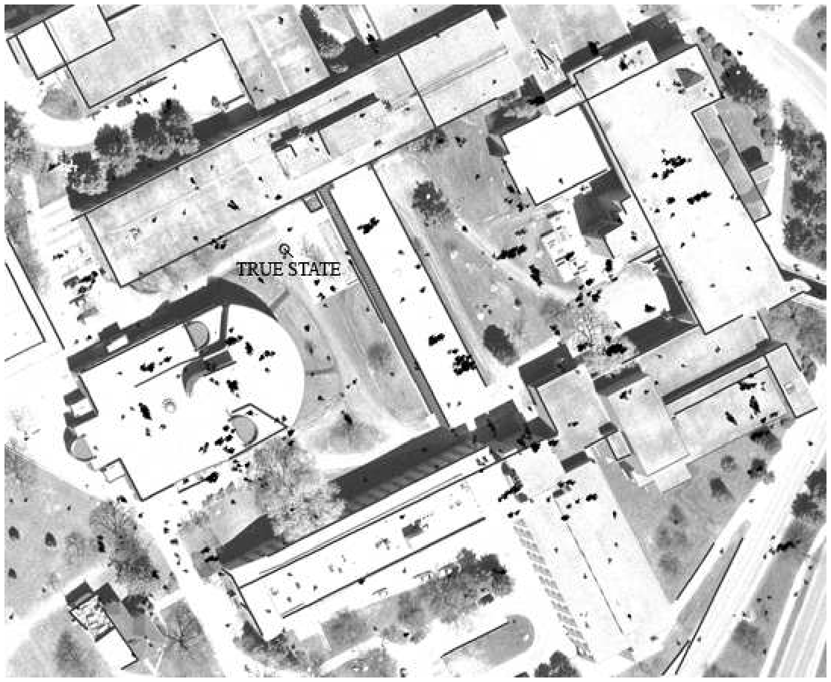


Figure C.2: particle filter localization result, set 1 - sequence 2 of 12

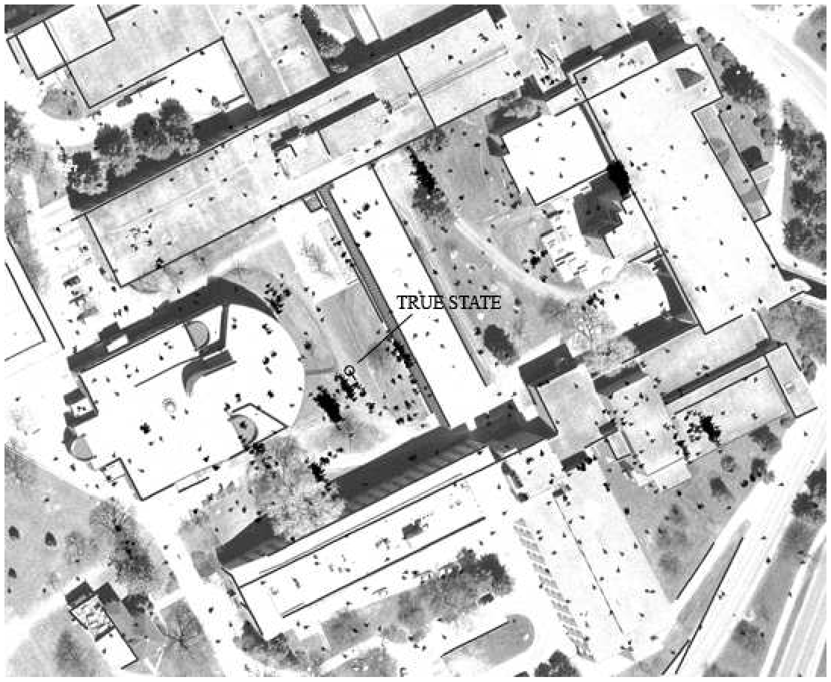


Figure C.3: particle filter localization result, set 1 - sequence 3 of 12

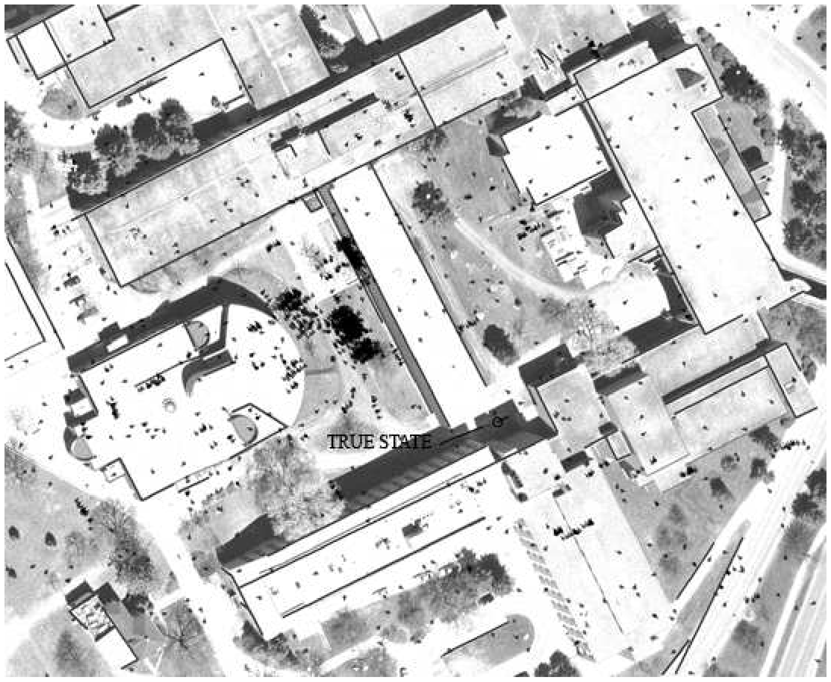


Figure C.4: particle filter localization result, set 1 - sequence 4 of 12

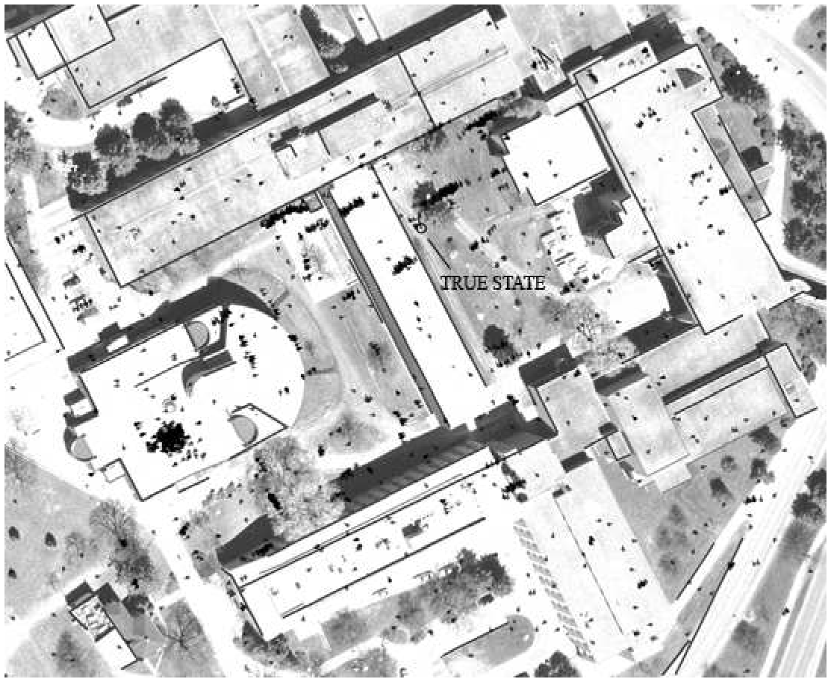


Figure C.5: particle filter localization result, set 1 - sequence 5 of 12

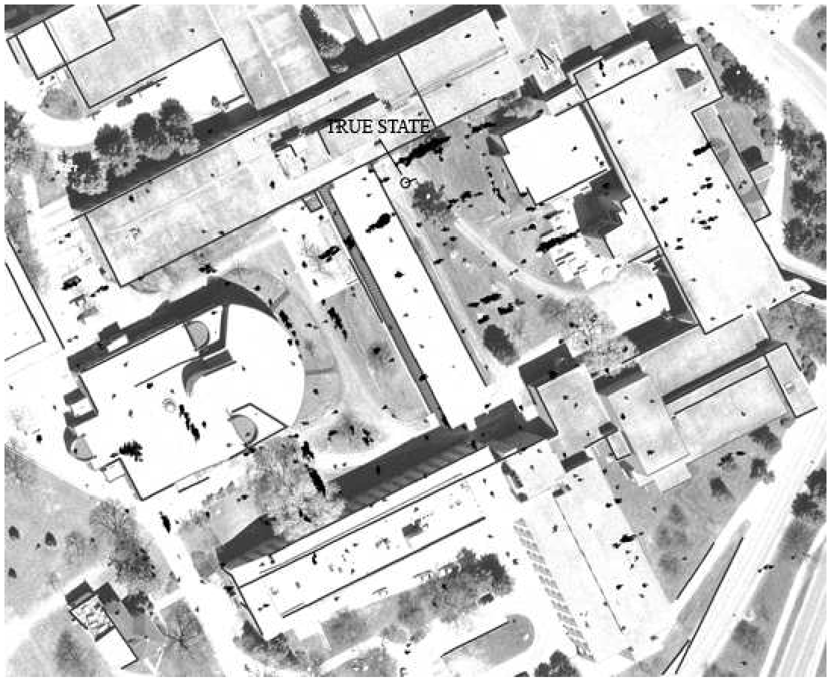


Figure C.6: particle filter localization result, set 1 - sequence 6 of 12

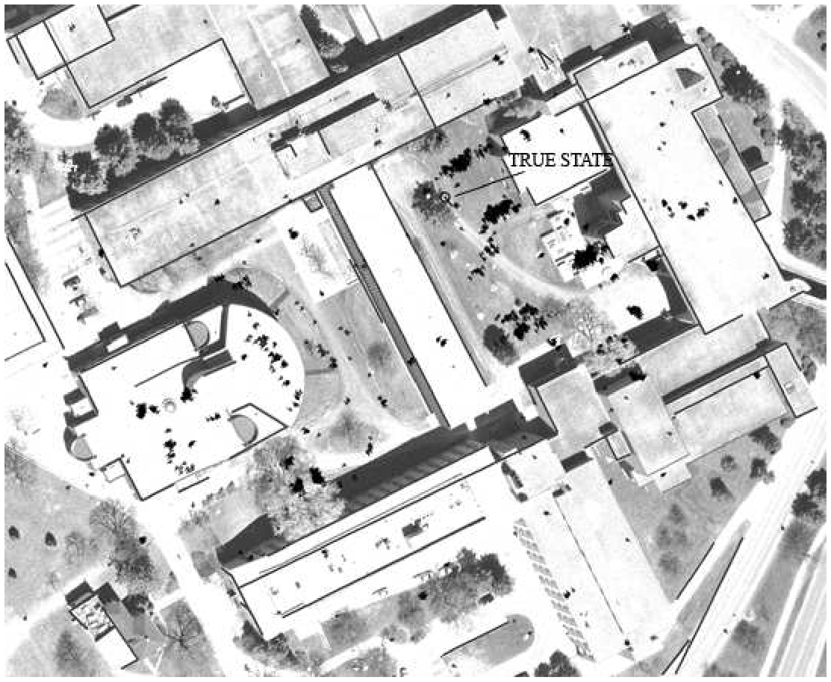


Figure C.7: particle filter localization result, set 1 - sequence 7 of 12

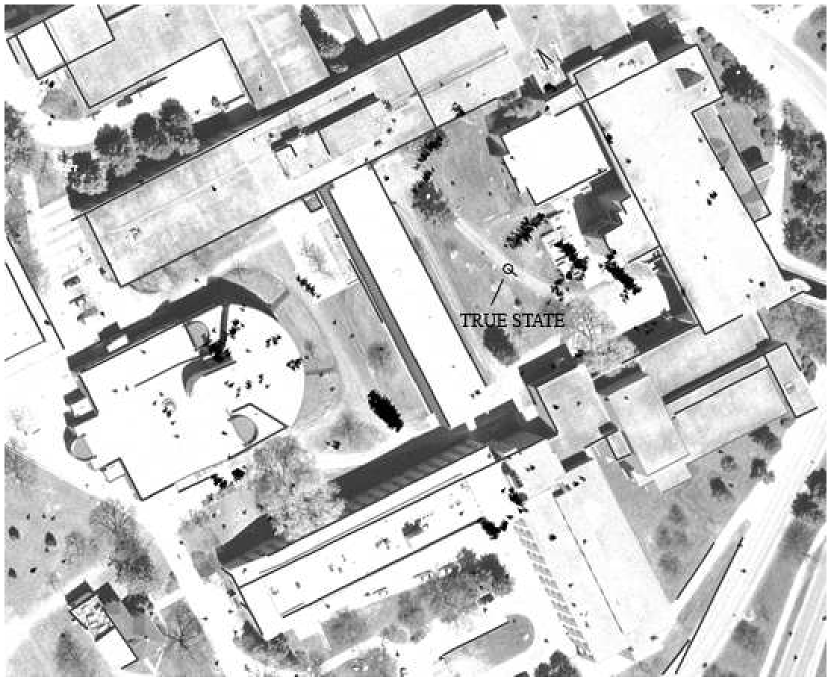


Figure C.8: particle filter localization result, set 1 - sequence 8 of 12

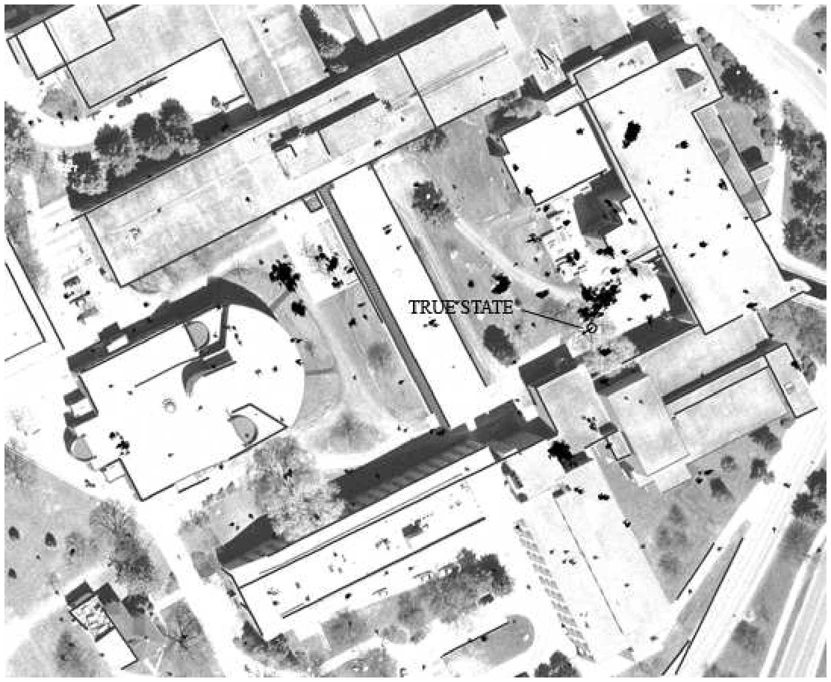


Figure C.9: particle filter localization result, set 1 - sequence 9 of 12

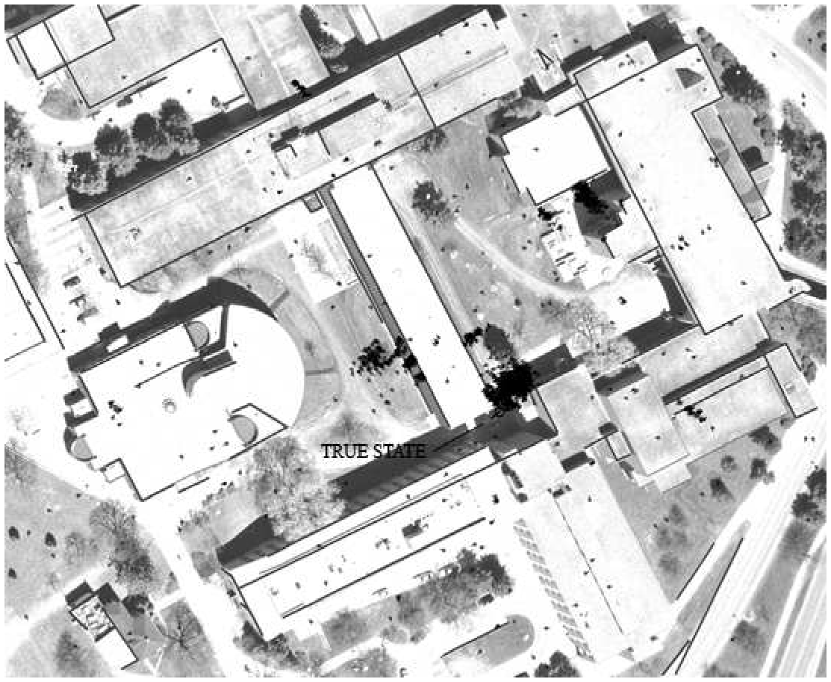


Figure C.10: particle filter localization result, set 1 - sequence 10 of 12

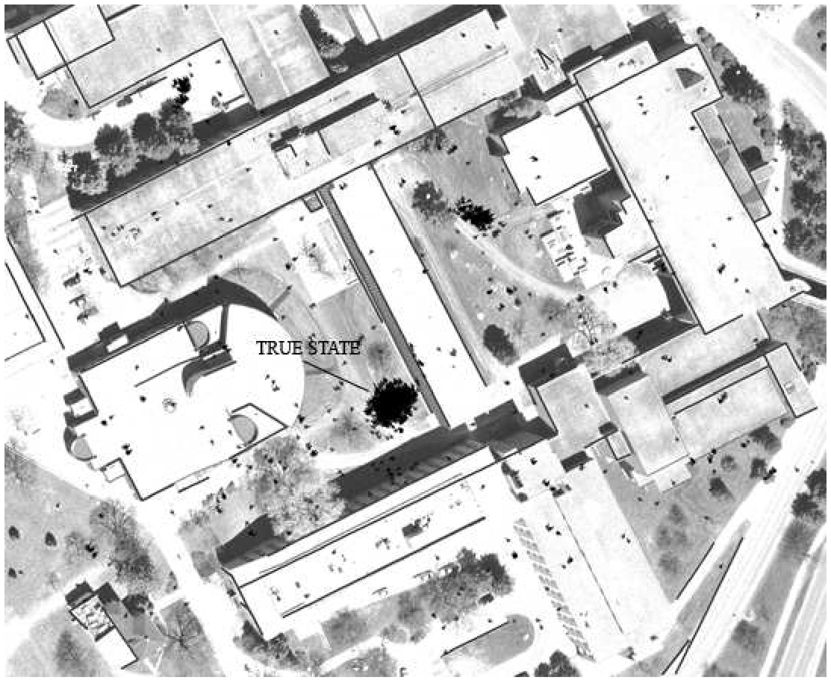


Figure C.11: particle filter localization result, set 1 - sequence 11 of 12

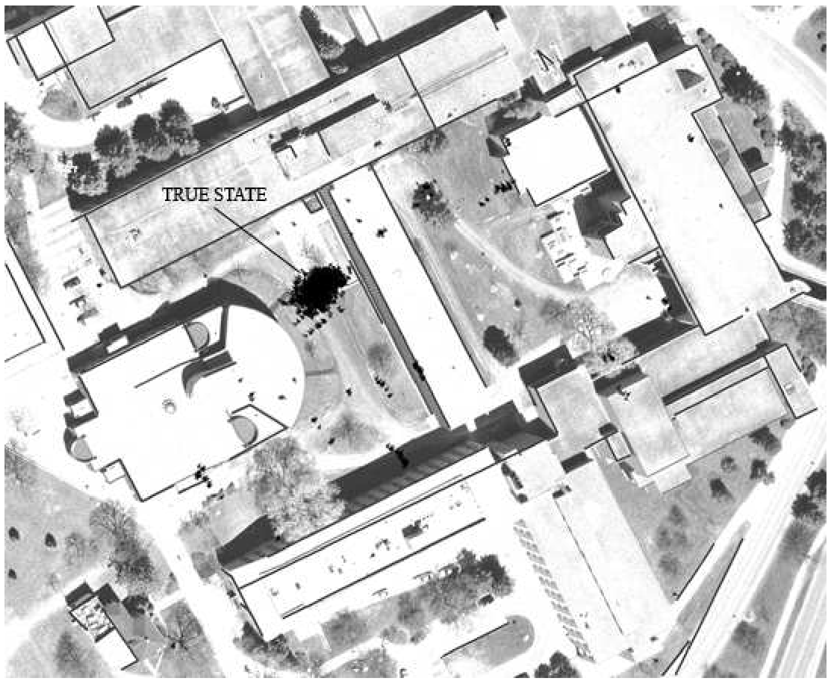


Figure C.12: particle filter localization result, set 1 - sequence 12 of 12

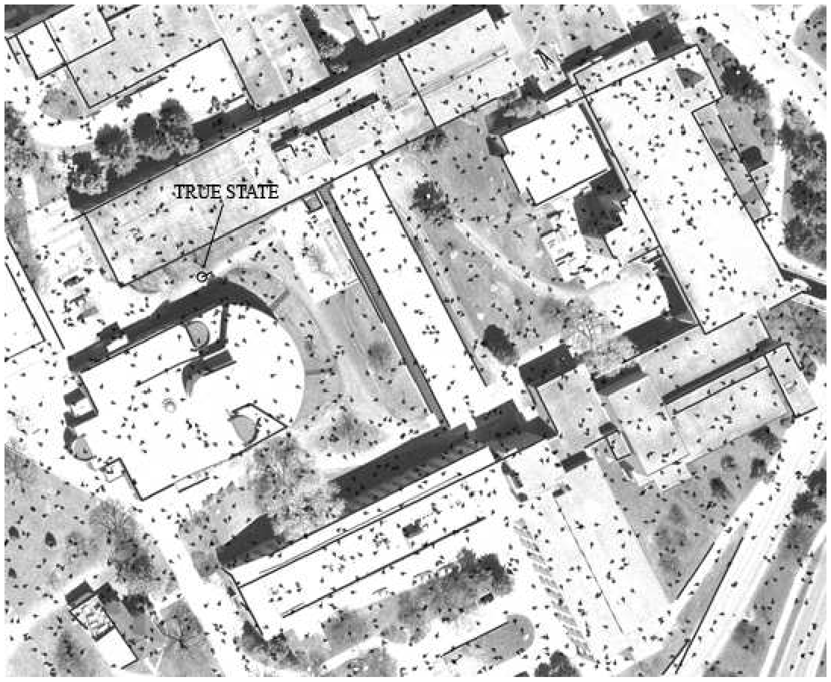


Figure C.13: particle filter localization result, set 2 - sequence 1 of 12

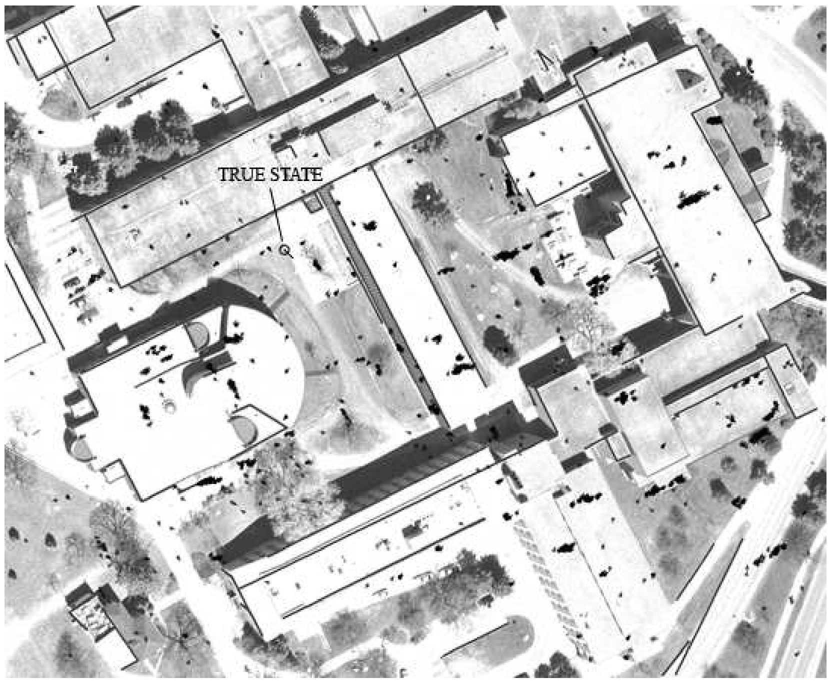


Figure C.14: particle filter localization result, set 2 - sequence 2 of 12

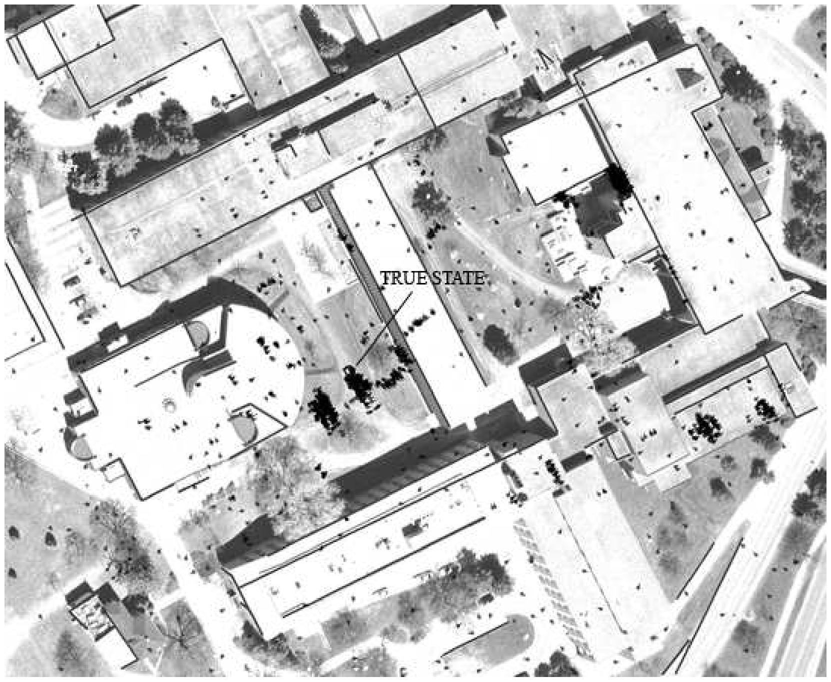


Figure C.15: particle filter localization result, set 2 - sequence 3 of 12

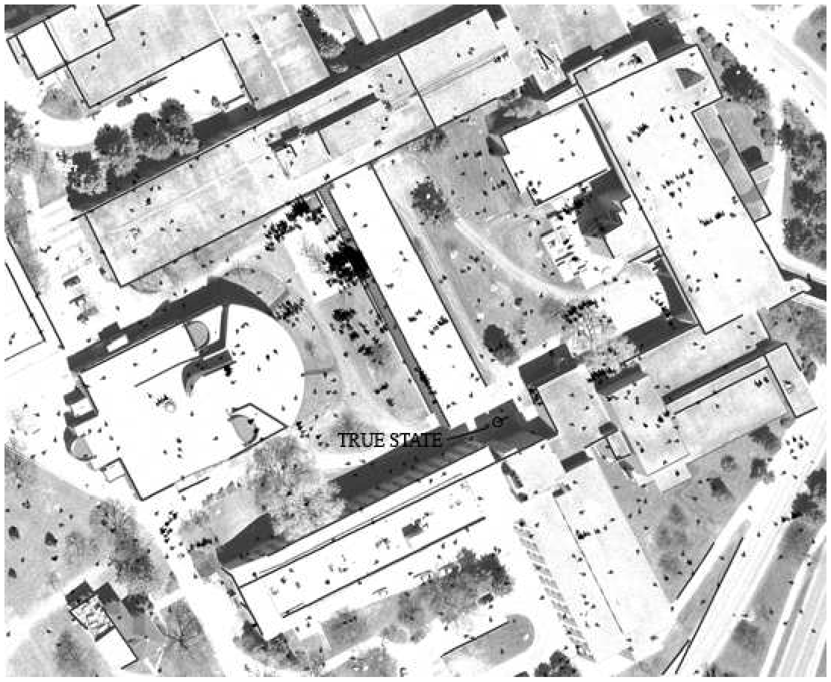


Figure C.16: particle filter localization result, set 2 - sequence 4 of 12

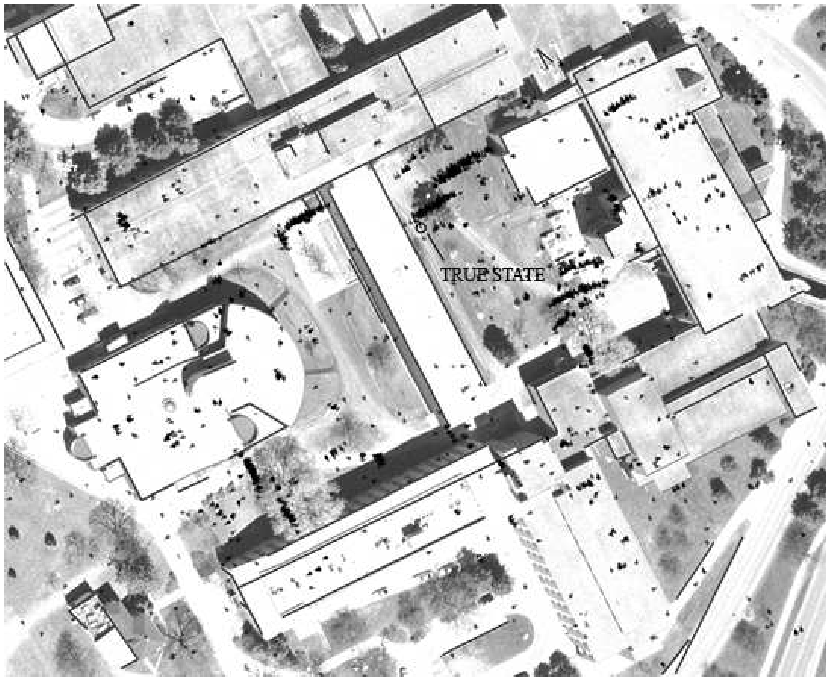


Figure C.17: particle filter localization result, set 2 - sequence 5 of 12

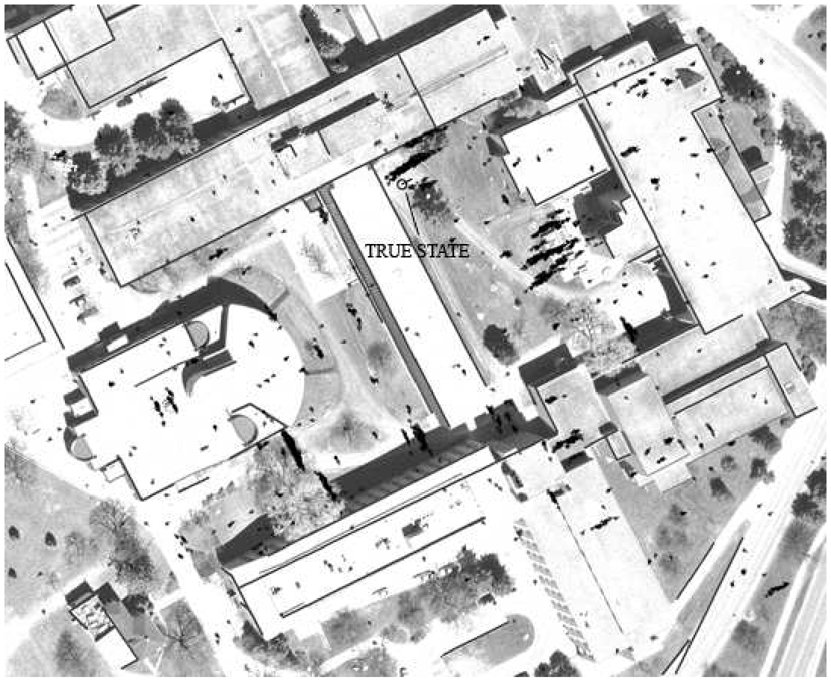


Figure C.18: particle filter localization result, set 2 - sequence 6 of 12

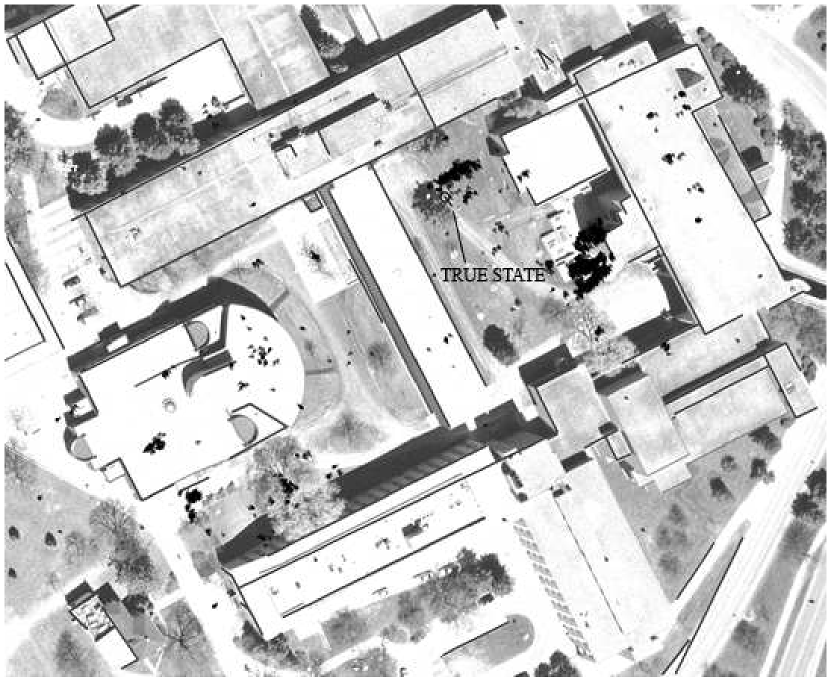


Figure C.19: particle filter localization result, set 2 - sequence 7 of 12

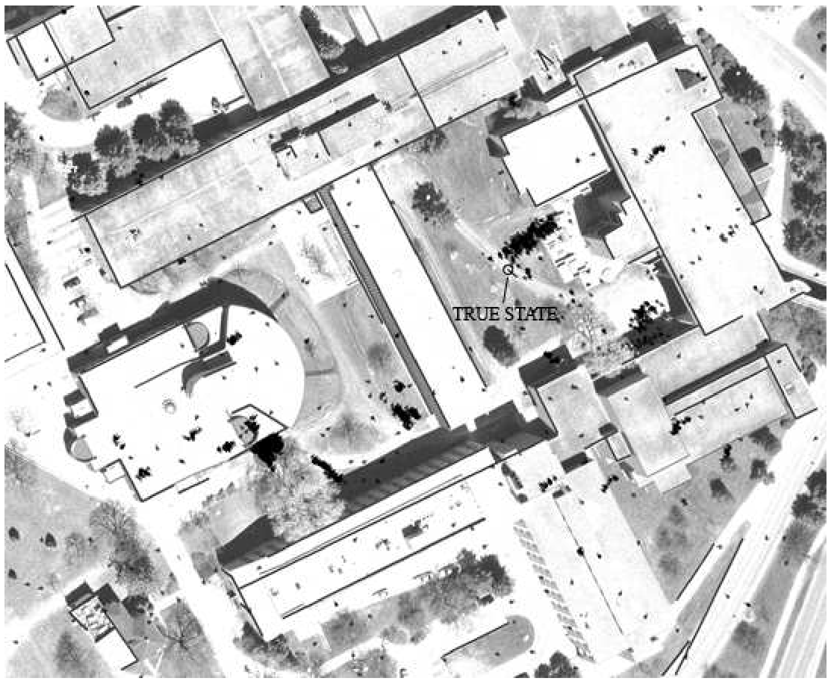


Figure C.20: particle filter localization result, set 2 - sequence 8 of 12

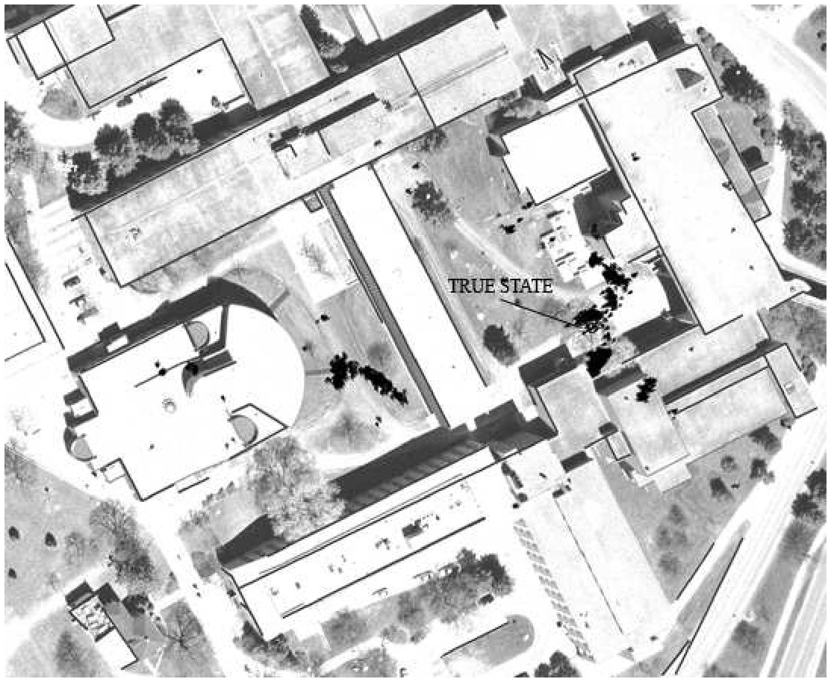


Figure C.21: particle filter localization result, set 2 - sequence 9 of 12

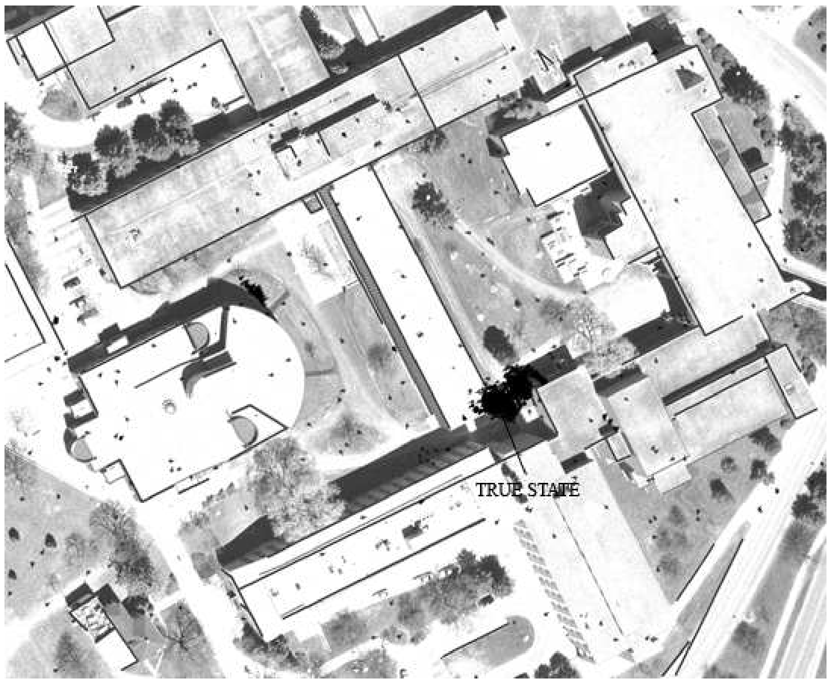


Figure C.22: particle filter localization result, set 2 - sequence 10 of 12

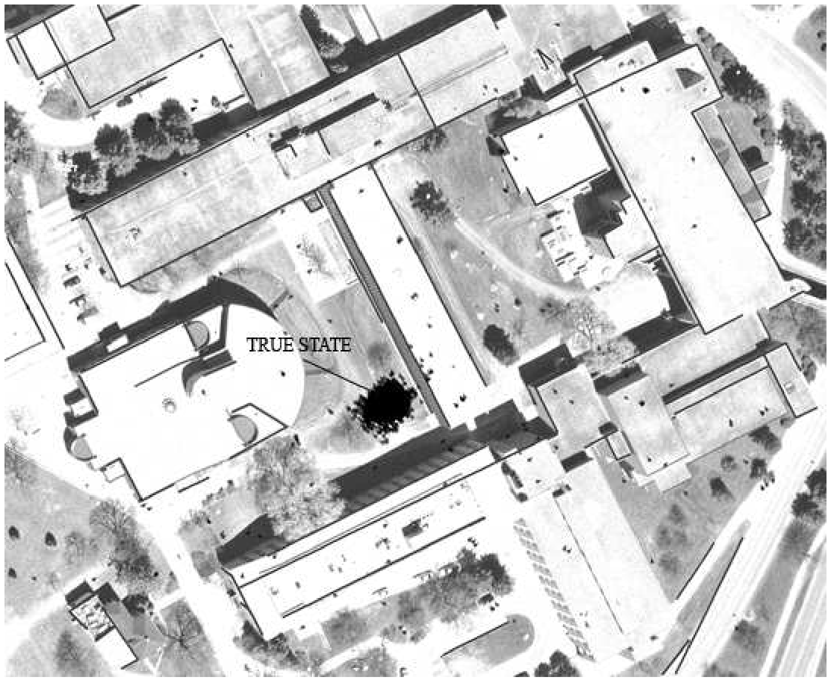


Figure C.23: particle filter localization result, set 2 - sequence 11 of 12

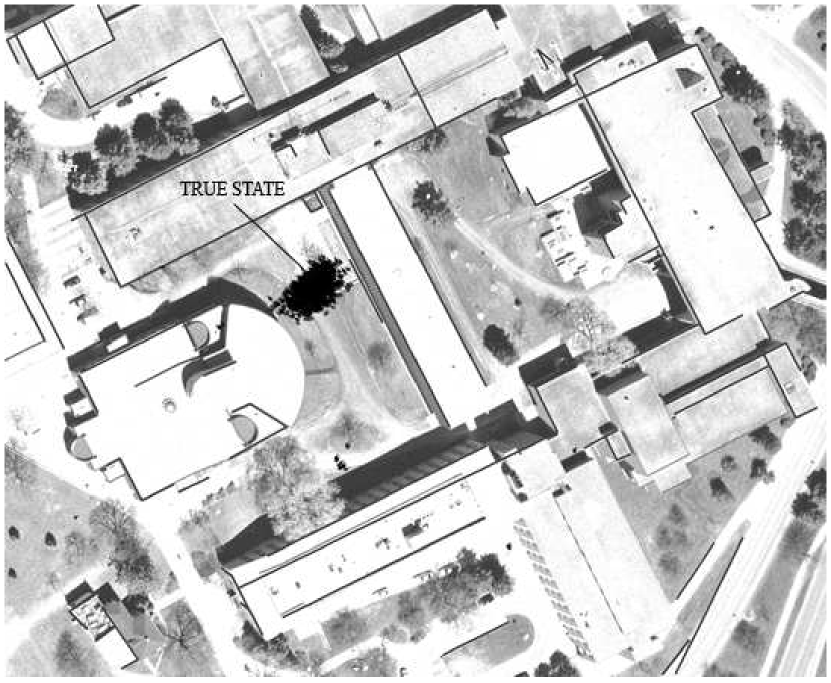


Figure C.24: particle filter localization result, set 2 - sequence 12 of 12

Bibliography

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotcs.* USA: The MIT Press, 2005.

[2] P. Debenest, E. Fukushima, and S. Hirose, “Proposal for automation of humanitarian demining with buggy robots,” in *Proceedings of IEEE Intelligent Robots and Systems*, 2003.

[3] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, “The interactive museum tour-guide robot,” in *Proceedings of AAAI Artificial intelligence/Innovative applications of artificial intelligence*, 1998.

[4] I. Ulrich, F. Mondada, and J. Nicoud, “Autonomous vacuum cleaner,” *Robotics and autonomous systems*, vol. 19, no. 3-4, pp. 233–245, 1997.

[5] T. Thurston and H. Hu, “Distributed agent architecture for port automation,” in *Proceedings of Computer Software and Applications Conference*, 2002.

[6] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier, “The cycab: a car-like robot navigating autonomously and safely among pedestrians,” *Robotics and Autonomous Systems*, vol. 50, no. 1, pp. 51–67, 2005.

[7] M. Whalley, M. Freed, M. Takahashi, D. Christian, A. Petterson-Hine, G. Schulein, and R. Harris, "The nasa / army autonomous rotorcraft project," in *59th Annual Forum of the American Helicopter Society*, 2003.

[8] J. Yuh, "Design and control of autonomous underwater robots: A survey," *Autonomous Robots*, vol. 8, no. 1, pp. 7–24, 2000.

[9] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots.* USA: The MIT Press, 2004.

[10] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion.* USA: The MIT Press, 2005.

[11] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *Internation Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, 2007.

[12] L. Matthies *et al.*, "A portable, autonomous, urban reconnaissance robot," *Robotics and Autonomous Systems*, vol. 40, no. 2-3, pp. 163–172, 2002.

[13] W. Zhang and J. Kosecka, "Image based localization in urban environemnts," in *Proceedings of the International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006.

[14] D. Johns and G. Dudek, "Urban position estimation from one dimensional visual cues," in *Proceedings of the Canadian Conference on Computer and Robot Vision*, 2006.

[15] M. Adams, "Lidar design, use, and calibration concepts for correct environmental detection," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 753–761, 2000.

[16] S. Thrun *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of Robotic Systems*, vol. 23, no. 9, pp. 661–692, 2006.

[17] U. Özgüner, K. Redmill, and A. Broggi, "Team terramax and the darpa grand challenge: a general overview," in *IEEE Intelligent Vehicles Symposium*, 2004.

[18] C. Q, U. Özgüner, and K. Redmill, "Ohio state university at the 2004 darpa grand challenge: developing a completely autonomous vehicle," *IEEE Intelligent Systems*, vol. 19, no. 5, pp. 8–11, 2004.

[19] SICK AG, 2007. [Online]. Available: http://www.sick.com

[20] E. Kaplan, *Understanding GPS: principles and applications.* USA: Atech House Inc., 1996.

[21] J. Tsui, *Fundamentals of Global Positioning System Receivers: A Software Approach.* Canada: John Wiley and Sons, Inc., 2000.

[22] G. Xu, *GPS Theory, Algorithms and Applications.* Germany: Springer Verlag, 2003.

[23] T. Acharya and A. K. Ray, *Image Processing Principles and Applications.* USA: John Wiley and Sons, Inc., 2005.

[24] K. Deguchi and T. Nakagawa, "Active and direct acquisition of 3d map in robot by combining motion and perceived images," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[25] Y. Kim and H. Kim, "Dense 3d map building for autonomous mobile robots," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[26] H. Chen and Z. Xu, "Local 3d map building and error analysis based on stereo vision," in *Proceedings of IEEE Industrial Electronics Society Conference*, 2005.

[27] S. LaValle, *Planning Algorithms.* Cambridge University Press, 2006.

[28] G. Welch and G. Bishop, "An introduction to the kalman filter," in *Special Interest Group for Computer Graphics Conference*, 2001.

[29] M. Arulampalam, S. Maskell, N. Gordon, and T.Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[30] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[31] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736–746, 2002.

[32] C. Lin and R. Nevatia, "Building detection and description from a single intensity image," *Computer Vision and Understanding*, vol. 72, no. 2, pp. 101–121, 1998.

[33] J. Shufelt and J. D.M. McKeown, "Fusion of monocular cues to detect man-made structures in aerial imagery," *CVGIP: Image Understanding*, vol. 57, no. 3, pp. 307–330, 1993.

[34] A. Croitoru and Y. Doytsher, "Right-angle rooftop polygon extraction in regularised urban areas: Cutting the corners," *The Photogrammetric Record*, vol. 19, no. 118, pp. 311–341, 2004.

[35] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[36] T.-Y. Liow, "Use of shadows for extracting buildings in aerial images," *Computer Vision, Graphics, and Image Processing*, vol. 49, pp. 242–277, 1990.

[37] I. Fogel and D. Sagi, "Gabor filters as texture discriminator," *Biological Cybernetics*, vol. 61, pp. 103–113, 1989.

[38] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," in *IEEE Systems, Man, and Cybernetics Conference*, 1990.

[39] T. P. W. amd W. E. Higgins and D. Dunn, "Efficient gabor filter design for texture segmentation," *Pattern Recognition*, vol. 29, no. 12, pp. 2005–2015, 1996.

[40] D. Dunn and W. E. Higgins, "Optimal gabor filters for texture segmentation," *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp. 947–964, 1995.

[41] M. Tuceryan and A. Jain, "Texture analysis," in *The Handbook of Pattern Recognition and Computer Vision 2nd Edition.* World Scientific Publishing, 1998, ch. 2, pp. 207–248.

[42] C. Harris and M. Stephens, "A combined edge and corner detector," in *4th Alvey Vision Conference*, 1998.

[43] S. Grossman, *Elementary Linear Algebra 5th Edition.* USA: Saunders College Publishing, 1994.

[44] W. K. Pratt, *Digital Image Processing.* USA: John Wiley and Sons, Inc., 1978.

[45] V. F. Leavers, *Shape Detection in Computer Vision Using the Hough Transform.* Germany: Springer-Verlag London Limited, 1992.

[46] J. Illingworth and J. Kittler, "A survey of the hough transform," *CVGIP*, vol. 44, pp. 87–116, 1988.

[47] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Computer Vision and Image Understanding*, vol. 78, pp. 119–137, 2000.

[48] C. Galambos, J. Matas, and J. Kittler, "Progressive probabilistic hough transform for line detection," in *IEEE Computer Vision and Pattern Recognition Conference*, 1999.

[49] H. Kälviäinen, P. Hirvonen, L. Xut, and E. Oja, "Probabilistic and non-probabilistic hough tranform: overview and comparisons," *Image and Vision Computing*, vol. 13, no. 4, pp. 239–252, 1999.

[50] N. Kiryati, H. Kälviäinen, and S. Alaoutinen, "Randomized or probabilistic hough transform: unified performance evaluation," *Pattern Recognition Letters*, vol. 21, pp. 1157–1164, 2000.

[51] F. van den Heuvel, "Vanishing point detection for architectural photogrammetry," *International Archives of Photogrammetry and Remote Sensing*, vol. 32, no. 5, pp. 652–659, 1998.

[52] V. Vantoni, L. Lombardi, M. Porta, and N. Sicard, "Vanishing point detection: Representation analysis and new approaches," in *Proceedings of the Internation Conference on Image Analysis and Processing*, 2001.

[53] S. Barnard, "Interpreting perspective images," *Artificial Intelligence*, vol. 21, no. 4, pp. 435–462, 1983.

[54] J. Shufelt, "Performance evaluation and analysis of vanishing point detection techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 282–288, 1999.

[55] A. Tai, J. Kittler, M. Petrou, and T. Windeatt, "Vanishing point detection," *Image and Vision Computing*, vol. 11, no. 4, pp. 240–245, 1993.

[56] A. Gallagher, "A ground truth based vanishing point detection algorithm," *Pattern Recognition*, vol. 35, no. 7, pp. 1527–1543, 2002.

[57] E. Lutton, H. Maitre, and J. Lopez-Krahe, "Contribution to the determination of vanishing points using hough transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 4, pp. 430–438, 1994.

[58] C. Bräuer-Burchardt and K. Voss, "Robust vanishing point determination in noisy images," in *IEEE International Conference on Pattern Recognition.*

[59] R. Collins and R. Weiss, "Vanishing point calculation as a statistical inference on the unitsphere," in *Proceedings of the Internation Conference on Computer Vision*, 1990.

[60] M. Magee and J. Aggarwal, "Determining vanishing points from perspective images," *Computer Vision, Graphics, and Image Processing*, vol. 26, pp. 256–267, 1984.

[61] C. Rother, "A new approach to vanishing point detection in architectural environemnts," *Image and Vision Computing*, vol. 20, no. 9-10, pp. 647–655, 2002.

[62] K.-S. Seo, J.-H. Lee, and H.-M. Choi, "An efficient detection of vanishing points using inverted coordinates image space," *Pattern Recognition Letters*, vol. 27, no. 2, pp. 102–108, 2005.

[63] B. O'Mahony, "New method for vanishing point detection," *CVGIP: Image Understanding*, vol. 54, no. 2, pp. 289–300, 1991.

[64] A. Almansa, "Vanishing point detection without any a priori information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 502–507, 2003.

[65] R. Schuster, "Steering a robot with vanishing points," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 4, pp. 491–498, 1993.

[66] A. Webb, *Statistical Pattern Recognition.* USA: Oxford University Press Inc., 1999.

[67] R. Yager and D. Filev, "Generation of fuzzy rules by mountain clustering," *Journal of Intelligent and Fuzzy Systems*, vol. 2, no. 3, pp. 209–219, 1994.

[68] S. Chiu, “Fuzzy model identification based on cluster estimation,” *Journal of Intelligent and Fuzzy Systems*, vol. 2, no. 3, pp. 267–278, 1994.

[69] G. Box and M.E.Muller, “A note on the generation of random normal deviates,” *The Annuals of Mathematical Statistics*, vol. 29, no. 2, pp. 610–611, 1958.

[70] B. Horn and B. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[71] J. Campbell, R. Sukthankar, and I. Nourbakhsh, “Techniques for evaluating optical flow for visual odometry in extreme terrain,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[72] F. Shafait, M. Grimm, and R.-R. Grigat, “Low-complexity camera ego-motion estimation algorithm for real time applications,” in *Proceedings of IEEE INMIC*, 2004.

[73] Z. Zhang, P. Cui, and H. Cui, “Recovery of egomotion from optical flow with large motion based on subspace method,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[74] A. Branca, G. Cicirelli, E. Stella, and D. A, “Mobile vehicle’s egomotion estimation from time varying image sequences,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.

[75] W. Burger and B. Bhanu, “Estimating 3-d egomotion from perspective image sequences,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 11, pp. 1040–1058, 1990.