Image-Based Localization for Mobile Robotic Navigation

by

Philip Barber
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Sciences
Computer Science

Committee:

Dr. Zoran Duric______________________________ Co-Director
Dr. Jana Kosecka______________________________ Co-Director
Dr. Harry Wechsler___________________________
Dr. Henry Hamburger________________________
Dr. Lloyd Griffiths___________________________ Dean, School of Information
Technology and Engineering

Date:  ________________________________ Spring Semester 2002
George Mason University
Fairfax, VA
Image-Based Localization for Mobile Robotic Navigation

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Philip Barber
Bachelor of Science
George Mason University, 1991

Directors: Dr. Zoran Duric, Assistant Professor and Dr. Jana Kosecka, Assistant Professor
Department of Computer Science

Spring Semester 2002
George Mason University
Fairfax, VA
Dedication

This project is dedicated to my parents who instilled in me a love for learning. Also to my wife, whose unimaginable support has made my continued learning possible. And finally, also to my children in the hopes that they, too, will learn throughout their lives.
Acknowledgements

I would like to recognize the vast number of hours spent in support of this project by my advisors, Dr. Zoran Duric and Dr. Jana Kosecka. I would also like to recognize my wife, Shari, who gave up much of her free time to allow me to work on this project.
# Table of Contents

Abstract ...................................................................................................................... viii  
1. Introduction......................................................................................................... 1  
  1.1. Motivation .................................................................................................. 1  
  1.2. Overview .................................................................................................... 2  
2. Related Works .................................................................................................... 3  
3. Methodology....................................................................................................... 6  
  3.1. Low Level Representation – Images & Features ........................................ 6  
  3.2. Intermediate Level Representation .............................................................. 6  
  3.3. Comparing Edge Histograms ....................................................................... 6  
  3.4. Learning Phase - LVQ ............................................................................... 6  
  3.5. Recognition Phase ...................................................................................... 6  
4. Experiments ........................................................................................................ 6  
  4.1. Results of Learning and Classification Experiments .................................. 6  
  4.2. Results of Learning and Classification Experiments Using Sub-Images..... 6  
  4.3. Different Edge Strength Thresholds ............................................................. 6  
  4.4. Distinguishing Abilities of Different Histogram Types ............................ 6  
  4.5. Similarity Matrix Showing Clusters ............................................................. 6  
  4.6. Pair-wise Over Time Graph ........................................................................ 6  
  4.7. Code Files After Teaching .......................................................................... 6  
5. Conclusion .......................................................................................................... 6  
  5.1. Future Work ............................................................................................... 6  
6. References........................................................................................................... 6
Table of Tables

Table 1. Distance values of different measurement techniques ........................................40
Table 2. Image J vs image K and image J vs image L.......................................................47
Table 3. Comparing a single image to other images using various techniques ...............57
Table 4. Code vector statistics ...........................................................................................62
Table 5. 100 iterations of the classification test with different parameters ......................53
Table 6. 100 iterations of the classification test using sub-images.....................................54
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Examples images. Note the similarity of image A &amp; B</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>Images from Figure 1 after edge detection</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>Edges after non-maximum suppression</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>Edge pixels after thresholding</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>Histograms of edge direction with $10^\circ$ bins</td>
<td>33</td>
</tr>
<tr>
<td>6</td>
<td>Comparing remaining pixels of light and dark image</td>
<td>34</td>
</tr>
<tr>
<td>7</td>
<td>Comparing histograms of similar images from different angles</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>Comparing histograms of similar images from different distances</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>Simple Sammon projection</td>
<td>43</td>
</tr>
<tr>
<td>10</td>
<td>Example of an image misclassified, then reclassified correctly</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>Comparing different threshold values</td>
<td>56</td>
</tr>
<tr>
<td>12</td>
<td>Similarity matrix show inter-histogram distances</td>
<td>59</td>
</tr>
<tr>
<td>13</td>
<td>Pair-wise comparisons over time</td>
<td>60</td>
</tr>
<tr>
<td>14</td>
<td>Sammon projection of all code vectors</td>
<td>63</td>
</tr>
</tbody>
</table>
Abstract

IMAGE-BASED LOCALIZATION FOR MOBILE ROBOTIC NAVIGATION

Philip M. Barber, M.S.
George Mason University, 2002

Thesis Directors: Dr. Zoran Duric, Assistant Professor and Dr. Jana Kosecka, Assistant Professor

This thesis describes a technique for using edge direction histograms with standard learning algorithms for image-based localization of mobile robots within man-made environments. The system operates in two phases: a learning phase and a localization phase. The data used for learning and localization are based on the edges of color images of the environment. Those edges are used to produce histograms which are treated as vectors representing each image. By treating the histograms as points in an n-dimensional space, the natural clustering of the data is obtained by the supervised algorithm to generate the database. By using existing pattern recognition techniques to examine the clusters, new images of the hallways traversed by a robot can be classified according to the cluster it most closely resembles.
1. Introduction

1.1. Motivation

After many years of research, autonomous navigation of robots remains elusive. Inertial navigation systems are unreliable due to errors in measurements that are difficult to predict and adjust for. Reliable navigation systems rely upon the installation of external stimuli to provide the robot with the information necessary to determine its location. Examples of these systems include visible clues such as colored strips or barcodes in the hallway, or hidden wires that are used to provide guidance. Systems such as these are often unattractive, expensive to install and/or expensive to operate and maintain. We would like to be able to implement a mobile robot that could be taught the layout of a building and that would then be able to guide itself through the corridors using only the stimuli already in existence. This goal significantly narrows the number of options available. For this project we used visible light-based images obtained by a camera on the mobile robot.

1.2. Overview

Our overriding goal in this project is to create a system that would be able to determine its location using images. We subdivided this problem into a number of areas that could be solved individually. First, we identify those pixels in images that we
believed most differentiated it from images of other locations. Next, we use existing
histogram techniques to represent those pixels in a way that would allow for efficient
comparisons. After implementing those algorithms we used the Learning Vector
Quantization algorithms to generate representative histograms based on a database of
images obtained a priori. Finally, we implement a classification algorithm that compares
new histograms with the representative histograms to determine which location the new
image most closely resembles.

In chapter 2 we review related works that led to this project. Chapter 3 contains a
description of the methods we used during this project. In chapter 4 we describe the
experiments we performed and the results obtained. In chapter 5 we give our conclusions.
Finally, in chapter 6 we list the references used during this project.
2. Related Works

Our work here is intended to be a step towards robotic navigation. We envisioned combining the techniques in this project with techniques developed by others for storing topological maps as graphs. The work documented in [1] and [2] describes storing topological maps as graphs. We believe that a robot could use our techniques to localize the robot while making macro-navigation decisions based on the graphs.

Once the choice of topological representation is established, the different works differ in what corresponds to a node of a graph. The most common choice is a visibility region associated with a landmark, where a landmark is a natural or artificial geometrical structure in the environment. In particular, our approach was quite similar to that described in the Map-based approach section of [3] in that we attempt to incorporate knowledge of what to expect into the system. Our overall algorithm is also very similar to the four steps for localization that they describe: acquire sensory information, detect landmarks, establish matches between observation and expectation, and calculate position [3]. Ultimately, we envision integrating our localization technique with incremental localization, also described in [3]. In that technique, information obtained at the robot’s current position is combined with knowledge gained from the previous position to determine the current location with a greater degree of reliability.

The presence of geometric landmarks enables accurate metric localization (i.e. the
robot can tell exactly what (x,y,è) he is at). In order to achieve our goal of localization, we focused on the recognition of landmarks. However, instead of choosing geometric representations of the landmarks in the environment, we studied so-called image-based techniques, where the appearance of a particular region will determine what place we are at. Using appearance-based landmarks for navigation has been investigated in the past. The landmarks used previously fall into two categories: artificial [4] and existing [5]. We considered our landmarks to be locations that were significant for robot navigation such as hallway intersections and doorframes. In order to recognize different landmarks/locations, we looked to previous work on recognizing objects. If you can consider that a location is a collection of objects, then techniques that successfully recognize objects may be able to be extended to recognize a static collection of objects. In [6] we are presented with a highly successful technique for recognizing objects. We decided to use similar techniques for recognizing locations. To do this, we created histograms of images obtained a prior and used those histograms as a database. The histograms were created from edge values calculated using standard techniques as described in [7]. After the histograms were created and stored as a database, the same process was performed on the new images that needed to be recognized.

In order to use the appearance-based techniques for modeling the environment, we needed to understand how to define a distance between two images. To compare the histograms from the database with the histograms from the new images, we looked at and tested several histogram measures. A large number of distance metrics have been documented such as unfolding which is described in [8] and the Earth Mover’s Distance
which is described in [9]. We decided to use histogram intersection and the chi-squared
distance measures as described in [6] and [7].

Finally, we also use pattern matching techniques to build a representation of our
database and to perform the recognition step. Learning Vector Quantization, as described
by Kohonen [10] provides an excellent tool for pattern matching. For our project we
wanted to take advantage of the work already done in this field, so we used an existing
package called LVQ_PAK with minor modifications of the distance calculations.
3. Methodology

3.1 Low Level Representation – Images & Feature

We began this project with a two-phase system in mind. The first phase, the learning phase, acquires images and processes them until a set of representative histograms are produced which represent the a priori knowledge of a mobile robot navigating an environment. The second phase, recognition, acquires new images which are processed the same way as the original images, but are then compared to the representative histograms to attempt to localize the mobile robot.

To acquire the images that were used to build the database, we simulated the camera on a mobile robot by using a digital camera mounted on a tripod approximately two feet off the ground. The camera, an HP 618 with a 2.1 megapixel CCD, was then used to acquire 640x480 images of the test environment. The images were stored in the camera as JPG files. Upon being downloaded to the test computer, the images were converted to PPM files for processing. While the images were being taken, each separate location in the environment was given an arbitrary label. After the images were converted to PPM files, the label was embedded in the file as a comment so that it would be ignored by software used to display the images, but could be used by the system to learn. The images in Figure 1 are examples of both hallways where the images were obtained and
the types of images used by the system. Images 1(a) and 1(b) are taken in the same
hallway looking at the same intersection but from different directions. Due to the similar
locations in the two images, the images and the resulting data are also similar. Since the
pose of a robot is as important as its location, differentiating between image 1(a) and 1(b)
was an important part of this project. Images 1(c) and 1(d) are from a different hall at two
different locations. These images are examples of the two extremes of detail in the
images obtained. Image 1(c) is complex and has many edges while image 1(d) has far
fewer edges.

Figure 1. Examples images from the fourth floor of the S&TII building on the campus of George
Mason University in Fairfax, Virginia. Note the similarity of images 1(a) & 1(b) that depict the same
area from a different direction.
Images of the size we use, though small by some standards, still provide an abundance of information that can be used for localization. In fact, far more information than we are interested in. For this project we wanted to narrow the data to the pixels in the images that carry the most significant information about the image. Our hypothesis is that the most prominent edge pixels are the pixels with the most distinguishing information. To build our database we decided to isolate those pixels using proven techniques.

The first thing we do is to process each image with an edge detection algorithm which assigns an edge magnitude and direction to each pixel. The edge magnitude of a pixel is equivalent to the strength of the edge detected at that pixel while the edge direction is determined by the vector that is normal to the edge detected at that pixel. We use the standard technique documented in [11]. First, we use the partial derivatives, r_x, r_y, g_x, g_y, b_x, b_y for each color band, to produce the matrix S described in Equation (1). The partial derivatives are obtained by applying the Sobel edge detector described in Equation (2) and Equation (3) to the image.

\[
S = \begin{pmatrix}
    r_x^2 + g_x^2 + b_x^2 & r_x r_y + g_x g_y + b_x b_y \\
    r_x r_y + g_x g_y + b_x b_y & r_y^2 + g_y^2 + b_y^2
\end{pmatrix}
\]  

(1)

\[
\frac{\partial}{\partial x} = \frac{1}{8} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad \text{and} \quad \frac{\partial}{\partial y} = \frac{1}{8} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}
\]  

(2) and (3)
We then calculate the two eigenvalues of the matrix. The larger of the eigenvalues, $\tilde{\varepsilon}_1$, corresponds to the edge strength and we take the square root of $\tilde{\varepsilon}_1$ to calculate the edge magnitude. After finding $\tilde{\varepsilon}_1$ we can use its value to find its eigenvector. That eigenvector, $(n_x, n_y)$ is a vector normal to the edge detected at that pixel. After calculating the edge information, we then look for connected edge pixels. We only keep pixels that form connected components containing at least twenty pixels. All isolated pixels are removed. In Figure 2, we display the results of processing the four images from Figure 1. The images in Figure 2 are grey-scale images such that the brightness of the pixels in Figure 2 is set as a direct proportion to the edge magnitude of their corresponding pixels in the images in Figure 1. Therefore, light pixels in the images in Figure 2 correspond to pixels with high edge strengths in images in Figure 1. Any pixels in Figure 1 with an edge strength at or near zero has a brightness of zero in the images in Figure 2.
This leaves edges which are visually recognizable, but not as distinct as possible. In order to further narrow down the pixels to those that most clearly define the objects that define the locations, we use a non-maximum suppression algorithm to narrow the edges to a single pixel width in the edge direction. Finally, we filtered out all the pixels except the ones with the highest edge strengths. At this point, we are left with the pixels we believe are most likely to distinguish the image from an image from a different location. During non-maximum suppression adjacent edge pixels which lie normal to the edge direction are compared and only the pixels with the highest edge strengths are retained. The other pixels are set to zero. This process leads to better defined edges. The results of the non-maximum suppression algorithm are displayed in Figure 3.
After reducing an image to the pixels with the most information about the scene, a large number of pixels still remain. The remaining pixels generally cover a large range of edge strengths. The final step in this section was to implement a threshold for filtering out all pixels except those with the greatest edge strengths. In the first experiment described in the experiments section, we show how and why we decided on a relative threshold of which includes the pixels with the top 2%, 4%, and 6% edge strengths. For most of the images tested, between 10,000 and 15,000 pixels remained after thresholding. In Figure 4, we show the pixels remaining after thresholding of the four images in Figure 1. For
this Figure, the threshold was set to 4%. The images shown are binary images. They only show which pixels are retained. The pseudo-code in Algorithm 1 shows the algorithm we used to select the pixels shown in Figure 4.

Figure 4. Images from Figure 1 after removing all pixels except those with the top 4% of edge strengths.

- For each pixel in the image
  - Calculate the partial derivates in the x and y directions
  - Use the partial derivatives to create the matrix in Equation (1)
  - Calculate the eigenvalues for the matrix
    - Use the eigenvalues to calculate the associated eigenvectors which represent the edge vectors
- Find connected components and filter out isolated edge pixels
- Use non-maximum suppression on edge pixels

Algorithm 1. Calculating edge vectors and filtering
3.2. Intermediate Level Representation

For a couple of reasons, we chose to represent the image data using histograms. The first reason we chose to use histograms was for computational considerations. Since the images we were using were 640x480, or 307200 pixels, we needed to drastically reduce the amount of data to process. As stated previously, filtering out all the pixels except those with the greatest edge strengths reduced the number of pixels to around 12,000 pixels. However, for a project that we envisioned working as a part of system that would need to run in real-time, even that reduction was not enough. For that reason, histograms make perfect sense. By building a histogram based upon the remaining pixels, we further reduce the amount of data for each image to a manageable amount. In our case, our histograms are one-dimensional with thirty-six bins. Clearly, with so few data points, we have more freedom to use algorithms that are computationally complex to compare the histograms.

The second reason for using histograms was that we were able to use them to represent the information that we believed was important. What we wanted was a histogram of those pixels that reflected the visual clues that humans use to navigate our environment. Things such as corners, doors, and bulletin boards, which stay relatively constant, tell us where we are. They also contribute heavily to the edge pixels that remain in each image after processing. More importantly however, most locations have a distinct arrangement of those types of items and therefore of the edge pixels. Each of these different arrangements translates to a different number of edges going in different directions depending on the location of the image. That information, the number of edge
pixels and the edge directions, is used to generate a histogram which represents the image. Those histograms are the information that we use for comparisons in this project.

Before building the histograms, we needed to determine which characteristics of the pixels we would use to build the histograms. There are many different characteristics available to us such as total intensity, intensity in a given color band, saturation, etc. However, we wanted to focus on characteristics that we believed would give us a high level of invariance under a variety of conditions. In particular, we wanted characteristics which would tend to remain the same with different light intensities, different light locations resulting in different glares, and even color changes due to paint or activity on a bulletin board. For our purposes, edge direction fits quite well. The edge direction histograms for the images (a)-(d) in Figure 1 are shown below in Figure 5. Since the example images were taken in man-made environments, each histogram has a peak near 0, 90, 180, 270, and 360 degrees. All of the images we used for this project displayed this behavior. In fact, it is the variations in those peaks that reflect the different objects in the images and contribute to the differentiability of the locations.

![Figure 5](image-url)
Changes in overall lighting intensity have virtually no effect on edge direction. Only at the far ends of the saturation scale is its effect on edge direction detection significant. Of course, saturation changes have an effect on edge magnitude, but we mitigate that effect by considering only the pixels with the greatest edge strength relative to each individual image. This has a normalization effect since the pixels with the greatest edge strength tend to remain the same even if their absolute magnitudes change.

In Figure 6, we show an example of images taken at virtually the same location. The difference between the two images is the lighting conditions. On the right side of the Figure, we show the pixels that remain after processing. While there are visible differences, particularly where the chairs are, the overall resulting sets of pixels are actually quite similar. This leads to similar histograms and the two images would be classified together.
Figure 6. (a) and (b) show the same location under different lighting conditions. (a') and (b') show the pixels remaining after processing (a) and (b).

Other attributes of the image that tend to cause changes in the characteristics of edge pixels involve the position and angle of the camera. We did not want to rely on a camera angle that remained constant. However, we did want to take advantage of the fact that, as a robot is moving down a hallway, it will have a narrow range of poses relative to the hallway. Therefore, in our selection of pixel characteristics we wanted characteristics which would be unaffected by their position in the image. Again, this is the case with edge direction. Unless the significant portions of the image edges go out of the field of view, panning the camera will have little effect on histograms based on edge direction.
We show in Figure 7 the results of one of our tests using images from different angles. Images 7(a) and 7(b) are taken at the same location, but with the camera panned approximately 10°. In this example, the resulting edge direction histograms are strikingly similar. As the camera continues to pan, this similarity degrades rapidly, but for small changes in direction, the choice of histogram variable seems to be invariant.

![Figure 7](image)

Figure 7. (a) and (b) show the same location with the camera at a different angle. (H_a) and (H_b) show the histograms of the edge directions of the pixels remaining after processing.

The other important camera attribute here is its location. As the robot moves down a hallway, the scale of the objects at the far end of the hallway changes significantly. However, if we look at the simple case where there are no foreground
objects, then a change in scale is reflected in the histograms as a similar change in scale. The shapes of the histograms remain similar resulting in a high level of invariance for those pixels. In Figure 8, we show two images that are taken at different distances from a window. This time the histograms are visibly different, however, the differences seem to be mainly in the scale of the peaks.

Figure 8. (a) and (b) show approximately the same location with the camera moved a few feet. (H_a) and (H_b) show the histograms of the edge directions of the pixels remaining after processing.

After choosing potential characteristics, we evaluated several combinations of characteristics involving edge direction, edge strength, intensity, and flow direction. After running experiments comparing histograms based on different combinations, we found
that the best differentiation occurred using histograms based on edge direction and strength. However, instead of including the edge strength directly in the histograms, we used it to filter out all the pixels except those with the greatest edge strength values. The process of building the edge direction algorithms is summarized below in Algorithm 2. First we generate image gradients, and then we sort the gradient magnitudes using a bin sort. For each magnitude within the range represented by a bin, we incremented the count for that bin. Then, starting with the bin for the greatest magnitudes, we total the number of pixels in each bin until the number is 4% of the total number of pixels. Then the gradient magnitude represented by the last bin added to the total is used as the threshold. Using this kind of sort is very fast since it is $O(n)$ and it is appropriate since we do not need all the information provided by a complete sort.

- Use Algorithm 1 to calculate image gradients and gradient magnitudes
- Perform a bin sort on the gradient magnitudes
- Use the bins of the largest gradient magnitudes to calculate the threshold needed to retain approximately 4% of the pixels.
- For each pixel in the image
  - If the edge strength is greater than or equal to the threshold
    - Calculate the angle, in degrees of the edge vector
    - Divide the angle by 36, truncating at the decimal point.
    - Increment the associated bin

**Algorithm 2. Building the histograms**

### 3.3. Comparing Edge Histograms

The next step in the process was to determine the best way to compare the histograms. Most methods involved considering the histograms to be vectors and calculating the distance between the vectors. These formulae were tested for how well they differentiated histograms from similar images with histograms from dissimilar
images. Throughout this paper, we use $d(U, V)$ to refer to a measure between two histograms. In most cases, we specify which formula $d(U, V)$ represents. If we do not say specifically which formula is represented, then the formula is not relevant.

We started our project using a simple Euclidean distance. For two histograms, $U$ and $V$, with 36 bins, the formula we used is described in Equation (4).

$$
\text{dist}(U, V) = \sqrt{\sum_{i=1}^{36} (U_i - V_i)^2}
$$

(4)

The next two distance measures we used were based on a chi-squared formula. The first chi-squared distance formula is shown in Equation (5).

$$
\chi^2_1(U, V) = \sum_{i=1}^{36} \frac{(U_i - V_i)^2}{V_i}
$$

(5)

This formula had a drawback however. For our purposes, two histograms should have the same distance measure no matter which histogram is listed first. In other words our distance measure needed to meet the following requirement: $d(U, V) = d(V, U)$. The second chi-squared formula took care of that problem by including the values from both histograms in the denominator as is shown in Equation (6).

$$
\chi^2_2(U, V) = \sum_{i=1}^{36} \frac{(U_i - V_i)^2}{U_i + V_i}
$$

(6)
The last formula we used was based on the intersection of the two histograms. It should be noted that this measure was not a distance measure by itself. As an intermediate step, it created a temporary histogram that was compared to the database histogram using a distance measure. Comparing each element of the two original histograms as specified in Equation (7) creates the temporary histogram which is subsequently compared to the database histogram using one of the earlier distance functions.

\[ \cap(U,V) = \sum_{i=1}^{36} \min(U_i, V_i) \]  

(7)

We show below, in Table 1, values for each of these formulas using images shown previously. First, edge direction histograms were generated to represent images (a) and (b) from Figure 1 and images (a) and (b) from Figure 6. Then the distances between the histograms in Figure 1 are calculated and the distances between the histograms in Figure 6 are calculated using the same formulae. The images from Figure 1 are of different poses while the images from Figure 6 are similar poses but different lighting conditions. Therefore, we would like a distance measure that gave a much higher value when comparing the images from Figure 1 than when comparing the images from Figure 6. In fact, this is the case for the three distance measures. However, both chi-squared formulas provide a more significant difference between the two sets of images. The intersection formula is actually an inverse distance measure. An intersection value close to the area of the larger histogram indicates similarity.
Table 1. Distance values of different measurement techniques

<table>
<thead>
<tr>
<th></th>
<th>Euclidean</th>
<th>Chi-Squared₁</th>
<th>Chi-Squared₂</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Pair (a) and (b) from Figure 1.</td>
<td>1014</td>
<td>2556</td>
<td>1185</td>
<td>9910</td>
</tr>
<tr>
<td>Image Pair (a) and (b) from Figure 6.</td>
<td>891</td>
<td>870</td>
<td>444</td>
<td>10499</td>
</tr>
</tbody>
</table>

3.4. Learning Phase - LVQ

At this point the problem has essentially become one of pattern recognition. Significant research has already been done in this field and we decided to use an existing package for this step. Learning vector quantization (LVQ) is an established method of learning from data that is attractive due to its simplicity, effectiveness, and the fact that there is an existing implementation that meets our needs. The implementation that we used for this project was the LVQ_PAK package produced by the Laboratory of Computer and Information Science at the Helsinki University of Technology [12]. Source code is provided with the package and we made one modification to the system. The original package based its distance calculations on the Euclidean distance as described earlier. We replaced that algorithm with an algorithm based on the second chi-squared algorithm, also described previously.

The LVQ methodology examines the data and builds a set of vectors that represent different regions in the n-dimensional data space. Initially, the algorithm chooses a set of prototype vectors to represent the data. The prototype vectors are chosen such that a given prototype vector is based only on vectors with the same label. For this
project, the data was labeled in a previous step. That label is used as the label of the prototype vectors. After the initial prototype vectors are created, they are refined using one of the heuristic approaches proposed by Kohonen [13]. The refinement algorithm attempts to minimize distances between the original data vectors and the prototype vectors by removing vectors from over-represented groups and adding vectors to groups that are under-represented. This process involves three user-set parameters. The first parameter, and in this experiment, the most important, is the maximum number of prototype vectors. Our system produced much more reliable results with a large number of prototype vectors. When the number of prototype vectors was small, the ability of the system to correctly label a test vector diminished rapidly. The second is the number of initial prototype vectors to produce. This parameter did not seem to make a big difference as long as the third parameter was set sufficiently high. The third parameter controlled the number of iterations of the refinement algorithm. As long as the refinement algorithm had enough time to make up for a small number of initial prototypes, the resulting prototype vectors were always the same.

Since we used pre-packaged software, as shown in Algorithm 3, this step was straight-forward. Our work here came down to finding the optimal parameters to send to the LVQ routines and replacing the distance functions.

- For each image, generate the histogram using Algorithm 2
- Create a text file containing the histograms as vectors
- Use LVQ routines to generate the code vectors

**Algorithm 3. Database generation**

As part of the analysis of this part of the project, we used the Sammon function included in LVQ_PAK. The Sammon function attempts to map high-dimensional data
points to a two-dimensional image while preserving the relative distances between the points. In Figure 9, as an example, we show a simple Sammon projection. To generate this simple example, we generated the code vectors for just two of the classes of vectors. Then we ran the Sammon routine on those code vectors. The image shows two distinct clusters of data points. This is exactly the result that we were looking for in that there is a clear separation between both groups of code vectors. In this example, images classified using these code vectors would most likely have a high degree of success.

Figure 9 - Thirty images used to produce the example Sammon projection shown in Figure 10
Figure 10. Simple Sammon projection
3.5. Recognition Phase

After generating representative histograms to represent clusters of data, we are able to classify new images using the representative histograms. Classification of a new image is accomplished by processing it in the same way as the original images during the learning phase. The edge direction and strength are calculated for each pixel, and then those values are used to generate the same type of histogram as was used to build the current test database. After the histogram is generated for the new image it is compared to each of the vectors generated by LVQ using the same distance measurement used during the learning phase. The new image is then classified as belonging to the same group as the vector to which it is closest.

During the search for the closest vector, the algorithm also keeps track of the second closest vector with a different classification. At the end of the classification, the algorithm generates a confidence level based on the distances between the new image and the two closest code vectors. The distance of the second closest vector is divided by the distance of the closest vector and then the result is arbitrarily multiplied by 100. This calculation attempts to quantify the reliability of the choice. If the confidence level is very close to 100, say 100-150, then the two distances are close enough that the classification is not reliable. In our experience, confidence levels greater than 160 tended to be quite accurate. We summarize this part of the system below in Algorithm 4.
• Generate the histogram for the new image using Algorithm 2
• Set MinDistVal1 = MinDistVal2 = Some Large Number
• For each information vector in the database
  o Calculate the distance between the histogram of the new image and the current information vector
  o If the current distance is less than MinDistVal1
    • If the current information vector is a different class than MinDistVal1 Then MinDistVal2 = MinDistVal1
    • MinDistVal1 = current distance
  o Else if the current distance is less than MinDistVal2 AND the current information vector is a different class than MinDistVal1 Then MinDistVal2 = current distance
• confidenceLevel = MinDistVal2 / MinDistVal1
• If confidenceLevel < confidenceThreshold
  o Perform sub-image comparison using the new image, and the images closest to the information vectors of MinDistVal1 and MinDistVal2
• Return the class specified by MinDistVal1

Algorithm 4. Image classification

The final step we took in this project was to attempt to take advantage of the information contained in the confidence level, and improve the results of the overall system. When the confidence level for a given classification was below a given threshold, we decided to refine the classification by comparing sub-images of the new image and the images in the database closest to the vectors in the database. An example where this improved our results is shown in table 2 using the image in Figure 11.

![Figure 11. Example of an image misclassified, then reclassified correctly](image-url)
Image 11(a) was compared to image 11(b) and image 11(c). In this case, image 11(a) should have been classified with image 11(b), nevertheless the similarity value led it to be classified with image 11(c). However the confidence level was relatively low. Therefore, the sub-image comparison was used. As show in the table, when the sub-images are compared, the differences are much higher as is the confidence level. Because of that, the image is re-classified with image 11(b). The sub-images we used were generated by first dividing the images into quarters with each quarter defining a sub-image. We also used a fifth sub-image defined by an area the same size as the other sub-images, but centered in the image.

Histograms for sub-images are generated using the methods already described, except that only the pixels from the sub-image section of the image are used. Then the distances between those histograms are calculated using the same methods as described for histograms of the complete images. The new image is then re-classified based on the median of those distances. We experimented with two different methods of re-classifying images using sub-image histogram distance. In one experiment we simply totaled the distance values for each sub-image. In the second experiment we used the values of the median sub-image distance to make the decision. There was a slight improvement by using the median distance value as opposed to using the total distance. The overall classification algorithm we used is summarized in Algorithm 5.

The first row of Table 2 displays the initial comparison values that support a classifying image 11(a) with image 11(c), but with a low confidence level. The next five rows show the sub-image comparison values. The final two rows show the two methods for testing the sub-image values. Both sub-image comparison methods support
classifying image 11(a) with image 11(b), the correct choice, and with a much better confidence level than the complete picture confidence level.

Table 2. The Similar Location column shows the similarity values when comparing Image 11(a) with Image 11(b). The Different Location column shows the similarity values when comparing Image 11(a) with Image 11(c). The Confidence Level column shows the ratio of the similarity values multiplied by 100.

<table>
<thead>
<tr>
<th></th>
<th>Similar Location: Image 10(a) with Image 10(b)</th>
<th>Different Location: Image 10(a) with Image 10(c)</th>
<th>Confidence Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete Picture</td>
<td>531</td>
<td>352</td>
<td>150</td>
</tr>
<tr>
<td>Similarity Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper Left</td>
<td>503</td>
<td>1591</td>
<td>N/A</td>
</tr>
<tr>
<td>Upper Right</td>
<td>319</td>
<td>971</td>
<td>N/A</td>
</tr>
<tr>
<td>Lower Left</td>
<td>574</td>
<td>1213</td>
<td>N/A</td>
</tr>
<tr>
<td>Lower Right</td>
<td>333</td>
<td>633</td>
<td>N/A</td>
</tr>
<tr>
<td>Center</td>
<td>285</td>
<td>384</td>
<td>N/A</td>
</tr>
<tr>
<td>Median Sub-image</td>
<td>333</td>
<td>971</td>
<td>292</td>
</tr>
<tr>
<td>Similarity Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Sub-image</td>
<td>1729</td>
<td>4408</td>
<td>255</td>
</tr>
<tr>
<td>Similarity Value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. Experiments

In this chapter, we describe the experiments we used to verify our choices and to test the system. The first experiment described below details how we test the complete system and gives the results. After that we show the effects of including sub-image comparisons in the system. The third experiment we describe discusses the methods we used to set edge strength thresholds. Then we show the results of an experiment in which we tested different histogram comparison types with different histograms. After choosing the histogram type and histogram comparison type we further examined the differentiation capabilities of our choices by displaying the results of comparisons in a similarity matrix and a pair-wise over time graph. Finally, we look at the statistics of the code vectors generated by the LVQ algorithms. Figures 12-14 show most of the images used for all these experiments.

4.1. Results of Learning and Classification Experiments

At this point in the project we were ready to analyze the system as a whole. To accomplish that we wanted to include every function that would be involved in a deployed version of the system. Therefore, for this experiment, we produced a program that would call each sub-system in order with each successive call relying on the results of the previous calls. We started with 185 images obtained using the method described earlier. The images were taken in hallways that we divided into seven regions. We then
manually classified these images according to the region in which they were taken and annotated the image files so that they contained a reference to their classification that could be read by our system.

After getting the images ready, we ran the experiment. The first thing our testing program did was to generate histograms for each image and store the histogram in a text file for later use. When the histograms were complete, the testing program randomly selected approximately five percent of the histograms to act as unclassified images. The remaining histograms were used to generate a text file that was processed by the LVQ software resulting in the code vectors used for classification. Finally, the five percent of the histograms that were selected in the randomization step were classified using the methods described in section 2.5. For this experiment, the classification was done without using sub-image comparisons. After a histogram was classified, the classification was compared to the classification assigned to it manually. For each iteration of this test, the percentage of correctly classified histograms was recorded in a file. Each set of parameters was tested this way by iterating through the test 100 times, giving us 100 accuracy percentages.

We show in Table 3, the average accuracy for the test using each six combinations of parameters. Throughout this project, there have been parameter values that we tested and then subsequently used the values that produced the best results. Even though certain values seemed to work best at those times, as the project evolved we recognized that the previous parameter values need to be reevaluated in case their effectiveness had changed. For our first complete test of the system, we decided to include the possibility that normalization of the histograms might prove to be effective. We also wanted to test the effectiveness of different edge strength thresholds.
As predicted by earlier experiments, the results were best when the top 4% of pixels were used to generate the histograms. We also found that non-normalized histograms produced better results. This would seem to be due to the fact that the number of pixels used to generate the histograms is already nearly the same in each image because of the thresholding.

Table 3. 100 iterations of the classification test with different parameters

<table>
<thead>
<tr>
<th></th>
<th>6%</th>
<th>4%</th>
<th>2%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Normalized</td>
<td>91.12</td>
<td>92.35</td>
<td>78.21</td>
</tr>
<tr>
<td>Normalized</td>
<td>89.68</td>
<td>90.34</td>
<td>83.03</td>
</tr>
</tbody>
</table>

4.2. Results of Learning and Classification Experiments Using Sub-Images

The final experiment we performed for this project was an attempt to differentiate similar images of different locations. This involved comparing sub-images of the images being compared. We used the same techniques here as in section 3.6. Each combination of parameters was again tested 100 times, and the values shown are the average accuracy for that pair. Other than comparing sub-images, we used the same parameters as the experiment in section 3.6. As seen in Table 4, using sub-images produced a modest but significant increase in the reliability of the classification for each version of the test.
Table 4. 100 iterations of the classification test using sub-images

<table>
<thead>
<tr>
<th></th>
<th>6%</th>
<th>4%</th>
<th>2%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Normalized</td>
<td>92.37</td>
<td>95.40</td>
<td>81.39</td>
</tr>
<tr>
<td>Normalized</td>
<td>91.84</td>
<td>94.41</td>
<td>84.70</td>
</tr>
</tbody>
</table>

4.3 Different Edge Strength Thresholds

Our first experiment involved deciding on which pixels we wanted to base our future experiments. We needed to determine a range of pixels that reflected our hypothesis that the most prominent edge pixels contained the distinguishing information we wanted to use. To do this, we generated binary images similar to those in Figure 4. These images were generated by first calculating the edge strength of each pixel using the previously described technique. Next, the locations of the pixels with edge strengths equal to or above a threshold were saved in a file. We used Matlab to display those pixels in a 640x480 grid - effectively displaying a binary image of the most prominent edge pixels. Using this method, we were able to display and evaluate several different threshold values including both absolute thresholds and relative thresholds.

Thresholds were initially chosen by examining histograms of the edge strengths such as the histogram in Figure 15. The curve shown in the figure is very similar to the histograms of the other images in this project. That histogram uses the edge strength of pixels to populate the bins. By choosing a threshold on the curve just before the curve becomes flat, we are able to include only those pixels with the greatest edge strengths without including pixels from lighting gradations or noise. Based on those histograms and the results shown in Figure 16, we decided that we wanted roughly 10000 to 15000
pixels. For a 640x480 image, that number of pixels is approximately 4%. Therefore, we used 2%, 4%, and 6% for most of our experiments.

In Figure 16, image 16(a) is generated by retaining the pixels with the top 10% of edge strengths while the image 16(b) is built by using 96% as the threshold. In the image on the left, it is easy to see the additional features that might be desired such as the chairs which don’t show up well in the image on the right. Yet, you can also clearly see that there is much more noise in the image on the left. We used on 96% because it seemed to most closely reflect our goals, however, we also recognized that there was no material to indicate that one choice was inherently better than the other, we routinely used 94%, 96%, and 98% in our experiments.

Figure 15 - Histogram of gradient magnitudes in image 6(a) using. The thin blue vertical line represents the threshold. All pixels to the left of the line are dropped. Pixels with gradient magnitudes below 100 are not included in the histogram.
4.4. Distinguishing Abilities of Different Histogram Types

There were a number of different types of histograms that we thought might be capable of helping to distinguish images of different locations. Based on the reasons discussed in the methodology section, we choose histograms using edge direction. For this experiment, we also used histograms using edge direction combined with edge strength and a third histogram using edge direction combined with intensity. To conduct this experiment we compared the histogram of a given image to the histograms of other images in the same area as well as images from other locations. We were looking for combinations of histogram types and comparison types that showed a similarity between the first histogram and the histograms of images from the same location and dissimilarity between the first histogram and the histograms of images from different locations.

The results for one of these experiments are shown in Table 5. In this particular experiment, we compared histograms using an absolute edge strength threshold that used approximately 10000 pixels from each image to build its histogram. We built three types of histograms. The first two histograms are a one-dimensional histogram based on the
edge orientation of the pixels a two-dimensional histogram using edge strength and orientation of the pixels. The third histogram was based on consecutive image flow calculations which we abandoned early, but include in this table because we included it in this experiment.

Both previously stated goals of correctly detecting similarity and dissimilarity appear in the results shown in the Table 5. Images near the bottom of the list have low similarity values, indicating that the images are close to the original image. Images near the top of the list, particularly the top four, have very high similarity values, indicating images that are not close to the original image. This is the effect we were looking for since the images taken in a location similar to the main image are at the bottom of the list and as you go up the list, the images are farther from the main image location and the four images at the top of the list are in completely different locations.

There was not a large difference in results between the histogram types, however, in general, the histograms using only the edge direction produced slightly better results. For comparison type, the differences were more pronounced with the comparisons using the second chi-squared formula giving the best results. Based on these results, we were also able to choose to use the histograms based on edge direction and the second chi-squared distance measure.

<table>
<thead>
<tr>
<th>Image</th>
<th>Threshold</th>
<th>Orientation X²</th>
<th>Orientation X²</th>
<th>Orientation Inter.</th>
<th>EdgeStrength-Orientation X²</th>
<th>EdgeStrength-Orientation X²</th>
<th>EdgeStrength-Orientation I/U</th>
<th>Flow X²</th>
<th>Flow X²</th>
<th>Flow Inter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST2MainHall092.ppm</td>
<td>300</td>
<td>90147, 8994</td>
<td>640</td>
<td>58588, 7576</td>
<td>2354</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST2MainHall089.ppm</td>
<td>300</td>
<td>62955, 7398</td>
<td>702</td>
<td>45624, 6431</td>
<td>2674</td>
<td>293, 344</td>
<td>10871</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST2MainHall062.ppm</td>
<td>300</td>
<td>14262, 2663</td>
<td>1054</td>
<td>8485, 2778</td>
<td>3379</td>
<td>343, 484</td>
<td>8821</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST2MainHall059.ppm</td>
<td>300</td>
<td>9714, 1863</td>
<td>1139</td>
<td>7501, 2478</td>
<td>4025</td>
<td>349, 385</td>
<td>9337</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.5. Similarity Matrix Showing Clusters

Another approach we used to analyze the effectiveness of different types of histograms was to generate a three dimensional graph showing the distance values between each histogram. As in the previous experiment, this experiment also shows the ability of a graph type and comparison method to differentiate between locations. To generate the graphs, a histogram for each image is generated and then compared against all other image histograms. The distance value is then plotted as the height of a 3-d graph with a square grid as its base. An example of this graph is shown below in Figure 17.

Peaks in the graph represent pairs of images that are dissimilar. Conversely, low areas in the graph represent pairs of images that are very similar. In performing this experiment we were looking for clusters. Specifically, it was our belief that there would be a cluster corresponding with each location. In fact, upon examining the graphs displayed the clustering that was expected. This result confirmed the fact that our techniques had the ability to differentiate between images from different locations.
4.6. Pair-wise Over Time Graph

The final experiment we performed as a test of our choice of histogram type and comparison algorithm was to generate a graph showing pair-wise distances over time. To produce this graph, we generated the histograms for each image in the database and then ordered them in a time sequence as though they were taken by a robot navigating the hallways. After that we ran the comparison algorithm for all adjacent pairs of histograms. In the graph shown below in Figure 18, the values displayed represent the similarity measure of a histogram compared to its immediate neighbor. Again, large values represent significantly different images. The spikes seen in the graph correlate quite well with dramatic changes in the location of the robot. Dramatic changes are those that involve turning to face a different hallway or turning around. This graph is another
confirmation of the differentiating capabilities of the system. We divided our environment into seven regions that we believed were separate locations and would be differentiable. Visual inspection of the graph shows the seven different locations as valleys between peaks or at the beginning or end of the graph.

![Graph showing pair-wise comparisons over time](image)

**Figure 18. Pair-wise comparisons over time**

### 4.7. Code Files after Teaching

A measure of the success of a given teaching method was found in the analysis of the code files produced by the LVQ software. After LVQ produces the set of code vectors that represent each location we generated statistics based on those code vectors. We calculated the distance between each pair of vectors in each class. We also
calculated the mean distance (D) and the standard deviation ($\sigma^2$). We believed that classes that were represented by a group of code vectors that were close to each other would have a greater degree of success when used to classify new images. In table 6 we show a representative example of the distances statistics that were typical for the code vectors generated. By examining the values in the table, you can see that the code vectors for the first four classes are compact – the mean distance between vectors is low. However, the code vectors for the last three classes are much more spread out. This correlates with the pair-wise comparison graph in the previous section, where the intra-class distances are small for the first four classes, but in the last three classes, even though the boundaries are still distinct, the pair-wise distances are much larger. The statistics in table 6 also agree with the two dimensional representation of the code vectors generated by the Sammon algorithm. The results of the Sammon projection for this experiment are shown in Figure 19. In that representation there are clear groupings for the first four classes (A,B,C,D). In contrast, the other three classes (E,F,G) have areas of concentration, but also have several points that are scattered around the chart.

Note that each point is labeled with a letter and a number. The number is used to identify the code vector that correlates with that point. The code vectors used to generate this projection are available in the appendix of this document. By finding the code vector in the list with a class and number matching a point in the projection, you can see which vector matches that point.
<table>
<thead>
<tr>
<th>Class</th>
<th>D</th>
<th>$\delta^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>538</td>
<td>391</td>
</tr>
<tr>
<td>B</td>
<td>243</td>
<td>169</td>
</tr>
<tr>
<td>C</td>
<td>413</td>
<td>600</td>
</tr>
<tr>
<td>D</td>
<td>233</td>
<td>139</td>
</tr>
<tr>
<td>E</td>
<td>1494</td>
<td>1499</td>
</tr>
<tr>
<td>F</td>
<td>1479</td>
<td>964</td>
</tr>
<tr>
<td>G</td>
<td>1263</td>
<td>1469</td>
</tr>
</tbody>
</table>
Figure 19. Sammon projection of all code vectors
5. Conclusion

Navigation of mobile robots in man-made environments continues to be a significant problem with several sub-problems still unsolved. One of these sub-problems is the localization of mobile robot. Sometimes a robot may not have any information on its whereabouts other than its own internal map. Also, while navigating hallways, the robot will frequently need to use external stimuli to verify its location. Both of those problems provide the impetus to solve the localization sub-problem.

In this paper we describe our work towards the localization of mobile robots using images. Our technique involves building a database of information vectors which represent the different locations in the robot’s environment. The information vectors were histograms based on the edge direction and strength of the edges in the images of the environment. In a two phase process, the first phase consisted of collecting images of the environment to build the database. In the second phase, new images were obtained to simulate a robot traversing the environment and attempting to localize itself. All of the images were processed using the same edge detection algorithm and then refined into histograms using the same parameters. After the database was created using the initial images, the second images were compared to the information vectors and classified based on a closest neighbor algorithm. Using this technique, we were able to achieve an overall accuracy rate of 95%.
Each of the sections of this project presents a number of opportunities for future research. In the representation phase, we chose edge direction as basis for our histograms. Edge direction seems to produce excellent results, however, there are other variables that may contribute independent information and could be combined with edge direction to produce multi-dimensional histograms.

After the histograms are built, the distance measure used to compare them is also an area where additional research should take place. There has been interesting work done on histograms unfolding to compare histograms. For this project we looked at unfolding, but due to time constraints were unable to complete a satisfactory test. In addition to unfolding there are other comparison techniques that should probably be explored such as the Earth Mover’s Distance.

When classifying histograms, we chose to classify an image based on the nearest neighbor. However, alternatives exist such as the k-nearest neighbors algorithm. In that algorithm, the k-nearest neighbors are calculated and then the class which has the most representatives in the list of neighbors is used to classify the input image. Looking at the Sammon projection, you can find arguments which both support and contradict the intentions of the algorithm. Areas in the projection, with different classes represented, might benefit from a voting-like classification scheme. On the other, if a single vector represents a distinct type of image of a given class, but has a couple of vectors from a different class nearby, then the k-nearest neighbors algorithm could cause problems.

There also may be other ways to implement the sub-image comparisons. Perhaps using the standard deviation of the distances or some other way of prioritizing the differences in the sub-images would improve the results here.
Ultimately, we would like this work to lead to a navigation system using graph techniques to store a representation of a physical map. In that system, we envision the locations recognized by the system corresponding to the nodes of the graph. This would provide a navigation system in which the robot would use the map for movement planning and the localization system described in this paper to verify location in the graph.
References
6. References


CURRICULUM VITAE

Philip M. Barber was born on January 31, 1967, in Wellington, Kansas, and is an American citizen. He graduated from Herndon High School, Herndon, Virginia, in 1985. He received his Bachelor of Sciences from George Mason University in 1991. He has been employed as a computer analyst in Northern Virginia for eleven years.