Enhanced Visual Navigation for Mobile Robots

by

Brian West

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 4, 2014

Approved by

Thaddeus Roppel, Chair, Associate Professor of Electrical and Computer Engineering
Prathima Agrawal, Samuel Ginn Distinguished Professor of
Electrical and Computer Engineering
John Hung, Professor of Electrical and Computer Engineering

Abstract

Navigation solely by vision has been a goal in robotics for many years. It is desirable for a robot to plan and navigate the same way we do naturally as humans; however computation speed, changing illumination, complex algorithms, and a host of other problems have long plagued progress. One of the most prominent issues in using visible light outdoors and indoors is the unpredictable and changing ambient lighting. Shadows can vastly affect the robot's ability to navigate on a simple path, and often look like separate objects to a robot. Features detected and described under perfect light may not register under dim light. This work focuses on a method developed to address these issues and construct a navigable path using color information, despite ambient and direct lighting. The results show a vast improvement over traditional intensity-based algorithms both in navigation and localization.

Acknowledgment

For my wife, who has sacrificed much for me to pursue my dreams. I couldn't have done it without her love, support, and encouragement. Thank you Mary Grace.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

SIFT	Scale Invariant Feature Transform
SSD	Sum of Squared Differences
SLAM	Simultaneous Localization and Mapping
RGB	Red-Green-Blue
SAD	Sum of Absolute Differences

Chapter 1: Introduction

Since the dawn of automation, man has dreamed of a way to create machines capable of perceiving and interacting with their surroundings. From the first creation of Zadoc P. Dederick's Steam Man [1], we have been seeking new ways to make our creations navigate, manipulate, and understand the world we live in.

Since then, there have been many devices constructed for mobile robots to sense their environments. Tactile sensors are used to feel surroundings by registering touch, force, or pressure, but must make physical contact [2]. Sonar sensors "ping" the environment searching for sound waves which bounce off of objects [3], but are inherently noisy. Laser and lidar systems use light in a similar manner to produce point clouds [4], but can be very costly, with the most reliable state of the art systems costing thousands of dollars. Infrared has been used in this manner as well, and also with a camera to view pattern distortion (such as in the Microsoft Kinect™) [5], but does not work well outside due to solar interference.

Using the visible light spectrum in the context of computer vision as a sole means of data acquisition for robots has been avoided for many years due to its various problems. Computation speed is just starting to catch up to the increasingly complex algorithms required, and lighting conditions can cause conventional methods to fail. However, while radar, sonar, tactile and other methods of navigation and localization have proven useful, it is desirable for a robot to plan and navigate the same way we do

naturally as humans. We have naturally built our environments to support the transfer of information through sight, and enabling robots to see the world as we do will simplify programming, human-robot interaction, and especially navigation.

There is also a considerable cost advantage using a simple camera over many costly sensors in tandem. Most vision-based systems actually work best with relatively low resolution imaging, reducing complexity of algorithms and drastically reducing processing time. In fact, simple inexpensive webcams have proven substantial in most applications. Thus expanding vision-based technology will vastly improve accessibility in all levels of education as well.

One of the most prominent issues in using visible light as the sole means of data acquisition outdoors is the unpredictable and changing ambient lighting. Shadows can vastly affect the robot's ability to navigate on a simple path, much less paths with complex coloring and texture. Shadowing can also present challenges indoors, as shadows often look like separate objects to a robot. To confront this issue, it is necessary to construct a color-invariant image of the robot's surroundings. This simply means that despite changes in ambient and direct lighting, the color information should remain largely the same.

During the course of this research, a method for obtaining a color-invariant image was developed and tested in tandem with computer vision techniques for mobile robot navigation, and will be presented throughout the remainder of this thesis.

Chapter 2: Literature Survey

This chapter serves to provide essential background information regarding the field of robotics and computer vision. First, a brief history of computer vision and its use in robotics is presented. Second, key methods for providing and interpreting information for navigation and motion estimation are reviewed. Next, the topic of color-invariance is explored both in history and current research, and finally, current robust methods of localization, pose estimation, and map building, and path planning with obstacle avoidance pertaining to mobile robots will be discussed.

2.1 Computer Vision

In the most general sense, computer vision is the practice of describing the environment, and providing information of its properties including shape, illumination, objects, and colors. Though some researchers include non-visible light (such as infrared or lidar) as a subset of the field of computer vision, the usage within this thesis refers strictly to information gathered in the visible light spectrum through means of a camera.

2.1.1 A Brief History

Computer vision first found its mark in the 1970s, as the agenda for artificial intelligence was becoming more aggressive. It was originally thought that pairing the visual input with “intelligent” computers and systems would be an easy step towards a

functioning human-like perception and decision making system. According to Boden, a professor at Massachusetts Institute of Technology once asked a student to "spend the summer linking a camera to a computer and getting the computer to describe what it saw" [6]. Computer vision quickly distinguished itself from the field of image processing, as full scene understanding and three dimensional reconstructions were pursued.

By the 1980s, techniques for stereo correspondence had been established, and motion estimation started taking place in the form of optical flow. Image pyramids (using different levels of resolution) were being used in coarse-to-fine searches and scale-space property inspection, and many algorithms were unified using the same basic mathematical infrastructure [7].

In the 1990s, work was devoted to topics such as structure from motion, facial recognition, image segmentation, and invariant descriptors. Physics-based vision was born out of using color and intensity measurements along with physical models, and optical flow and stereo vision methods were vastly improved. The computer animation business boomed, while animators could reconstruct models from images [7].

Computer vision is becoming increasingly prevalent in today's society. Character recognition is used to read postal codes and license plates [8], industrial parts are quickly assessed for quality control, object recognition is used in self-checkout lines [9], and three dimensional models are created from a sequence of images. Computer vision has also found its way into the automotive world, detecting objects and obstacles, and researchers have even built self-driving cars [10]. It is also used for surveillance in diverse applications such as building security, traffic analysis, and locating drowning

victims in pools. The most recent research comes from feature-based object recognition and navigation partnered with machine learning and Bayesian techniques [7].

2.1.2 Feature Detection and Tracking

Methods of feature detection and tracking throughout this thesis have been restricted to the Scale Invariant Feature Transform (SIFT) algorithm. SIFT was first proposed by Lowe in [11], and has become a staple in the computer vision community and a baseline to measure new keypoint detectors and descriptors.

There have been many spin-offs of the SIFT algorithm, such as Principle Component Analysis (PCA)-SIFT [12], Gradient Location-Orientation Histogram (GLOH) [13], and Speeded Up Robust Features (SURF) [14], all of which improve upon computation and work marginally better.

More recent methods such as BRISK [15] involve different sampling methods for the descriptors and sub-pixel level accuracy, accomplished by quickly searching at lower resolutions and interpolating for fast sampling and feature recognition.

For ease of use and prototyping reasons, all feature detection and tracking in this thesis is performed using a SIFT type algorithm, as its speed and robustness are more than adequate for the purpose of this research.

2.1.3 Optical Flow

Optical (or optic) flow is a method of motion estimation by calculating the motion of color or intensity differences on each pixel of an image, first appearing in the 1970s [16]. The most common, and general way of doing this is by minimizing the sum of

squared differences (SSD) between two subsequent images. This is routinely done by summing over the overlapping regions of the image and looking for minima, or by using Markov random fields and looking for a global minimum.

Both methods can be refined in incremental steps to achieve sub-pixel estimation, and both methods have even been used together in instances when the type of motion is known [17]. For example, if a robot is navigating using a camera, the assumption can be made that most motion is from the movement of the robot within a static scene. Local optical flow calculations can be made, and the entire scene can be parameterized to form a global estimation of motion to update the robot's position.

Computationally, performing per-pixel search over an entire possible displacement is a slow process, and more efficient means of estimation have been developed more recently. These newer methods also utilize gradient descent and coarse-to-fine searches along with the classical optical flow techniques to attain motion estimation [17].

2.2 Color Invariance

Shadows can vastly affect the robot's ability to navigate on a simple path, even more so on paths with complex coloring and texture. Shadowing can also present challenges indoors as well as outdoors, as shadows often look like separate objects to a robot. To confront this issue, it is necessary to construct a color invariant image of the robot's surroundings. This simply means that despite changes in ambient and direct lighting, the color information remains largely the same. Xu et al. propose a method to remove shadows from an image using the L-2 normalization of the RGB color values

[18]. Another method, based on log-chromaticity space projection, is presented by Finlayson et al. [19]. This approach uses log-domain characteristics to create a color invariant image, projected into intensity values. Finlayson made the observation that similar colors in an image are projected onto lines having similar slopes in log-chromaticity space despite ambient and direct lighting. When projected in the direction orthogonal to the lighting, a color invariant image is obtained.

2.3 SLAM

Simultaneous Localization and Mapping (SLAM) has long been described as a "chicken and egg" problem. For a robot to accurately localize itself, it must have an unbiased map representation. However, to build an unbiased map, a robot must have an accurate pose estimate, accomplished by successfully localizing itself.

SLAM has the goal as described by Siegwart et al. of "starting from an arbitrary initial point, a mobile robot should be able to explore autonomously the environment with its on-board sensors, gain knowledge about it, interpret the scene, build an appropriate map, and localize itself relative to this map" [20].

There has been much research done to try to resolve the technical obscurities of the SLAM problem. It is understood that being able to successfully and completely answer the characteristic questions "*what does the world look like?"* and "*where am I in the world?"* will give a robot full and true autonomy.

Though there are many methods utilizing SLAM, there are three main algorithms which are integrated into nearly all of them, including numerous publications: Extended Kalman Filter (EKF) SLAM [21], Graph-SLAM [22], and particle-filter SLAM [23].

EKF SLAM is known as a more traditional approach, and was the first proposed method for obtaining useable results.

EKF SLAM uses an extended state vector with the robot's pose and feature locations in the map. As the robot moves, measurements are recorded, and the state vector is updated using an extended Kalman filter. As new features are discovered and described, the state vector grows, along with the noise covariance matrix, making EKF SLAM computationally inefficient.

Graph-Based SLAM, initially proposed in [22], uses the notion that the SLAM problem can be viewed as a set of robot locations and features, with constraints of relative position between them. Error propagation can be corrected by allowing these constraints to "flex" or adjust to updated locations. Therefore, finding the state of minimal energy is the solution to the SLAM problem.

Finally, particle filter SLAM works by randomly sampling the position error distribution [23]. Each particle has a corresponding estimate of the robot's path as well as locations of every feature in the map. With every measurement update an "importance factor" is assigned based on the probability of observing that measurement. According to this importance factor, the distribution is resampled with the same amount of particles, and feature locations are updated.

2.4 Path Planning and Obstacle Avoidance

The concept of path planning and avoiding obstacles has been around since autonomous robots first appeared, even before they were mobile [20]. Industrial robots had to be able to accurately plan their routes and avoid collision with walls, objects, and

other robots. While some roboticists may think of path planning as a solved problem, it actually has transitioned into an optimization problem.

Graph-search based methods such as A* [24], D* [25], Voronoi diagrams [26], cell decomposition, breadth- or depth-first, Dijkstra's algorithm[27], and all of their variants have long seen use in nearly every robotic platform imagined. These algorithms often work in free space using connectivity graphs, and are all appropriate for static (and sometimes slowly changing dynamic) environments.

Artificial potential field (APF) path planning has also proven useful in many robotic systems [26], and also simplifies calculations in most cases, as the path simply follows gradient descent methods. Of course, potential fields always run the possibility of running into local minima, for which there are several proven solutions [28, 29, 30].

Chapter 3: Hardware

3.1 Camera

Images were acquired using two cameras throughout this project. The first is an 8-megapixel Motorola Droid X2 phone camera, and the other is a Nikon Coolpix AW100. Both feature automatic brightness and contrast adjustment, as is industry standard [31].

It is important to note that the camera selection is irrelevant, as the process developed and tested is designed to work with any type of color capable camera with resolution of at least 240x320, as all images are sampled at this resolution.

3.2 Computer

All computing was done on a desktop machine equipped with a 3.6 GHz i7 processor. While the developed techniques have only been tested via post-processing, combining the image processing techniques with the latest and fastest iterations of feature detectors and descriptors and matching algorithms should be somewhat trivial to obtain real time performance.

It is also important to note that nearly all code was written in MATLAB, a notoriously slow programming language. MATLAB was chosen for its invaluable selection of built in functions for prototyping purposes.

At a resolution of 240x320, one frame is processed for color-invariance in .0156 seconds on average. This is fast enough to produce over sixty frames per second, fast enough for additional implementations of SIFT (or other feature detection and description algorithms), particle filter software, and nearly any path planning and obstacle avoidance method desired.

Chapter 4: Algorithms

4.1 Color Invariance

Two models were developed to work concurrently to produce the best results in changing lighting conditions: L-2 normalization and log-chromaticity space projection. It is important to note that any method should ideally work in a real-time vision system. Therefore, many different image processing techniques were explored and tested, but point-wise operations were given heavy favor due to easy and fast matrix manipulation.

4.1.1 L-2 Normalization

The first model, derived from a method proposed by Xu et al. [18] to remove shadows from single images, involves computing the L-2 norm using the standard red-green-blue (RGB) values for each frame as inputs. Normalizing each pixel value gives the effect of setting all pixels to the same "intensity". It is important to note that this is not the same intensity value given in an indexed or grayscale image, but rather standardization of RGB values for each pixel in the image. This was accomplished using the standard L-2 norm formula:

$$r' = \frac{r}{\sqrt{r^2 + g^2 + b^2}}$$

$$g' = \frac{g}{\sqrt{r^2 + g^2 + b^2}} \quad (1)$$

$$b' = \frac{b}{\sqrt{r^2 + g^2 + b^2}}$$

Where *r*, *g*, and *b* are the red, green, and blue input values, respectively, and *r', g',* and *b'* are the normalized red, green, and blue outputs, respectively. Figure 1 shows the result of this operation.

Figure 1, L-2 norm image comparison. Original image (left) compared to L-2 color normalized image (right).

4.1.2 Log-Chromaticity Projection

The second model is based on log-chromaticity space projection presented by G.D. Finlayson et al. [19]. Chromaticity is defined as a description of color "quality" regardless of luminance. Chromaticity has two independent components: hue and colorfulness (or saturation).

This approach uses log-domain characteristics to create a color invariant image, projected into intensity values. Finlayson made the observation that similar colors in an image are projected onto lines having similar slopes in log-chromaticity space despite ambient and direct lighting (and subsequently, the property of luminance). When projected in the direction orthogonal to the lighting, a color-invariant image is obtained:

$$inv = \cos\theta \cdot \ln\left(\frac{r}{g}\right) + \sin\theta \cdot ln\left(\frac{b}{g}\right) \quad (2)$$

In this equation, *r*, *g*, and *b* are the red, green, and blue input values, respectively, Θ is the projecting direction, and *inv* is the color-invariant image product. The value of Θ was chosen to be 43.58°, shown experimentally by Li et al. to be an acceptable approximation. To finish out the model, an exponential operation is applied to the color-invariant image and histogram equalization is performed to make the matrix easier to view and also work with. The result is shown in Figure 2.

Figure 2, Comparison of original image (left) with log-chromaticity projected image (right). Note that shadowing in the path is nearly invisible, while color differences (such as in the bottom right of the image) are preserved.

4.1.3 Combining Models for Path Classification

Deciding what is and is not a path can be accomplished a number of different ways. Using only color information, region properties can be parameterized, and the algorithm can be trained on specific surfaces, vanishing points can be determined, or texture analysis can be done.

The proposed method assumes that the robot is initialized on a traversable path, or region. The image known to be path is then sampled and analyzed, and all regions within the frame having similar properties of normalized color and chromaticity are considered to be traversable terrain.

This is accomplished by first creating two binary images from the two models presented using thresholding, and applying the AND operator to combine them. Since both models are susceptible to different types of error, the AND operator significantly reduces the amount of noise in the image. Next, the largest blob is found and kept, while the rest of the blobs are discarded. The result is a binary image, where the terrain is classified as traversable or not on a per pixel basis. This step is illustrated in Figure 3.

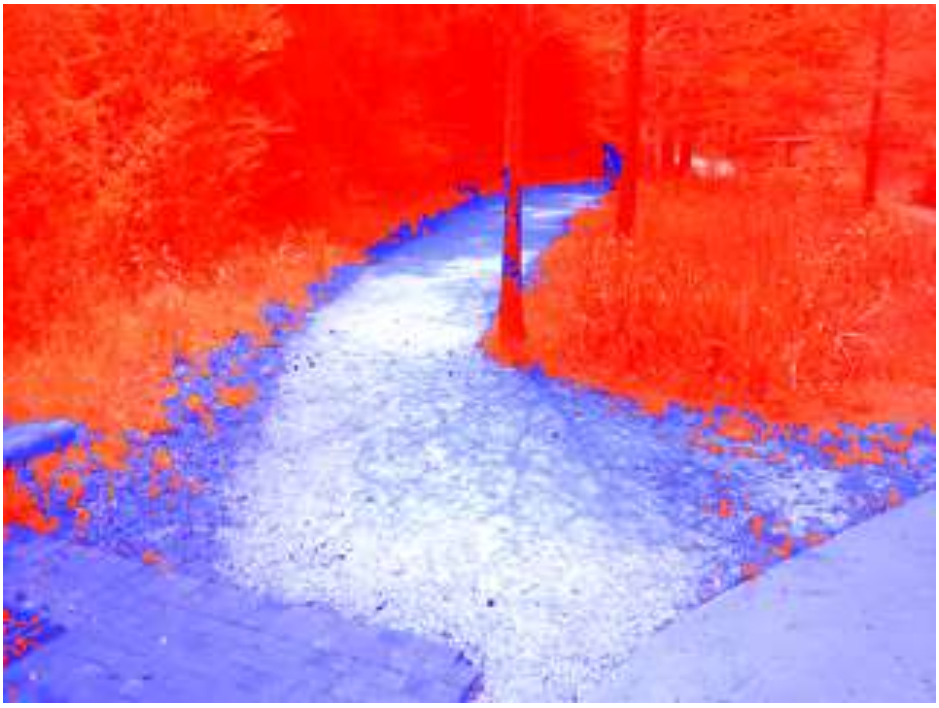


Figure3, Color invariant path detection. Binary mask overlaid on original image: Blue tinted regions represent detected path, and red tinted regions represent non-path.

Because there could potentially be obstacles or other important occlusions in the frame, the binary path is not simply flood filled to close gaps. Rather, a cost map can be created based on pixel density to quantify how safe a region is to travel. The downside to this is that even large obstacles in the background may not register until the robot gets closer, and a continuing and diminishing path in the distance will be down-weighted by the surrounding non-traversable pixels in the image. An example of such a cost map is shown in Figure 4.

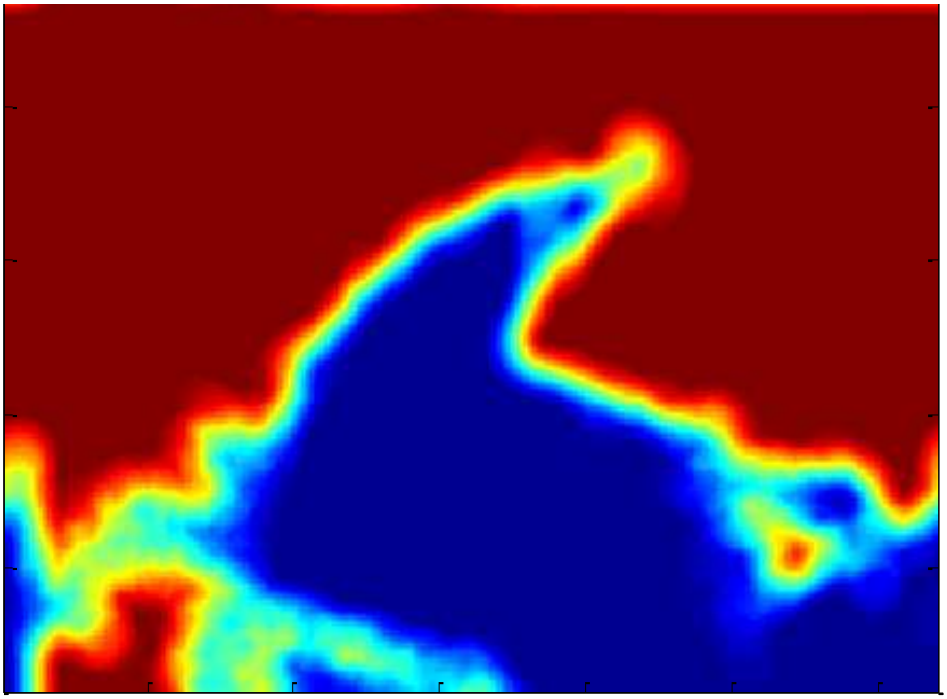


Figure 4, Cost map computed by pixel density. Blue levels represent low cost, versus red indicating high cost.

4.2 SIFT

Traditionally, feature matching algorithms are computed on entire image frames to extract as many usable features as possible. Due to spatial ambiguities, keypoints found in close two dimensional Euclidean distance may actually be infinitely separated in the third dimension. To overcome these ambiguities, many SLAM algorithms rely on camera calibration to estimate the distance a keypoint lies from the camera. Calculating locations of keypoints in three dimensions requires complex and computationally inefficient affine transform solutions.

To avoid making these calculations, the SIFT keypoint detection and descriptors are only computed on the binary classified output from the method of path identification in 4.1. This effectively restricts all keypoints to two dimensional space, i.e. those strictly lying on the ground plane within the detected path (see Figure 5). The important assumption is made that the ground is relatively flat, meaning that rapid changes in incline or altitude are not made. This allows a single projective spatial transformation (discussed in 4.5) to estimate the global two dimensional coordinates of each keypoint.

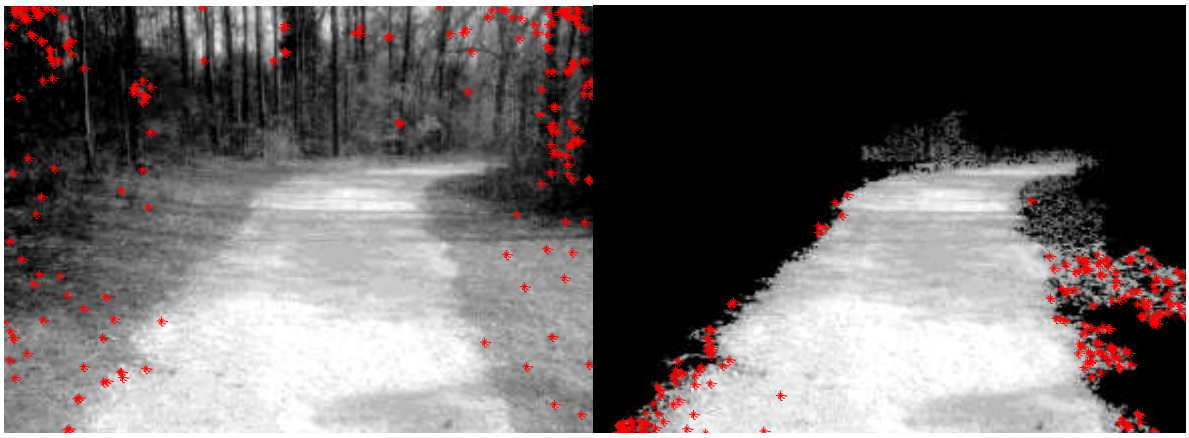


Figure 5, Comparison of full frame feature detection, and the proposed ground plane restrictions.

The SIFT algorithm first detects keypoints using a Noble's harmonic mean keypoint detector [36], defined as:

$$Hm = \frac{det(Hr)}{trace(Hr)} \tag{3}$$

Where *Hr* is the classic Harris corner detector [32], defined as:

$$Hr = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial x \partial y} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} \quad (4)$$

Noble's harmonic mean detector was chosen to simplify use by eliminating the need for tuning the Harris detector.

Once keypoints are located, a Normalized Laplace operator is used to find the characteristic scale for each corresponding keypoint. Local maxima are found and a threshold is used to determine if that keypoint is a "corner." This allows the keypoint descriptor to be invariant to scale.

After the characteristic scales are computed, a Gaussian blur function is applied to each local area around keypoints, each according to its scale. Next, the gradient is calculated at each keypoint, and a weighted histogram is computed to find the keypoint's main orientation. Main orientation is the rotation describing the keypoint, and falls into one of eight bins for searching. In the case where a keypoint does not have a main orientation (i.e., the histogram is flat), the keypoint is rejected.

A SIFT keypoint descriptor, similar to Lowe's method [11] is then applied by sampling the local image of each keypoint according to its characteristic scale and main orientation. The result is a 128 value vector describing a keypoint. These vectors are unique to each keypoint, and are compared in subsequent images to find matches.

Keypoints are tracked by searching for similar descriptors in an expected location window. That is, when a robot is moving, keypoints are expected to move only a small amount, thus only a small region around the initial detected keypoint is searched for the

corresponding match in the next image. This helps speed up computation time when matching features.

4.3 Optical Flow

Optical flow was implemented using Roborealm computer vision software. Roborealm's optical flow module uses a block matching strategy similar to the method presented in [35]. Since only the global motion is of interest, global horizontal and vertical vectors can be saved to file for post processing, or serially transmitted to any other software.

4.4 Particle Filter

A particle filter was used to filter the results from the SIFT algorithm outputs. Particle filters are based on Monte Carlo methods [33] of randomly sampling the robot's two dimensional belief distribution. This method allows the system to model any distribution or non-linear relationship, a powerful tool when tackling the SLAM problem.

At each time step (or image frame), K particles contain estimates of the robot's past and present poses $X_t^{[k]}$ and global feature positions having mean $\mu_{t,i}^{[k]}$ and covariance matrices $\sum_{t,i}^{[k]}$. Each particle is then defined as:

$$Particle[k] = X_t^{[k]}; (\mu_{t,0}^{[k]}, \sum_{t,0}^{[k]}); (\mu_{t,1}^{[k]}, \sum_{t,1}^{[k]}); \ldots ; (\mu_{t,n-1}^{[k]}, \sum_{t,n-1}^{[k]}); \quad (5)$$

Where *k* is the particle index, *t* is the time step, and n is the total number of features in the map.

Because keypoint locations are restricted to the ground plane, global locations of keypoints are estimated based on a linearized relationship between the robot's pose when

viewing the keypoint and the keypoint's location in the image frame. Furthermore, the assumption is made that the robot's pose within the image is exactly at the bottom center of the frame.

Each keypoint's location is then linearized by finding the polar coordinates of the keypoints from the origin, taken at the robot's pose. Each feature, then, has a perceived global location of:

$$x_i = X_t(x,y) + r_t^{6/5}[\, sin\Theta_t,\ cos\Theta_t] \tag{6}$$

Where x_i is the location of the ith feature, X_t is the pose of the robot at time step t, r_t is the radial distance of the feature in the image frame at timestep t, and Θ_t is the angle between the keypoint and image centerline. The exponent 6/5 on r is the relationship between radial distances of pixels versus actual units in the global scene, and was chosen experimentally.

At each time step, keypoint mean locations and covariance matrices are updated for each particle using the Kalman filter update rule.

With each movement u_t each particle updates its robot pose estimation, and makes another observation z_t. Next, each particle's *importance factor* $w_t^{[k]}$ is computed as the probability of z_t given the particles previous states and observations:

$$w_t^{[k]} = p(Particle[\mathrm{k}]|\, z_t\ ,\ u_t, n_t\ )/\ p(Particle[\mathrm{k}]|\, z_{t-1}\ ,\ u_t\ , n_{t-1}) \tag{7}$$

Montemerlo et al. shows in [23] that this representation can be mathematically approximated with the integral:

$$w_t^{[k]} \approx \int p(z_t|X_t\ ,\ Particle[\mathrm{k}]_t\ ,\ n_t)\ p(X_t)dX_t \tag{8}$$

This calculation comes with the common assumption of a uniform particle distribution.

Based on this importance factor, particles are then resampled to replace those with low importance factors with those with higher ones, and means and covariance matrices are updated. Resampling the particles each time with updated importance factors on a per particle basis allows the algorithm to assess different observation data association concurrently. This makes the particle filter SLAM solution more robust than traditional EKF methods, as particles with erroneous estimates will lie further from the center of the belief distribution, and thus less likely to be chosen when resampling.

4.5 SLAM

4.5.1 Localization

As mentioned in section 4.2, the SLAM problem is greatly simplified by restricting keypoints to the ground plane and within the perceived path. Each feature found is stored in a state vector containing a given identification number, descriptor info, and global x and y coordinates.

Given the location estimations of the SIFT descriptors from the particle filter output a pose estimate can be generated.

4.5.2 Map Building

For new maps (meaning the robot has never seen or navigated in this area before), the map is first initialized, and the starting pose of the robot is always [0 0 0] (x,y,theta). Using filtered motion estimation information from the output of the optical flow/SIFT motion estimators, the robot's pose change is updated for every movement.

Using pose information, a projective spatial transformation is applied to the binary output of traversable terrain to correct for perspective:

$$[x \;\; y \;\; 1] = [u \;\; v \;\; 1] \cdot T \tag{9}$$

Where *T* has the form:

$$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$$

and $h_{00}, h_{01}, \ldots, h_{21} = \boldsymbol{p}$ are parameters of the transformation to be solved. This can be done by iteratively solving for parameters $\boldsymbol{p}$ through minimizing the non-linear least squares problem, using robust least-squares, or RANSAC [34] (or its variants). A comparison of the binary image before and after this transformation is shown in Figure 6.

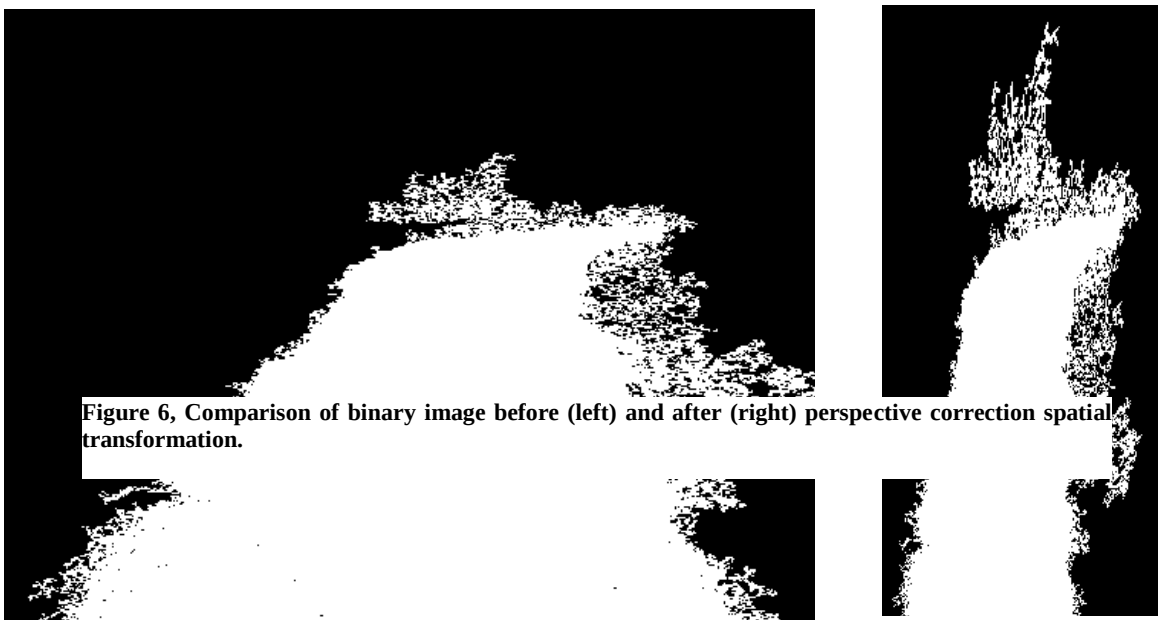


Figure 6, Comparison of binary image before (left) and after (right) perspective correction spatial transformation.

The resulting transformed image is then zero padded to create a "chunk." The chunk is then rotated and shifted to match the robot's pose estimation, and the chunk is ORed with the global map. The result is a binary map with real—not approximated—edges of the perceived path. Also, because of the OR operation applied, permanent obstacles are preserved while moving obstacles become filled. An example of this operation is shown in Figure 7.

Figure 7, Illustration of binary map building.

If instead a cost map is desired, the same process can be applied to the cost map information generated using the pixel density function output. The results can then be added and normalized to obtain a global cost map. This map will have the same features from the binary map, including real edges and obstacle preservation (see section 4.1.3).

4.6 Path Planning and Obstacle Avoidance

One long standing sensor problem in the area of robotics is the ability not just to plan an appropriate path, but where in the world the path is allowed to lie. Certainly other sensor systems have advantages over computer vision techniques when it comes to

identifying large obstacles such as rocks or trees, but they most definitely fall short to accurately paint the whole scene.

Consider a robot navigating in an urban environment: range-finding sensors will certainly do a good job locating people, poles and cars, but may not notice any difference between roads, medians, and sidewalks. Similarly, in rural areas, there may be no perceivable difference in a point cloud between the road or path and the surrounding field. If a robot using only range-finding sensors ventured into a mud pit, it would surely never recover.

Some systems supplement range-finders with computer vision, such as the famous robot Stanley that won the DARPA Grand Challenge in 2005. Stanley used an array of five laser sensors to scan the surrounding terrain to determine where the vehicle could navigate. Because of the limited range of the sensors (about 22 meters), the vehicle was limited to a speed of 25 mph. The Stanford team supplemented the laser data by analyzing the color of the path deemed navigable by the laser data, and searched a video feed for areas with similar color characteristics [10]. Due to the video's much farther field of view, the vehicle was able to path plan while traveling up to 35 mph, going on to be the first robot to ever complete the challenge.

Following this notion, similar techniques using only color information (discussed in 4.1) are applied to obtain a per-pixel binary image of perceived traversable areas. As previously discussed, there are many different ways to interpret the binary information to use in path planning.

Perhaps the most simple and straightforward way to use the binary image is to consider the "traversable" pixels the workspace, where any point is safe to travel. This

allows any path planning algorithm such as A* or D* (and their variants) to be directly applied. Note that this can be computed on a local or global scale. Applying the algorithm frame by frame, making the assumption that the robot's pose is located at the bottom center of the frame allows the robot to navigate dynamic and permanent obstacles. Applying the same spatial transformation that is applied to feature and view mapping using the robot's priors can give a transformation to and from the global workspace. This allows a path to be computed in the global workspace, and transformed to the local workspace for finer control and obstacle avoidance.

Of course, the downside to using a strict binary local workspace is the influence of noise. Without any further processing or decision making, a single pixel declared not traversable can hinder the robot's efficiency in traveling, and perhaps even impede it completely.

To avoid this, the binary image may be quickly and efficiently converted to a cost map or potential field map by applying a Gaussian blur function to the entire image. Inverting the result effectively computes the pixel density which is proportional (given gains and other parameters) to the desired cost or potential field map.

Cost maps have the advantage of smoothing out rigid motions and hindrances of a binary workspace while maintaining arithmetic calculations when applying the same path planning algorithms. A* and D* can just as simply be computed on pixels containing a value between zero and one versus strictly zero or one.

Potential field path planning has also proven useful in many robotic systems, and also simplifies calculations in most cases, as the path simply follows gradient descent methods. This map can be calculated by extending the cost map by one step: "tilting" the

map by either raising the bottom or lowering the top values (and all values in between according to a linear function), or tilting towards the target location.

Of course, potential fields always run the possibility of running into local minima, and for the sake of demonstration, the proposed solution will be to always have the target location at a global minimum, and minimizing the cost function of travelling to that value.

Chapter 5: Experiments and Results

As the title of this thesis suggests, the methods proposed are designed to enhance the current methods of visual navigation—not replace them. To this end, a series of tests were designed and comparisons will be drawn.

5.1 Color Invariance and Path Classification

To evaluate the method for path classification, a series of images will be compared to the tried-and-true image segmentation method of k-means clustering. It should be noted that the k-means clustering is hand-tuned for each picture to ensure that the most accurate account of the true path is given and compared to the color-invariant method proposed. Because both images being compared are binary, the simplest method of sum of absolute differences (SAD) was used, in the form:

$$E_{SAD} = \sum_{i} \left\| \boldsymbol{X}_{k-means} - \boldsymbol{X}_{compared} \right\| \tag{8}$$

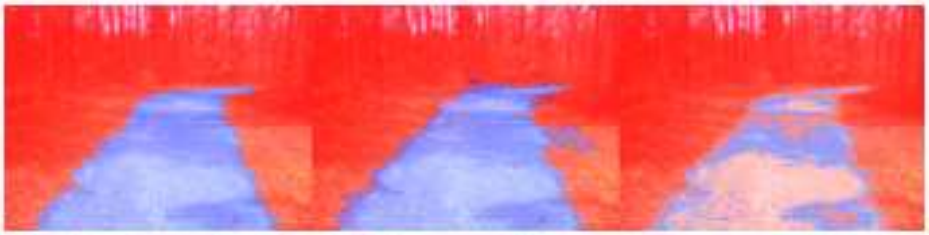
Where $\boldsymbol{X}_{k\text{-}means}$ is the k-means binary image and $\boldsymbol{X}_{compared}$ is the path extracted image using the methods of color invariance path classification, and path classification without color invariance, respectively. Table 1 shows these results, along with Figure 8, which shows the comparison of classifications. Note that in one instance the proposed color invariant path classification actually outperforms k-means clustering.

Image	E_{SAD} *with Color Invariant path detection*	E_{SAD} *without Color Invariant path detection*
1	1764	19341
2	6716	15637
3	8775	19218
4	415	8129
5	5403	10165
6	1208	11746
7	4518	18027
8	4925	17318
9	11775	24049
10	Outperformed k-means	--

Table 1, Sum of absolute differences between k-means image segmentation, color invariant path detection, and color based path detection.

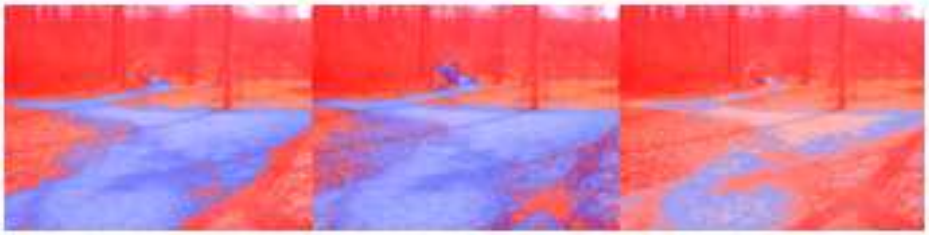
Along with comparative accuracy in determining where the path lies, color invariant path detection also performs much faster than k-means clustering, as discussed in section 3.2.

1)



2)

3)



4)

5)

6)

7)

8)

9)

10)

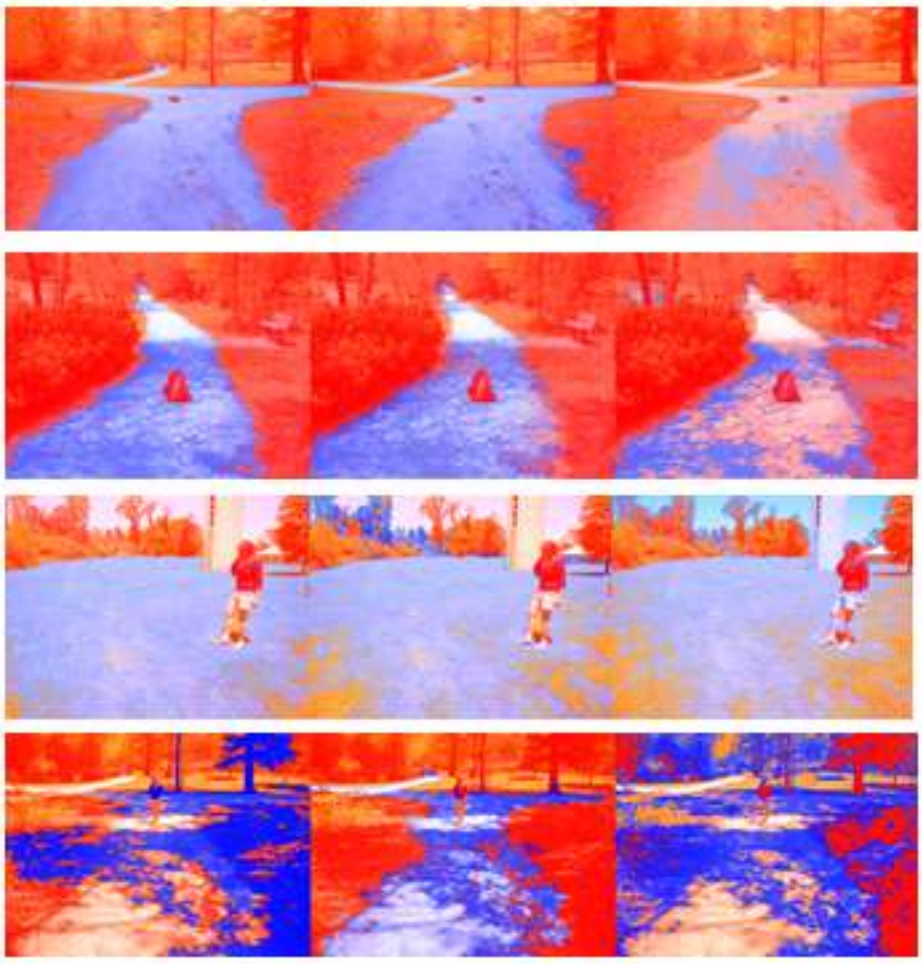


Figure 8, Path detection method comparison. Methods compared using hand-tuned k-means clustering (column 1), color invariance as proposed in this thesis (column 2), and color classification (column 3) on multiple surface types. Rows 1 through 10 correspond with SAD data in table 1. Note that color invariance outperforms color classification and is comparable to k-means clustering, even outperforming it in some instances, such as 10.

5.2 Color Invariance with SIFT

Although SIFT feature descriptors already have some color invariance properties, even more color invariance can be achieved by performing SIFT on the log-chromaticity projection, as the values in the chromaticity space are mapped to intensity values. Figure 9 shows a comparison of a Harris corner detector (red asterisk) and Shi and Tomasi's minimum eigenvalue corner detector (blue asterisk) applied in the tradition grayscale intensity image compared to the log-chromaticity projected intensity image.

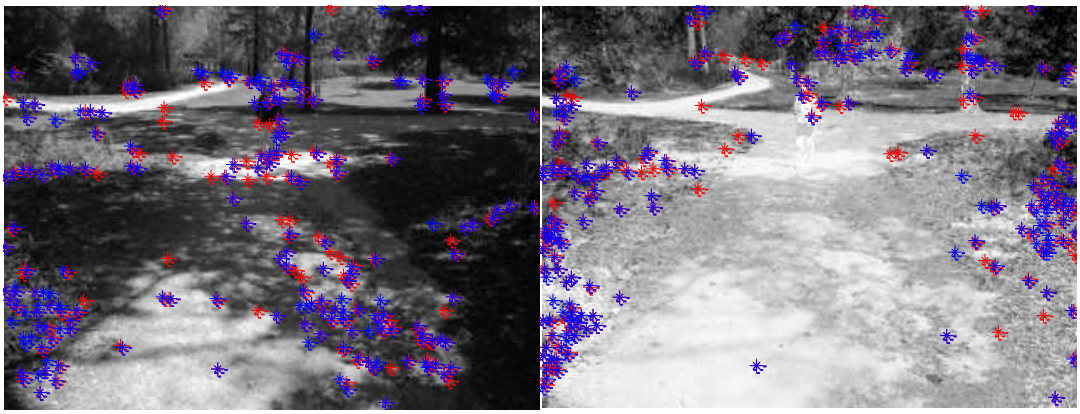


Figure 9, Comparison of keypoint detectors on traditional grayscale (left) vs. log-chromaticity projection (right).

It is easy to see that there are many more keypoints detected in the grayscale (left) image which fall directly onto shadow edges and features. It is precisely these types of keypoints a robot wants to avoid when building a map, as they directly rely upon the luminance, weather, and even wind direction and speed; all of which are time variant.

Another measure of performance critical to localization derived from SIFT is the number of keypoints that are matched between images. Ideally, it is preferred to have a relatively small number of very strong keypoints that can be uniquely identified across multiple frames. Figures 10 and 11 show two sets of images captured at different times

of day, with different lighting conditions. Comparing the grayscale SIFT feature matching with the log-chromaticity projection SIFT feature matching shows that in many cases log-chromaticity meets or exceeds the number of matches as traditional grayscale techniques. Table 2 shows a comparison of the number of feature matches between grayscale and log-chromaticity projection images among seven different data sets (the first two rows correspond to Figures 10 and 11, respectively).

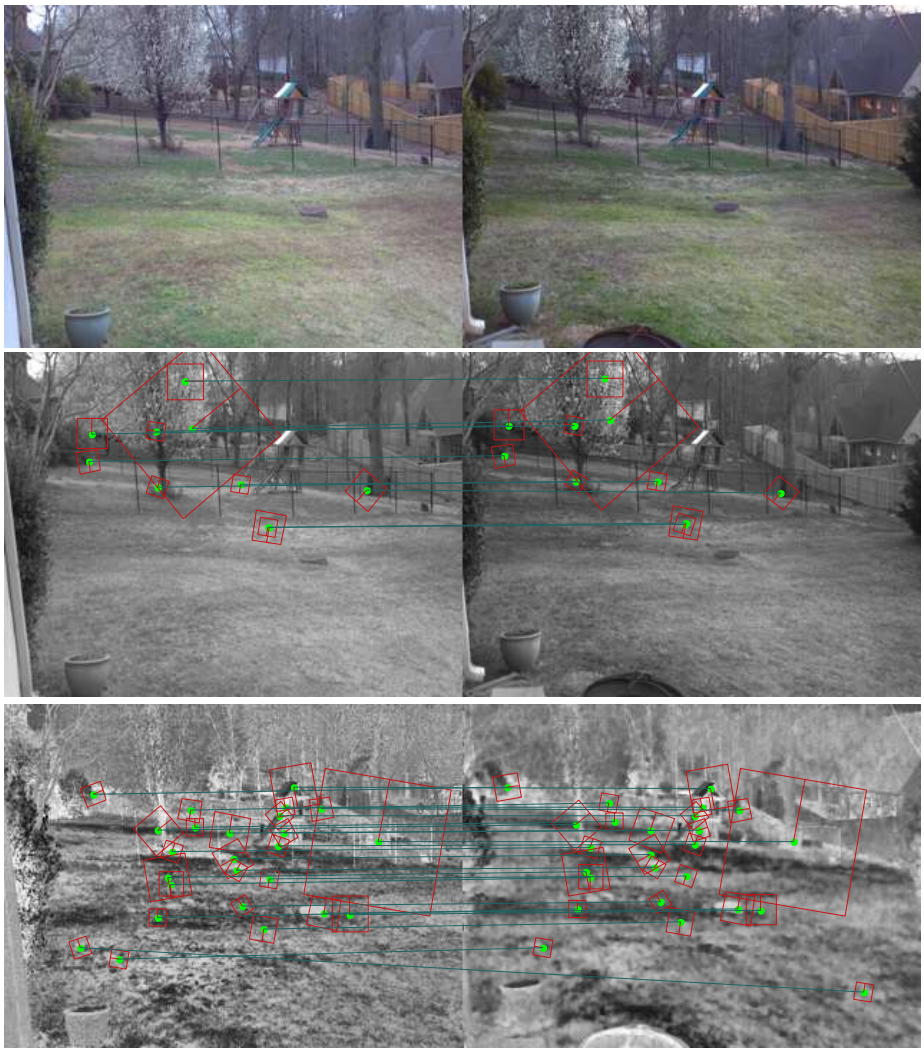


Figure 10, Comparison of SIFT feature matching (1). Original image (top), grayscale image (middle), and log-chromaticity projected image (bottom).

Figure 11, comparison of SIFT feature matching (2). Original image (top), grayscale image (middle), and log-chromaticity projected image (bottom). Note the incorrect matches from the grayscale images due to shadowing.

Number of grayscale matches	Number of log-chromaticity matches
10	25
4	4
25	22
125	103
34	18
1	1
3	4

Table 2, Comparison of the number of feature matches between grayscale and log-chromaticity projection images

5.3 Color Invariance with Optical Flow

Perhaps one of the most significant contributions the path detection alone could make is the ability to build a map with optical flow. As discussed in section 2.1.3 and 4.3, optical flow motion estimation is accomplished by measuring changes in pixel values, so there is not much thought given to ambient conditions. A robust optical flow algorithm will give the same results despite hard or soft shadowing, or type of environment.

For motion estimation on a mobile robot, the assumption is usually made that the scene the robot is in is mostly stationary; that is, most objects or obstacles are static. Extending this concept, this also means that most of the motions attributing to optical flow measurements are from the robot itself moving through the scene. For a robot, the sensing is analogous to humans feeling acceleration. Imagine closing your eyes while

riding in a car—you know when you move, and even which way you move, but you have no idea what objects you are moving in reference to.

Unless the surface being traveled on is homogenous (which is atypical for outdoor environments), it is very difficult to model the traversable areas using optical flow. However, when combined with the color invariant path recognition presented in this thesis, it is easy to create view and cost maps for initial mapping purposes (it should be noted that without proper filters and estimators error propagation will show large drifts from the true traveled path, as with any motion estimation).

The map building procedure described in 4.5.2 is applied along with global motion estimates from the optical flow algorithm to produce the results shown in Figure 12, and the view map in Figure 13. These results can be compared to the actual path as shown by Google Earth in Figure 14.

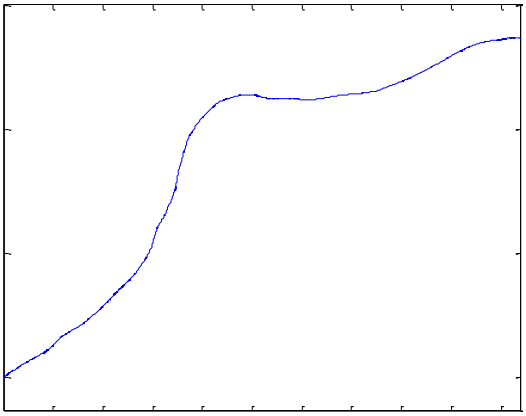


Figure 32, Path estimation from optical flow global motion measurements.

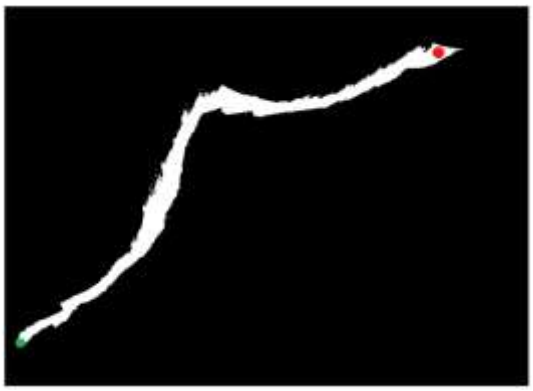


Figure 43, View map built from optical flow motion estimation and color invariance path detection. The green dot represents the starting point and the red dot represents the end point.

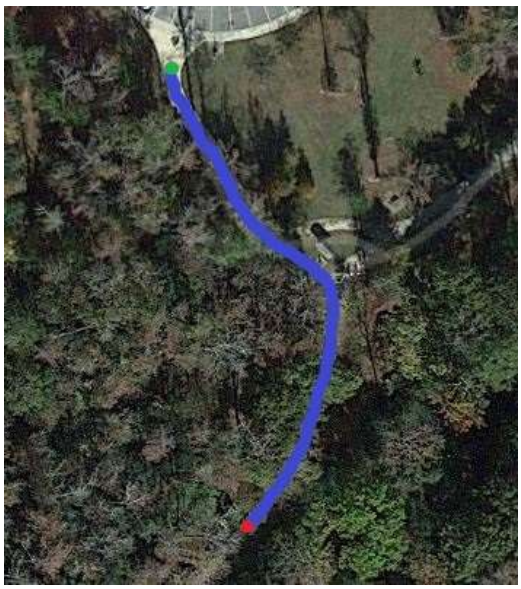


Figure 54, Google Earth image of path traversed in Figures 12, 13, 15, and 16. The green dot represents the starting point and the red dot represents the end point (used with permission under Google Earth policies).

5.4 Color Invariance with SLAM

Once keypoints have been detected, described, and matched, the SLAM problem remains the same. Using the path detection along with SLAM offers the advantage of being able to build not just a feature map and the path traveled, but a view map as produced in section 5.3. An example of this is shown in Figure 15, where the same path in Figure 12 is built instead using SIFT feature detection with the SLAM algorithm discussed in 4.5.

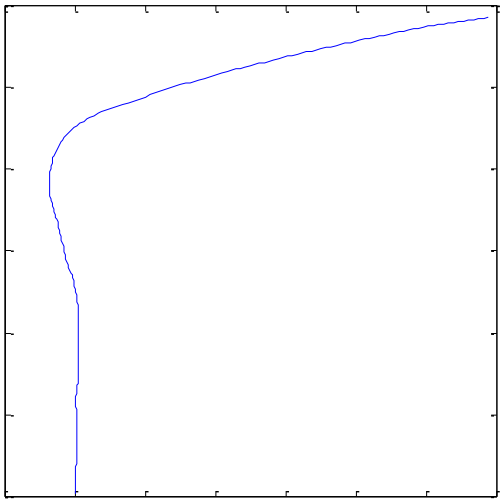


Figure 6, Path estimation from SIFT global motion measurements.

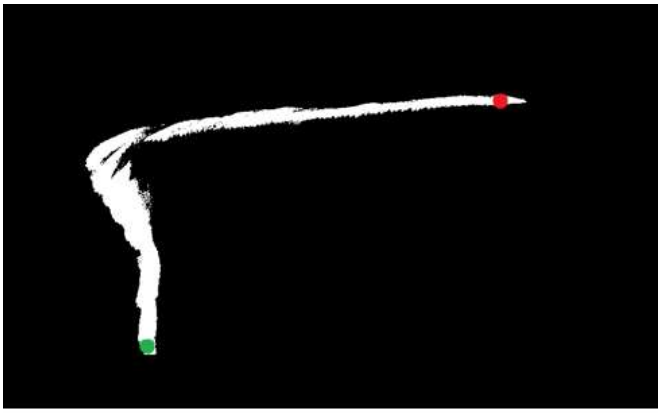


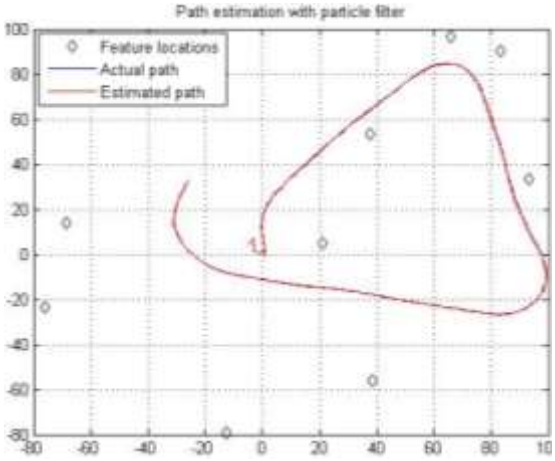
Figure 7, View map built from SIFT motion estimation and color invariance path detection. The green dot represents the starting point and the red dot represents the end point.

While using a particle filter in theory is a simple concept, the logistics of keeping track of thousands of features is non-trivial. Global feature locations should be updated upon every instance of detection, however features are not necessarily in subsequent frames. Thus, time constraints prevented the use of actual global feature locations. Instead, visual odometry data was taken to be the "actual" path and global feature locations were simulated using an arbitrary number of features. Gaussian observation noise was added to simulate actual measurements in x and y directions. The motion (visual odometry) parameter was taken to be the measured movement of matched keypoints, and the observation parameter was taken to be position using multilateration from observed keypoints.

Figures 17 through 19 show the true path via Google Maps, the visual odometry path (taken as actual path for simulation purposes) with simulated feature locations and pose estimations after filtering with the described particle filter, and the standard deviations of the particle filter clouds in two directions. The results are shown for multiple paths.

a)

b)



c)

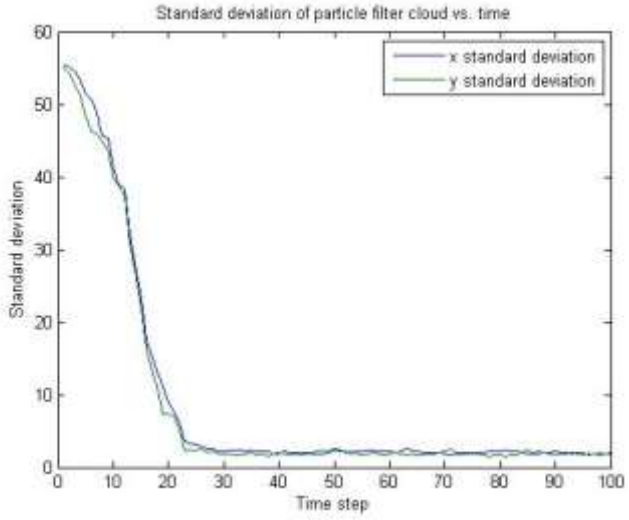
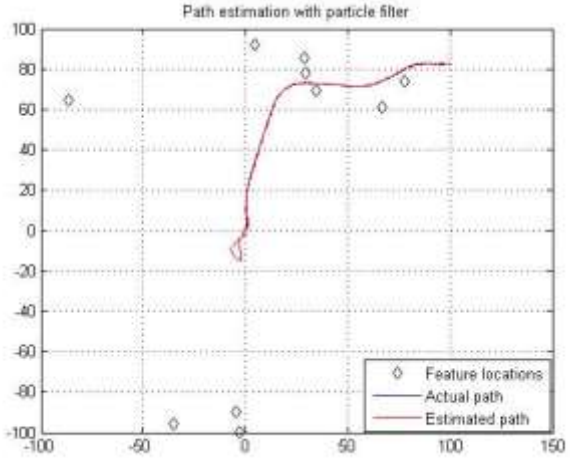


Figure 87, Particle filter demonstration using color invariant features (1). a) True traversed path (Google Earth), b) visual odometry path (blue) overlaid with particle filter estimated path (red), and c) standard deviations of particle filter cloud.

a)

b)



c)

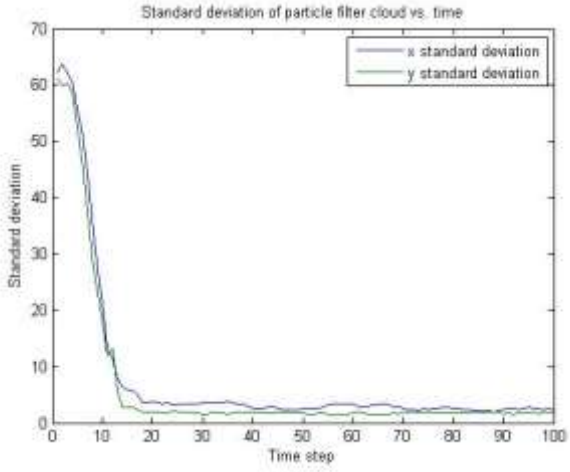
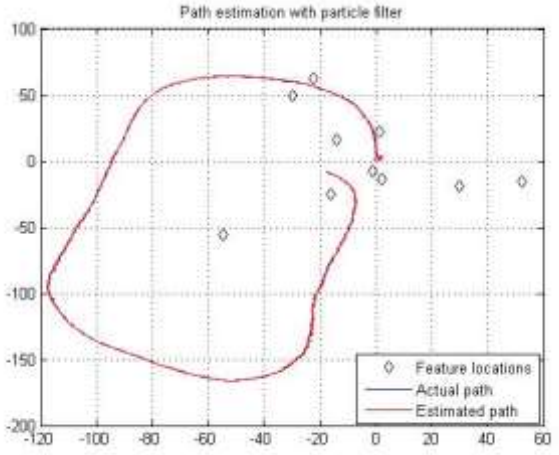


Figure 9, Particle filter demonstration using color invariant features (2). a) True traversed path (Google Earth), b) visual odometry path (blue) overlaid with particle filter estimated path (red), and c) standard deviations of particle filter cloud.

a)

b)



c)

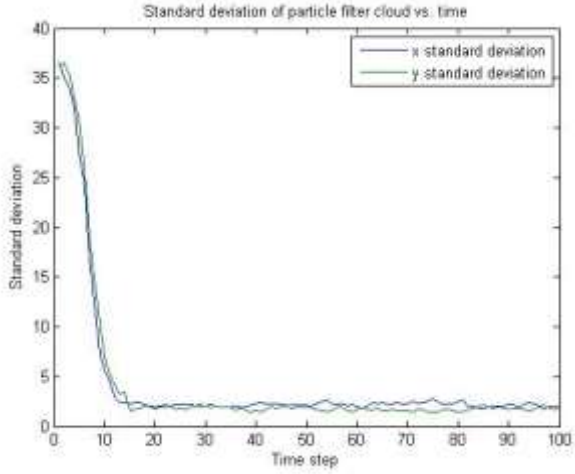


Figure 10, Particle filter demonstration using color invariant features (3). a) True traversed path (Google Earth), b) visual odometry path (blue) overlaid with particle filter estimated path (red), and c) standard deviations of particle filter cloud.

5.5 Color Invariance with Path planning

Given the binary output from color invariance path detection, practically any navigation algorithm may be run. Taking the bottom center of the image to be the robot's location (and thus start point), a path was formed to a hand selected goal (for demonstration purposes) using different algorithms as shown in Figure 20.

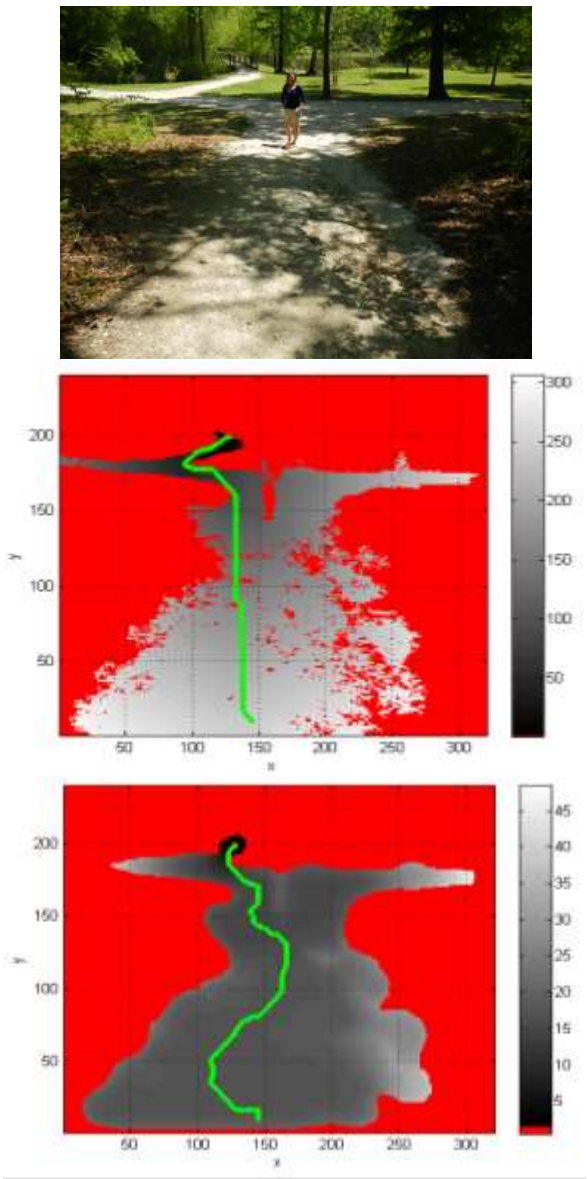


Figure 20, Path planning demonstration with color invariance. Original image (top) compared to binary (middle) and pixel density (bottom) applications of the D* algorithm.

Chapter 6: Summary and Future Work

This chapter serves to summarize the work presented in this thesis, and suggest future work for continuing the presented topic of study.

6.1 Summary

This thesis has shown that using color invariant properties of scene images can significantly reduce perception errors from traditional computer vision techniques. The fast computation of log-chromaticity projection and L-2 normalization serves to specifically improve upon visual navigation techniques for mobile robots operating outdoors.

The results also show that using computer vision as a sole means of data acquisition is not only feasible, but offers advantages such as path recognition and color information over other sensory methods.

6.2 Future Work

As this work was devoted to prototyping a new method, little effort was given to the computation speed of the particular algorithms used. Other research has shown very efficient and fast algorithms that perform all the necessary steps to obtain the results achieved from the prototype methods, and could be implemented for real-time use.

As mentioned in section 5.4, considerable effort needs to be given toward a robust global feature tracking algorithm. There is not only a complex geometric relationship between feature locations in a video frame and their respective global coordinates, but correctly updating observed locations for use in a particle filter is a difficult task.

There have also been considerable accomplishments in the research of color-specific descriptors, and comparisons should be drawn between the methods presented in this thesis and current state of the art color descriptors.

References

[1] Dederick, Z. (1868). *Improvement in steam-carriage.* US Patent No. 75874 A, Washington, DC: U.S. Patent and Trademark Office.

[2] Goto, T., Inoyama, T., & Takeyasu, K. (1974). Precise insert operation by tactile controlled robot. *Industrial Robot: An International Journal*, *1*(5), 225-228.

[3] Leonard, J. J., & Durrant-Whyte, H. F. (1992). *Directed sonar sensing for mobile robot navigation* (Vol. 448). Dordrecht: Kluwer Academic Publishers.

[4] Raibert, M., Blankespoor, K., Nelson, G., & Playter, R. (2008, July). Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress*(pp. 10823-10825).

[5] Zhang, Z. (2012). Microsoft kinect sensor and its effect. *MultiMedia, IEEE,19*(2), 4-10.

[6] Boden, M. A. (2006). *Mind as machine: A history of cognitive science* (Vol. 1). Oxford University Press.

[7] Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer.

[8] Schantz, H. F. (1982). *History of OCR, Optical Character Recognition*. Recognition Technologies Users Association.

[9] Bobbit, R., Connell, J., Haas, N., Otto, C., Pankanti, S., & Payne, J. (2011, January). Visual item verification for fraud prevention in retail self-checkout. In*Applications of Computer Vision (WACV), 2011 IEEE Workshop on* (pp. 585-590). IEEE.

[10] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., ... & Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of field Robotics*, *23*(9), 661-692.

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

[12] Ke, Y., & Sukthankar, R. (2004, June). PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (Vol. 2, pp. II-506). IEEE.

[13] Mikolajczyk, K., & Schmid, C. (2004). Scale & affine invariant interest point detectors. *International journal of computer vision*, *60*(1), 63-86.

[14] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer Vision–ECCV 2006* (pp. 404-417). Springer Berlin Heidelberg.

[15] Leutenegger, S., Chli, M., & Siegwart, R. Y. (2011, November). BRISK: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (pp. 2548-2555). IEEE.

[16] Nakayama, K., & Loomis, J. M. (1974). Optical velocity patterns, velocity-sensitive neurons, and space perception: a hypothesis. *Perception*, *3*(1), 63-80.

[17] Sun, D., Roth, S., & Black, M. J. (2010, June). Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (pp. 2432-2439). IEEE.

[18] Xu, L., Qi, F., & Jiang, R. (2006, October). Shadow removal from a single image. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on* (Vol. 2, pp. 1049-1054). IEEE.

[19] Finlayson, G. D., Hordley, S. D., Lu, C., & Drew, M. S. (2006). On the removal of shadows from images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *28*(1), 59-68.

[20] Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

[21] Smith, R. C., & Cheeseman, P. (1986). On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, *5*(4), 56-68.

[22] Lu, F., & Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous robots*, *4*(4), 333-349.

[23] Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002, July). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In*AAAI/IAAI* (pp. 593-598).

[24] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, *4*(2), 100-107.

[25] Stentz, A. (1994, May). Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on* (pp. 3310-3317). IEEE.

[26] Latombe, J. (1991). *Robot Motion Planning.* Kluwer Academic, Norwood, MA.

[27] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs.*Numerische mathematik*, *1*(1), 269-271.

[28] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, *5*(1), 90-98.

[29] Feder, H. J. S., & Slotine, J. J. (1997, April). Real-time path planning using harmonic potentials in dynamic environments. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on* (Vol. 1, pp. 874-881). IEEE.

[30] Li, G., Yamashita, A., Asama, H., & Tamura, Y. (2012, August). An efficient improved artificial potential field based regression search method for robot path planning. In *Mechatronics and Automation (ICMA), 2012 International Conference on* (pp. 1227-1232). IEEE.

[31] Lee, M. H. (1996). *U.S. Patent No. 5,546,134*. Washington, DC: U.S. Patent and Trademark Office.

[32] Harris, C., & Stephens, M. (1988, August). A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, p. 50).

[33] Metropolis, N., & Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, *44*(247), 335-341.

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

[35] Davis, C. Q., Karu, Z. Z., & Freeman, D. M. (1995, November). Equivalence of subpixel motion estimators based on optical flow and block matching. In *Computer Vision, 1995. Proceedings., International Symposium on* (pp. 7-12). IEEE.

[36] Noble, J. A. (1988). Finding corners. *Image and Vision Computing*, *6*(2), 121-128.