# LOCATION RECOGNITION USING A VERY HIGH DIMENSIONAL FEATURE SPACE

By

Christopher Costello

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

December, 2011

Nashville, Tennessee

Approved:

Dr. Mitch Wilkes

Dr. Kazuhiko Kawamura

Dr. Richard Alan Peters II

Dr. Benoit Dawant

Dr. Nilanjan Sarkar

# ACKNOWLEDGMENTS

I would like to thank my parents Dr. Joseph P. Costello III and Joan Costello for pushing me to always excel at what I enjoy. I would also like to thank my wife, Lindsey Costello, for always being there for me and encouraging me through each step in the process graduating. Without them this process would have been much more difficult.

Next, I would like to thank Dr. Mitch Wilkes for supporting, guiding, and encouraging me through this work. His ideas were invaluable in the completing this system, and his support and encouragement helped to motivate me through this work.

I would like to thank Dr. Alan Peters for being patient with me and helping me whenever I asked for it. Also, for the knowledge he passed to me in image processing and computer vision. This information will be very helpful in my future works.

I would also like to thank Dr. Kawamura, Dr. Dawant, and Dr. Sarkar for taking there time to help and guide me in my research.

I give special thanks to Jonathan Hunter for helping me with this system when the others had graduated. His help as a resource with this system has been and continues to be priceless.

Finally, I thank Flo for always being there to make the good days better and the bad days good. She will always be appreciated and greatly missed.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

# CHAPTER I

# INTRODUCTION

An understanding of "where we are" (spatial cognition) is a fundamental ability for humans. How people determine their location changes as they get older [39]. The experiments performed by Cornell et al. [40,41] show that as children age they are able to use more stable distant landmarks (building views) while younger children tend to use nearby simple landmarks (e.g., a fire hydrant). This gave the older children a better understanding of the world and made for a more robust means of navigation. Unfortunately, localization is not as simple as recognizing an unchanging environment. As time passes the features of objects tend to change. A simple example would be a room or hallway that is frequently used being painted a different color from time to time. This type of change is quickly recognized by humans and accounted for. There may be a moment of confusion, but it is quickly worked through. Another aspect of a human's spatial cognition is the ability to classify rooms. Although initially the label of a kitchen is given to a child, people are easily able to quickly and correctly classify a different kitchen in a new environment. This ability also allows people to provide meaningful location information to one another without the use of exact geometric measurements relative to the world.

In order for a robot to achieve this type of spatial cognition, the system will need to be created from a bottom up design similar to human development. Humans start as infants that learn to recognize objects. Then as they grow they become mobile. This mobility is what changes their view from simple scene understanding (what's around me

now) to a more contextual understanding (I'm in a kitchen). Also, because they are mobile their world expands to include many more objects for segmentation and locations. It is with this developmental paradigm in mind that this work is being proposed.

There have been numerous robotic systems that have attempted to solve the localization paradigm [6,7,13,21,13,21,26,32,35]. Many have done so for the purpose of navigation [6,7,13,32,35]. Although navigating through a local environment is very important in mobile robotics, it is not the only step to spatial cognition. These systems still need the ability to reliably recognize where they are in a larger contextual sense. A common problem to look at in the area of localization is referred to as Simultaneous Localization and Mapping (SLAM) [6]. The algorithms created to solve this problem are referred to as SLAM systems. The objective of a SLAM system is to generate a map of the area while still localizing the robot within that area. Two other means of localizing robots are through landmark detection or template matching. Landmark detection aims to robustly extract some features out of an image and use those features to determine where the robot is [13,35]. The final method of localizing a robot, template matching, attempts to use the information in an entire image [21,26]. All three principles have been successful to various degrees.

SLAM, at its roots, is based off of using a laser range finder or vision to map a new area and localize the robot within that area. It has been shown to be extremely useful in the area of navigation [6,7]. The glaring weakness of using only a range finder with SLAM is that without any visual appearance information, it is not capable of determining a difference between two geometrically identical locations. An example of this would be two floors of the same building. It is not inconceivable to think that a building would

have multiple floors with the same geometric features. So even though SLAM will be able to determine exactly where the robot is on that floor, it has no means of determining which floor it is on. In terms of spatial cognition this is very limiting. Because of this, a number of systems have attempted to use visual SLAM [13,18,19]. This approach to the SLAM problem typically combines both a range finding sensor with a camera. The range finder will generate the maps while the camera is used for landmark detection to supply more information about an area [13,18,19].

Landmark detection alone has also been used extensively for navigation [13,18,19,35]. The idea behind landmark detection is to find unique features in the environment and use them to localize the robot. These features can include artificial landmarks in an environment [35] or natural features [13]. Once the robot understands where it is, it can plan its path for navigation. However this type of localization is also limited for multiple reasons. The first reason is the sensitivity of finding the landmarks. Extracting exact information from a scene can be very difficult. If the landmarks don't appear exactly as expected, the system may need a great deal of robustness in order to deal with the changes. The second limitation on landmark detection is the dependency on the landmarks. Because the robot has no other means of interpreting its environment, missing the landmarks can render the robot lost. This dependency combined with the sensitivity may limit the robustness of this technique. This is why the combination of landmark detection with SLAM has helped with navigation.

At this point, it should be noted that the researchers consider these systems to be very good at navigation. Replacement of these techniques is not the intention of this work. The work presented is looking to add a new dimension to the way robots are able

to perceive the overall context of the environment. The addition of SLAM for the purpose of navigation should be considered once the full potential of the proposed system has been explored.

There has also been some work in template matching or contextual based localization done using epitomes [21]. The epitome is created based on the probabilistic information in the training images, and is used to compare the current image to a known location. However the criticism of this work is the same as that of the previously mentioned systems, and that is that they all limit the information used. Although it is suggested that the epitomes will be able to differentiate between "my kitchen and "a kitchen", there is no segmentation of individual percepts performed. This means that they are essentially looking for a measure of difference. Although this is acceptable for location recognition that is all this system will be able to accomplish. Once an epitome is created all of the other information about the individual percepts in the original image is lost. Because of this weakness, it seems that the epitomes would fail to recognize an area if a change occurred, such as painting the walls.

## Objective of Visual System

The goal of this work is to develop a system that can semi-autonomously (with minimal human interaction) generate a model of the world around it, and use this model to begin to understand the context of the world. The visual system these functions will be added to, has been previously used for studying the effectiveness of a Working Memory application [1,2], as well as for the analysis of human motion segmentation [3]. The system uses a sparse representation of a very high dimensional feature space of the hue,

saturation, and value (HSV) domain to maximize distance between the percept's colors. Because of the time requirements in processing each image, the segmentation of the image has been re-implemented using Nvidia's CUDA architecture for highly parallel computation.

The system that has been built has demonstrated the ability to: learn its own percepts to describe a larger environment, segment a large area into smaller more meaningful areas, segment out the global percepts from all areas, generate local models of each area based on the presence of the global percepts found, and update the models with more training as needed. The models of the world have successfully been used to determine the local area (or context) where images have been taken at different times. In addition to the location recognition function, the system is also capable of detecting reflections caused by distant light sources based on the behavior of segmented reflection as the robot moves, further classifying the percept blobs based on their behavior from one image to the next, locating novel objects, and recognizing novel areas. It is also worth noting that another goal of this system is to use as small amount of complexity while retaining the most information possible. The introduction of unnecessary complexity tends to lead to instability and fragility. Thus far all of this has been accomplished by only using a very high dimensional HSV-based color histogram domain.

## Organization of Paper

The organization of this paper will provide a description of three types of location recognition systems and a review of the previous uses of the visual system used as Chapter II and Chapter III respectively, then three papers organized for journal

submission will be provided. The first paper, Chapter IV, will cover the location recognition process while the second paper, Chapter V, covers the reflection detection, percept classification, and novelty detection. Finally the third paper, Chapter VI, will provide an overview of the very high dimensional feature space and what has been learned about it through multiple works [1,2,3,4,5]. Then Chapter VII will provide a final review of the work performed and final thoughts about the work.

# CHAPTER II

# LOCALIZATION TECHNIQUES

## Simultaneous Localization and Mapping

SLAM is a very common tool used for navigation in mobile robotics. The strength of the system is its ability to map out a new environment while continually localizing itself within that environment. This has been shown in many studies [6,13,17,18]. The most basic implementation of SLAM uses a laser range finder mounted on the mobile robot to detect all the edges nearby in order to create an extremely accurate map of the area and localize the robot with in that area. As mentioned, SLAM has been used by many researchers and so a complete background of all implementations would be impossible. Therefore the work of Thrun et al. [6], will be used to describe an example of a SLAM architecture using a range finder.

Thrun et al. [6] incorporated SLAM on the mobile robot Groundhog shown in Figure 2-1.


Figure 2-1: The Groundhog robot is a 1,500 pound custom-built vehicle equipped with onboard computing, laser range sensing, gas and sinkage sensors, and video recording equipment. [6]

Groundhog's objective is to explore coal mines that are too dangerous for humans to enter. Because Groundhog will be underground, the simplest method of localization, GPS, is not available.

The first level of processing that SLAM must go through is aligning identified points in consecutive scans acquired from the laser range finder [12,44]. It then minimizes the quadratic distance between each of the relative points in order to calculate the relative displacements and orientations [10]. This allows Groundhog to obtain two measurements: locally consistent maps and an estimate of Groundhog's motion. Because of the accumulation of the error in the scan matching, it is not possible to gain global consistency from the local maps [9,12,45]. In order to deal with the problem the system makes use of a Markov random field (MRF) [6]. They start by creating local maps of the area in five meter intervals. An example of a local map is shown in Figure 2-2.

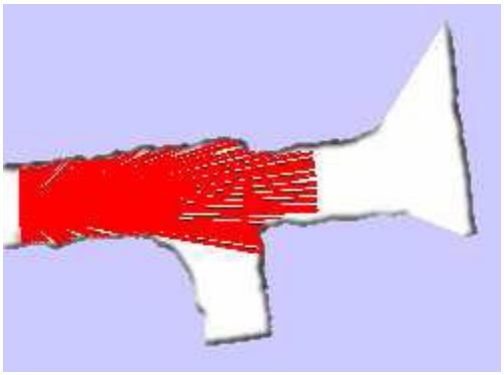


Figure 2-2. An example of a local map and 2d range scan [6]

The absolute locations and the orientation of the k-th map will be given by:

$$\xi_k = (x_k\, y_k\, \theta_k)^T \tag{2.1}$$

Where x and y represent the Cartesian coordinates and θ is the orientation. The set of coordinates for all local maps will be represented as $\chi = \{\xi_1, \xi_2, \ldots\}$.From scan matching, the relative displacement is shown as:

$$\delta_{k,k-1} = (\Delta x_{k,k-1}\ \Delta y_{k,k-1}\ \Delta\theta_{k,k-1})^T \quad (2.2)$$

where each delta value measures the relative displacement along its corresponding axis. If scan matching could be performed without errors, it would be possible to create a global consistency map through the following recursion (with the boundary condition $\xi_0 = (0\ 0\ 0)^T$):

$$\xi_k = f(\xi_{k-1}, \delta_{k,k-1}) = \begin{pmatrix} x_{k-1} + \Delta x_{k,k-1}\cos\theta_{k,k-1} + \Delta y_{k,k-1}\sin\theta_{k-1} \\ y_{k-1} - \Delta x_{k,k-1}\sin\theta_{k,k-1} + \Delta y_{k,k-1}\cos\theta_{k-1} \\ \theta_{k-1} + \Delta\theta_{k,k-1} \end{pmatrix} \quad (2.3)$$

However, since errors are inevitable, the recursive approach is generalized into a soft sequence of constraints that induces a Gaussian probability distribution over $\xi_k$ with a covariance $\sum$: $f(\xi_{k-1}, \delta_{k,k-1})$

$$\phi(\xi_k, \xi_{k-1}) = |\pi\Sigma|^{-\frac{1}{2}} \exp -\frac{1}{2}\left(\xi_k - f(\xi_{k-1}, \delta_{k,k-1})\right)^T \Sigma^{-1}\left(\xi_k - f(\xi_{k-1}, \delta_{k,k-1})\right) \quad (2.4)$$

The function $\phi$ is referred to as a potential and is used to link the consecutive maps together in a soft way [6]. Creating a global map is now equivalent to finding the sequence of map coordinates $\chi$ that minimizes the product of potentials $\prod_k \phi(\xi_k, \xi_{k-1})$. The MRF for the Bruceton Research Mine is shown in Figure 2-3.

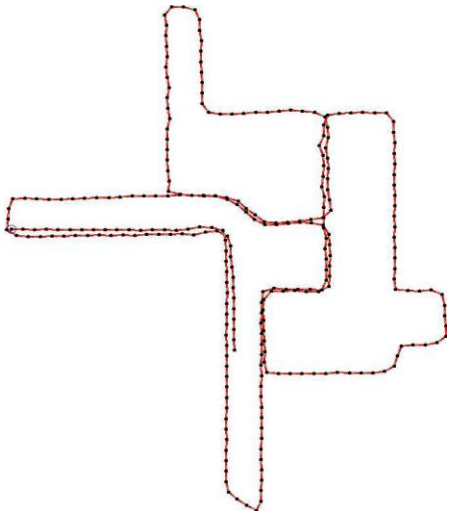


Figure 2-3: The resulting Markov random field: Each node is the center of a local map, acquired when traversing the Bruceton Research Mine near Pittsburgh, PA. [6]

An advantage of MRF is that it models the uncertainty of the local scan matching's. This allows the map to be altered with respect to the global consistency constraints [6]. Therefore if there exists a k-th map that overlaps a map j < k-1, observed in a previous run, and a potential between maps $\xi_k$ and $\xi_j$, defined as $\phi(\xi_k, \xi_j)$, is obtained, the new potential can be added to the set of potentials $\Phi = \{\phi(\xi_k, \xi_j)\}$ and softly enforce the displacement between $\xi_k$ and $\xi_j$. The resulting MRF will then be described as:

$$p(\chi) \propto \prod_{k,j} \exp -\frac{1}{2} \left(\xi_k - f(\xi_j, \delta_{k,j})\right)^T \Sigma^{-1} \left(\xi_k - f(\xi_j, \delta_{k,j})\right) \qquad (2.5)$$

Equation 2.5 can be thought of as the non-normalized probability over the joint global locations of all submaps, and the global map can now be created by minimizing this function over the locations $\chi$ of all submaps [6].

The final step in building a consistent map is defining the consistency constraints. This is more commonly called the data association problem [8,46]. This problem refers to the robot's ability to decide if two different measurements correspond to the same

object. The approach used does a lazy search through the data association tree while performing likelihood maximization [6]. The data association tree represents all the discrete data association decisions made during the global map building. An example tree is shown in Figure 2-4.

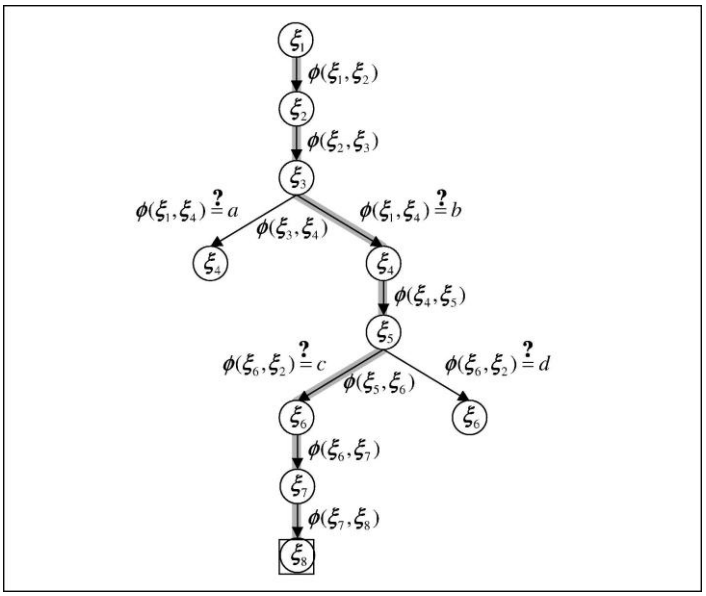


Figure 2-4. Shows the tree and a path chosen by locally determining the most likely data association. [6]

As new local maps are added to the data association map, the decision of whether a consistency constraint is needed is made. If a consistency constraint is needed then the value for the constraint must also be found. In this system's implementation a new consistency constraint is created if the current map overlaps a previously found local map with a sufficient probability. The next question that must be addressed is the orientation of the new local map with respect to the global map. It is at this juncture where the maximum log-likelihood function is used to decide a branch in a tree. In Figure 2-4 this can be seen at node $\xi_4$. In this case, the constraint $\phi(\xi_1, \xi_4)$ that maximizes the log-likelihood function is associated with the branch labeled b. Therefore, that constraint is

added to the set of constraints. Unfortunately, this method alone can still render errors. The solution to this problem that is used keeps track of all the log-likelihoods along the chosen path as well as those on what they refer to as the frontier of the tree [6]. The frontier of the tree is the complete set of leaf nodes in the tree [6]. If the log-likelihood of the current leaf node exceeds the value of a frontier node, then the system will attempt to revise the past data association decisions to find an increased log-likelihood.

As stated, SLAM is very good at creating maps of static environments and geometrically localizing a robot in those environments. However SLAM is not without its weaknesses. The first criticism of SLAM is that it has no means of determining a difference between geometrically identical regions. This fact alone facilitates the need for a vision based system to assist in the global localization problem. An example of this problem is, as stated in Chapter 1, a building with two geometrically identical floors. Without any type of vision based information, the robot will have no means of determining which floor it is on. The next criticism is that SLAM cannot handle change in an open environment. If something simple is moved then the world is no longer geometrically the same and SLAM has no means of determining or understanding this change. The second type of change that SLAM cannot detect is that of visual features changing. Although this type of change is not important for maneuvering around objects or generating a map, the ability to recognize such changes does indicate a level of contextual understanding of the environment. The final criticism of SLAM is that it has no means of understanding the concept of context. The only feedback it provides is if an object is there or not. This means that ultimately SLAM assumes every edge is a boundary, regardless of the type of boundary.

Although it has never been argued that SLAM is the solution to all spatial cognition, it is useful for demonstrating the need for visual information in attaining such a goal. The concept of adding vision to a system using SLAM is hardly novel [13,18,19]. Visual SLAM is an attempt to solve some of these issues. This will be discussed in conjunction with landmark detection.

## Landmark detection

Landmark detection is the use of very specific features contained within an environment as cues to indicate the robot's location typically with navigation in mind [13,18,19,35]. The objects used can be natural objects that stand out [13,18], or artificial objects placed within an environment [35]. As with SLAM there are many different implementations of using landmark detection so this paper will focus on one and offer a critique based on the principle of landmark detection with regard to spatial recognition. The work presented here was performed by Y. Lee and J. Song [13]. Their technique incorporates both landmark detection and SLAM to create a hybrid grid/vision map. The map will be constructed using an infrared scanner (IR) and autonomously detected objects.

The objects will be detected based on five different features. Those features are hue, saturation, intensity, SIFT (Scale Invariant Feature Transform) keypoints, and object contours. First the system must learn which objects to find. The structure of the system is shown in Figure 2-5.

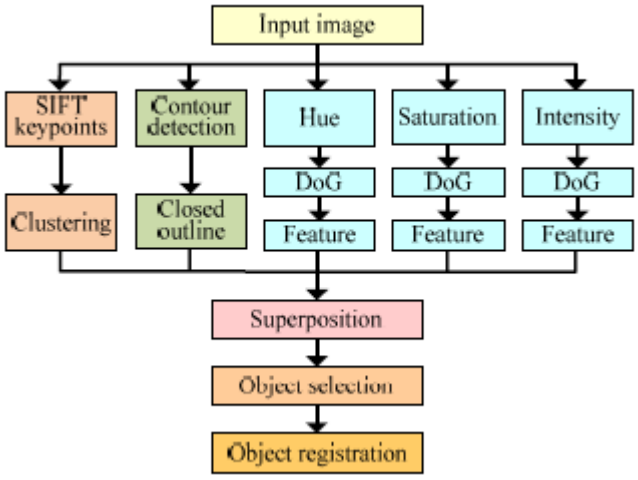



Figure 2-5 Overall structure of the proposed scheme. [13]

From the five uncorrelated features the SIFT keypoints and contours of objects are used to select object candidates and the color information is the criterion used to decide whether an object candidate is useful or not.

The SIFT keypoints are used because they are invariant to scale, rotation, and viewpoint. The system extracts the keypoints in a cascaded filtering approach that identifies candidate locations and examines them in further detail [14]. This is to reduce the amount of complex calculations performed. The four main stages of feature extraction are scale-space extrema detection, keypoint localization, orientation assignment, and keypoint descriptor [14].

The contours of the objects are found using the Canny edge detection algorithm [15]. This implementation adds a scale multiplication to the Canny edge detector [16]. Canny first demonstrated that a good edge detector should fit three criteria: good detection, good localization, and low spurious response [16]. The scale multiplication implementation modifies the original edge detection filter from Canny's work $f(x) = -xe^{-x^2/2}$ and makes it $f_s(x) = -x\,e^{-x^2/2s^2}/s^2$. A small scale $s_1$ and a large scale $s_2$ are used to detect the step edge. The responses of the scales are then multiplied

to provide a product $P_w(x)$. The purpose of the multiple scales is to reduce the amount of false edges allowed with too small of a filter and increase the accuracy from using too large of a filter. This increases the quality of the detection, thus propagating better edge detection through the localization and thresholding steps [15].

The color information used is the hue, saturation, and intensity. These are used because they are more intuitive and give more information than the RGB color domain. This comes from the HSI space being closer to how humans see and the three channels not being correlated [13]. After the conversion to the HSI space, the features from the hue, saturation, and intensity need to be extracted. This is done by first convolving each channel with a variance of σ and 2σ. The convolution is used to smooth the boundaries. The DoG convolution images are then calculated to represent the complexity of patterns in each of the channels [13]. They are calculated according to equation 2.6

$$I = |L(\sigma) - L(2\sigma)| \tag{2.6}$$

Here L(σ) and L(2σ) are the Gaussian convolution images with mask having a variance of σ and 2σ respectively. The magnitudes of the features are represented as a gray scale image and are shown in Figure 2-6 with the results of feature extraction shown in Figure 7.

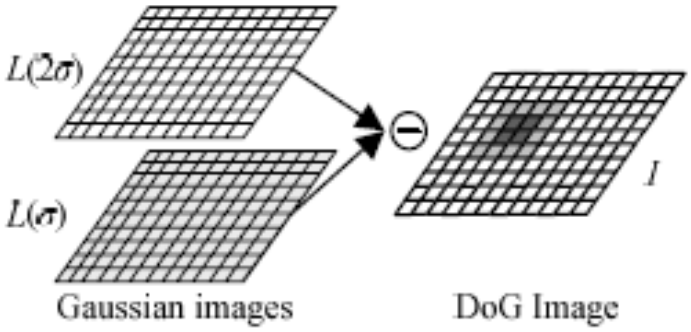



Figure 2-6. DoG image as a primitive feature image [13]

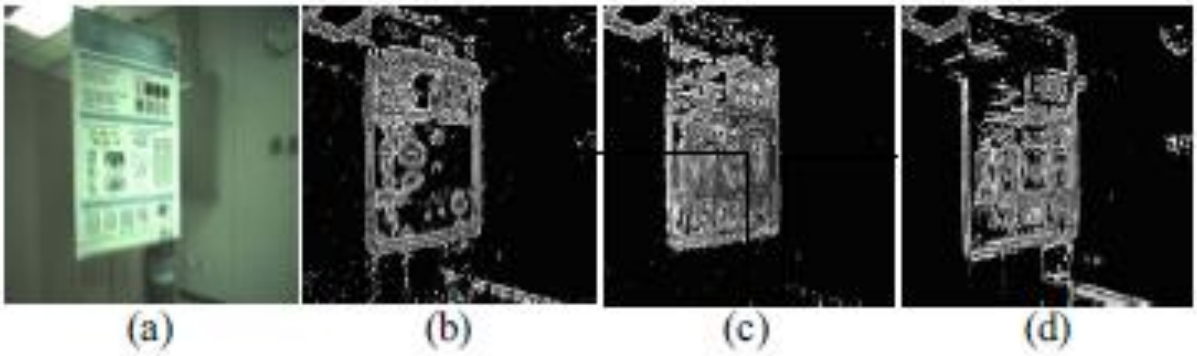



Figure 2-7. Feature images; (a) Input camera image, (b) features extracted from hue image, (c) features extracted from saturation image, and (d) features extracted from intensity image. [13]

Figure 2-7 shows the camera image and the features extracted from the hue image, saturation image, and intensity image in (a), (b), (c), and (d), respectively. Before the feature images are combined they are normalized. This is because they represent features with different ranges [13].

The combination of the feature images is done with an adaptive weighting approach. The weights are determined by the distribution of gray scale values. This means that the denser the features are in the area the greater their weight will be. The following equations mathematically describe this process.

$$I_F = \omega_H I_H + \omega_S I_S + \omega_I I_I \tag{2.7}$$

$$\omega_H = \frac{\sigma_S^2 \sigma_I^2}{\sigma_H^2 \sigma_S^2 + \sigma_S^2 \sigma_I^2 + \sigma_I^2 \sigma_H^2} \tag{2.8}$$

$$\omega_S = \frac{\sigma_H^2 \sigma_I^2}{\sigma_H^2 \sigma_S^2 + \sigma_S^2 \sigma_I^2 + \sigma_I^2 \sigma_H^2} \tag{2.9}$$

$$\omega_I = \frac{\sigma_H^2 \sigma_S^2}{\sigma_H^2 \sigma_S^2 + \sigma_S^2 \sigma_I^2 + \sigma_I^2 \sigma_H^2} \tag{2.10}$$

*I* represents the feature image, $\omega$ is the weight of each color channel's image, and σ represents the distribution of the gray scale values for each color channel. The subscripts H,S,I, and F represent the hue, saturation, intensity, and combined feature image,

respectively. An example of the combination of the color channels is shown in Figure 2-8.

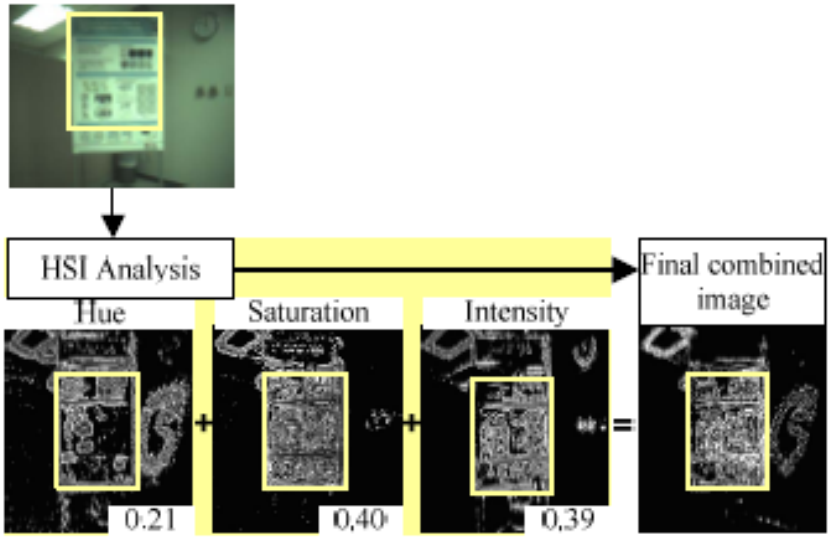



Figure 2-8. Adaptive weighting; final combined image where hue, saturation, and intensity feature images are combined with different weights. [13]

The SIFT features and contours are used to select the region within the yellow box. The variances of the three channels are then found within the selected region. These variances are then used to calculate the weights according to equations 2.8, 2.9, and 2.10. Finally, the final combined image is created based on equation 2.7.

Now with each of the features Figure 2-9 demonstrates how the system determines whether or not to keep a candidate object, based on the color features.

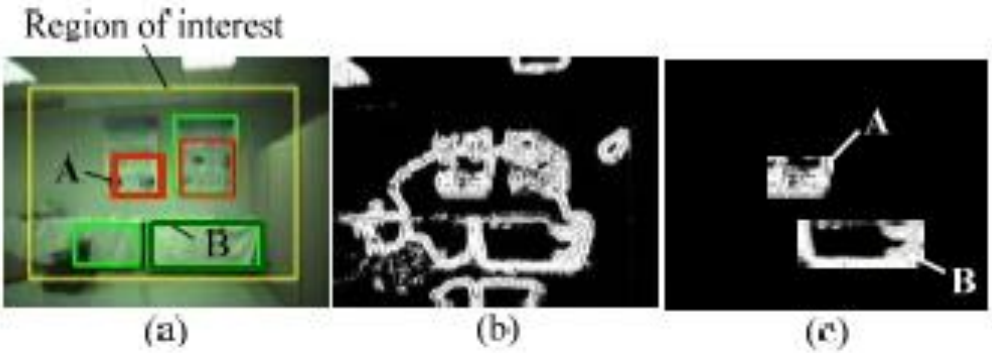



Figure 2-9 (a) Object candidates represented as rectangles, (b) the final combined image where hue, saturation, and intensity feature images are combined with equal weights, and (c) HSI information of regions A and B in the final combined image. [13]

Figure 2-9 (a) shows the original image with a region A and region B labeled. The others were eliminated in order to simplify the explanation. Region A was selected based on a clustered region of SIFT keypoints and region B was selected based on contours detected. The outer yellow rectangle marks the edges of the image that were processed. This self imposed edge is to eliminate detecting only parts of objects because they run off the image. Figure 2-9 (b) shows the three color channels combined into a single image. Figure 2-9 (c) shows just regions A and B of Figure 2-9(b). From Figure 2-9 (c) it can be seen that region A is very salient (or has a high grey scale value for the corresponding pixels) while region B is not. Therefore, region A should be kept as a candidate object.

The final step in selecting an object is to check the gradient of gray scale values up to 10 pixels outside the selected boundary of the object. The purpose of this step is to ensure that an entire object is grabbed. This process is shown in Figure 2-10.

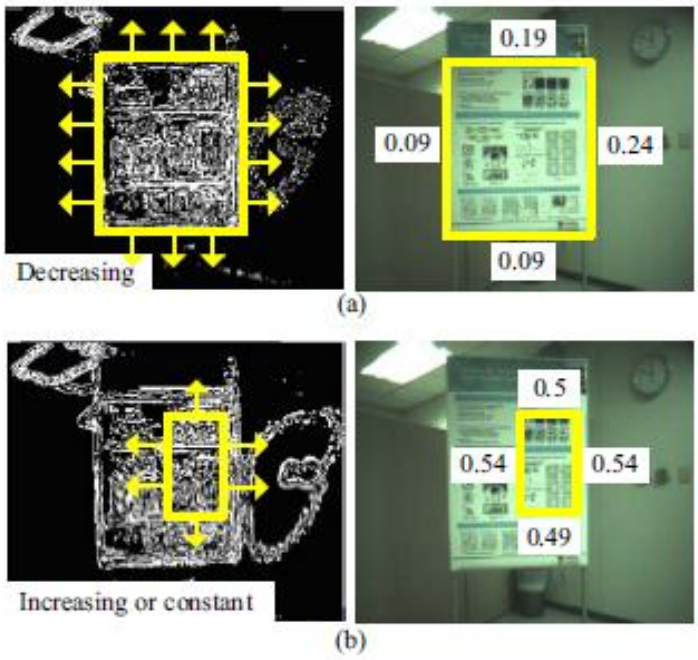



Figure 2-10 Investigation of gradient of object candidates. [13]

In the images to the left of Figure 2-10 (a) and (b) the arrows show the areas in which the gradients of the gray scale are taken. The images on the right show the values calculated. Because the values outside of Figure 2-10(a) are so small that object is kept, and because the values outside of the object in Figure 2-10(b) are large this object is discarded. Once the objects have been stored, they can then be used for navigation purposes.

The SLAM implementation used by this system operates on the same operational principles (using a range finder to map and localize the robot in the room) as the previously described SLAM implementation, however in this system an extended Kalman filter (EKF) replaces the use of the MRF and an infrared sensor (IR) is used in place of a laser range finder. Because one SLAM system has already been described in detail please refer to [17] for more information on this implementation of SLAM.

In practice the system was able to create a grid/vision map of a three room area. This area is shown in Figure 2-11.

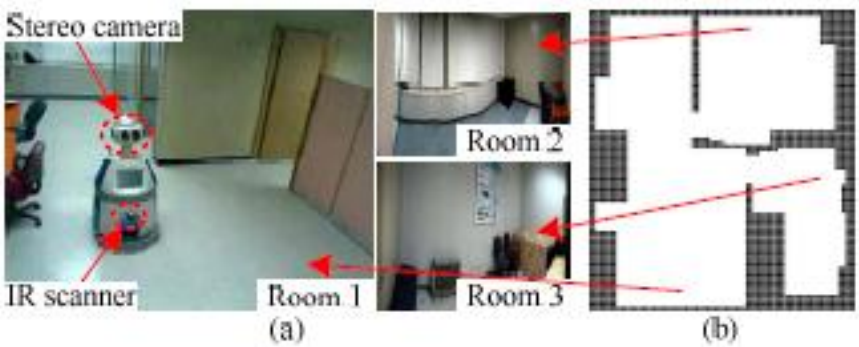


Figure 2-11 Experimental environment; (a) mobile robot platform and experimental environment, and (b) CAD data. [13]

The EKF-based SLAM recreation of this room is shown in Figure 2-12.

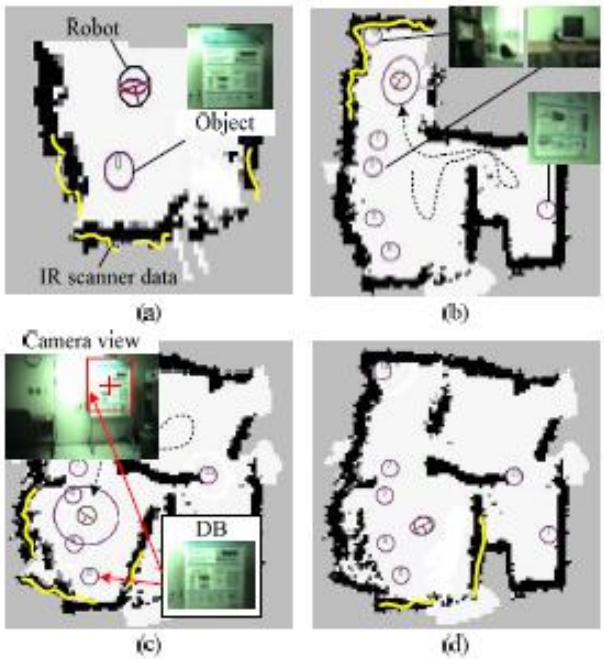


Figure 2-12. Indoor SLAM with autonomous object registration [13].

Figure 2-12(a) shows the initial state of the robot. In Figure 2-12(b) the robot has built the map of the environment while moving around in it. While in the area labeled as room 2 in Figure 2-11, the robot encounters slippage due to the carpet. This creates a distortion in the map shown in Figure 2-12(c). However due to the recognition of a registered object, the distortion is correct. This correction is shown in Figure 2-12(d). The

correction of this distortion is a very good example as to why landmark detection is so useful when incorporated with SLAM. It provides a solid solution to the data association problem faced by SLAM systems.

Once the map is created the ability of the robot to localize itself within the map needs to be determined. Figure 2-13 shows the performance of the EKF-based SLAM (solid line) versus using pure odometry (dotted line) for localization.

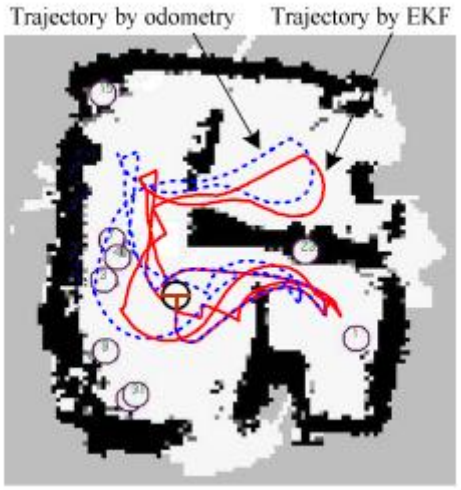



Figure 2-13 Comparison of robot trajectory by odometry (dotted) with that by EKF-based SLAM (solid). [13]

Figure 2-13 clearly shows the superior ability of the EKF-based SLAM approach to localization of just pure odometry. Although this improvement is obvious, it would have been helpful had the authors provided the exact position of the robot for an evaluation of the EKF-based SLAM to a ground truth.

Even without the actual position of the robot, this system shows the importance of visual information for a system. In this case the visual information provided a means for the system to recover from an error caused by the environment. Without the visual information the map generated would have been off by a significant margin. Although

the landmark detection proved itself to be useful it is still not adequate for generating a system capable of robust localization. A few criticisms of landmark detection are that the systems can be fragile due to the need for a robust recognition of specific landmarks, they do not have any broad sense of the world or contextual knowledge of it, and because of the top-down nature of the processing, extracting more information will be difficult.

A landmark detection system is based on finding very specific features in an environment. Therefore, locating those features is absolutely essential to the system performance. Unfortunately, it is not always easy to identify features in an environment that can change, and even when it is easy, the systems are often not robust enough to be practical. The fragility of a landmark detection based system comes from the need to find these very specific features in an image of a potentially changing world. The system shown previously can demonstrate this fragility. Imagine if the system had explored room 2 and suffered the same slippage due to the carpets. However, when the robot came out of the room, the objects it was capable of recognizing were occluded from its view. Because of the robot's need for specific objects it would have been incorrect in where it believed it was. Although in this case, SLAM may have corrected it, the weakness of looking for specific objects can be seen.

Another issue landmark detection cannot address, is any broad understanding of the world. The system shown uses multiple features to extract objects out of the images. As long as these objects meet the threshold requirements, the robot does not care what they are. Since the robot has no knowledge of the rest of the room, other than the edges, it has no means of knowing what type of room it is in. An example of this would be the knowledge of a kitchen as mentioned in Chapter I. Because the robot is looking for

specific features that allow it to segment something unique, it is unlikely the robot would recognize all the key objects (e.g., cabinets, sink, fridge, and stove) that belong in a kitchen. Even if the robot does happen to segment out one of the objects, except for the stove, they can all exist in other various types of environments. Which means in order to robustly recognize "a kitchen" versus "this kitchen", the robot must be able to gain context from everything in the room and then determine which parts of the room are unnecessary. Another feature that cannot be addressed is the notion of context change. This means that as with range finder based SLAM systems, the robot will not have any notion of a different area. It will merely map areas and locate some objects within the area. This means that it will only be able to give out geographical locations, or at best, a location based on the landmarks it sees, which may not be of any significance in describing the actual location to a human being.

The final criticism of landmark detection is that the top down approach to extracting information is not efficient for spatial cognition. It could be argued that these are merely tools for navigation and localization and a broad understanding could be added to these systems. However, finding specific features in an image first means that the entire image has already been processed, and in order to get a broad view of the image, the entire image must be reprocessed. This reprocessing is redundant. By taking a bottom up approach and gathering information on the entire image on the first pass, it is then possible to focus in on areas of attention. In a worst case scenario, this would only require the reprocessing of the area of the image that contains the interesting information. This type of processing will be covered in the next section, template matching.

## Template Matching

Template matching is the use of information extracted from the image as a whole for location recognition. As with the previous two techniques discussed, there exist multiple implementations of this principle [21,26,33]. The system focused on here will be the work of K. Ni et al. [21]. Ni et al used epitomes as a model of an environment for the purpose of location recognition. This work builds on the work of Torralba et al. who used global gist features for training a mixture of Gaussians model to represent the locations [26].

Using epitomes serves numerous purposes. The first is that it adds translation and scale invariance into the model of an area [21]. Also it allows for changes in viewpoint and illumination, motion, occlusions, and non-Lambertian effects [21]. The final improvement that an epitome provides is the computational efficiency that comes with using a compact and dense model of the area versus comparing a test image against a database of exemplar images [21].

The epitome used by [21] is derived from the work of Jojic et al [22]. Epitomes, according to Jojic, are a condensed representation of an image used as a generative model of the image. The original image is then described by the epitome and the mapping from the epitome to the set of pixels in the original image. An example of an original image, an epitome, and a reconstructed image is shown in Figure 2-14.

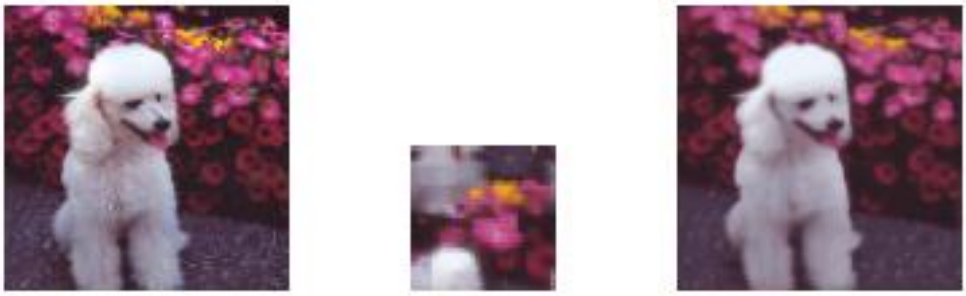



Figure 2-14: Appearance epitome: The input image (a), is epitomized in the texture (b), shown enlarged two times. The reconstructed image is shown in (c). [22]

The location epitome, used by Ni et al [21], is a panoramic representation of an area described by the mean $\mu(j)$ and precision (inverse variance) $\lambda(j)$ where j represents the pixels in the location epitome e. Figure 2-15 shows the mean and variance image of a panoramic view.

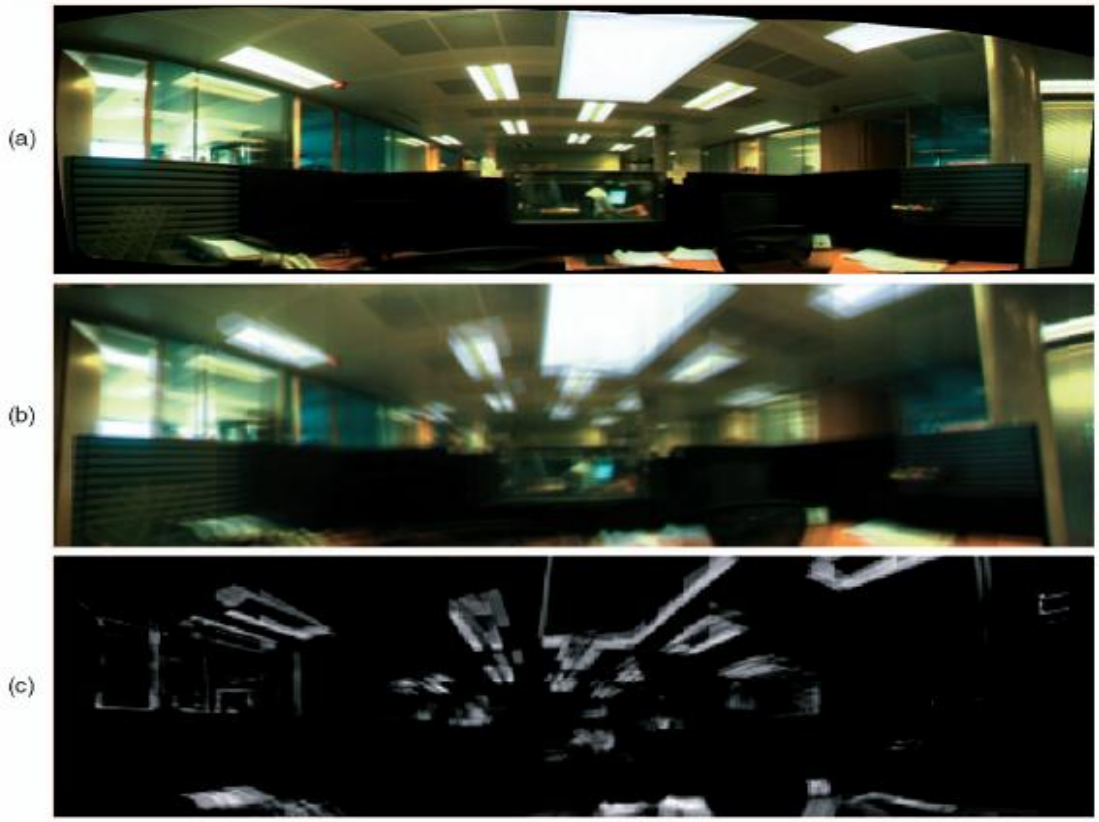



Figure 2-15. (a) Panorama, (b) epitome mean, and (c) epitome variance. The input images were taken with a camera rotating about a fixed point. The learned epitome looks similar to the stitched panorama of Fig. 1a, with the additional variance channel capturing uncertainties. [21]

The location epitome is described as e = (μ, λ). It is assumed that every image I with size N x M is generated from a $N_e$ x $M_e$ location epitome. A Normal Gamma prior is used over the epitome to guarantee the behavior of the model is well defined for unused locations in the epitome [21]. This is described as:

$$p(e) = \prod_j \mathcal{N}\left(\mu(j); \mu_0, \beta\lambda(j)\right) Gamma(\lambda(j); a, b) \tag{2.11}$$

where $\mathcal{N}(y; \eta, \gamma)$ is a Gaussian distribution over y with a mean of $\eta$ and precision of $\gamma$.

The mapping from each of the location epitome pixels to the coordinates in the image I is defined by $\mathcal{T}$. Ni's work uses two types of mappings, 2D translations and scaling. The translations are considered in both the horizontal and vertical directions, thus bounding them by the size of the epitome. The scaling has three discreet levels

(0.8;1.0;1.3). These levels make the computations tractable, and covers the scale spectrum completely enough for most scenes tested by Ni. Finally the prior distribution over the mappings $p(\mathcal{T})$ is assumed to be uniform.

The image I is generated based on the epitome e and the mapping $\mathcal{T}$. The pixels in I from each epitome mean have a Gaussian noise from the variance map added to them and are then copied according to $\mathcal{T}$.

$$p(I|\mathcal{T}, e) = \prod_i \mathcal{N}\left(I(i); \mu(\mathcal{T}(i)), \lambda(\mathcal{T}(i))\right) \quad (2.12)$$

where coordinate i is defined on the input image and I(i) is the feature of the pixel i in the image. $\mathcal{T}(i)$ is the location in the epitome of the $i^{th}$ pixel maps to [21].

The next step in this work is to find a single epitome $e^* = (\mu^*, \lambda^*)$ that maximizes the probability of observations. Using a generative model, every image is independent and identically distributed given an epitome e [21]. The joint distribution over e, a set of T images $\{I_t\}$, and their mappings $\{\mathcal{T}_t\}$ is

$$p(\{I_t\}, \{\mathcal{T}_t\}, e) = p(e) \prod_{t=1}^{T} p(\mathcal{T}_t) p(I_t|\mathcal{T}_t, e) \quad (2.13)$$

The posterior distribution, given $\{I_t\}$, over e and $\mathcal{T}_t$ of $\{I_t\}$ is

$$p(\{\mathcal{T}_t\}, e|\{I_t\}) = p(e|\{I_t\}) \prod_{t=1}^{T} p(\mathcal{T}_t|I_t, e) \quad (2.14)$$

This happens as a result of the mapping of an image into an epitome being independent of all other images, given the epitome and the image. Because this work is looking for the $e^*$ that maximizes $p(\{I_t\})$, the exact posterior distribution is approximated as

$$p(\{\mathcal{T}_t\}, e|\{I_t\}) \approx \prod_{t=1}^{T} p(\mathcal{T}_t|I_t, e^*), \text{ with} \quad (2.15)$$

$$p(\mathcal{T}_t|I_t, e^*) \propto p(I_t|\mathcal{T}_t, e^*) p(\mathcal{T}_t) \quad (2.16)$$

using $p(e|\{I_t\}) = \delta(e-e^*)$. This represents a variational inference on the model and from [22], the log $p(\{I_t\})$ can be bound as

$$\log p(\{I_t\}) \geq \beta = \sum_t \sum_{\mathcal{T}_t} p(\mathcal{T}_t|I_t, e^*) \log \frac{p(I_t, \mathcal{T}_t, e^*)}{p(\mathcal{T}_t|I_t, e^*)} \tag{2.17}$$

This bound can now be maximized using the Expectation Maximization algorithm. The maximized bound is found by iterating between finding $p(\mathcal{T}_t|I_t, e^*)$ as shown in (2.16) and then updating $e^*$. For simplification Ni described the function

$$\mathcal{P}_j^x = \sum_t \sum_i \sum_{\mathcal{T}_t: \mathcal{T}_t(i)=j} p(\mathcal{T}_t|I_t, e^*) I_t(i)^x \tag{2.18}$$

which allows the update for $e^*$ to be written as

$$\mu(j)^* = \frac{\beta\mu_0 + \mathcal{P}_j^1}{\beta + \mathcal{P}_j^0} \tag{2.19}$$

$$\lambda(j)^* = \frac{b + \beta\mu_0^2 - (\beta + \mathcal{P}_j^0)(\mu(j)^*)^2 + \mathcal{P}_j^2}{a + \mathcal{P}_j^0} \tag{2.20}$$

The epitomes do not necessarily have to model appearance information. They can also be used as a generative model of categorical data such as image labels [21]. This is done as follows: First assume training information has been obtained from K difference locations. Then let $e^L$ represent a label epitome with every pixel coordinate j modeling the discrete distribution over K labels, $e_k^L(j)$. Also place a Dirichlet prior with psuedocount α over each label. Next, given $e^L$ and the mapping $\mathcal{T}$ an image $I^L$ of discreet values is created according to the equation

$$p(I^L|\mathcal{T}, e^L) = \prod_i \prod_k [e_k^L(\mathcal{T}(i))]^{\delta(I^L(i)=k)} \tag{2.21}$$

Then performing the same variational inference as before, the update for the location epitome is obtained as follows

$$e_k^L(j) = \frac{\alpha + \sum_t \sum_i \sum_{\mathcal{T}_t: \mathcal{T}_t(i)=j} p(\mathcal{T}_t|I_t, e^L) \delta(I^L(i)=k)}{K\alpha + K \sum_t \sum_i \sum_{\mathcal{T}_t: \mathcal{T}_t(i)=j} p(\mathcal{T}_t|I_t, e^L)} \tag{2.22}$$

A final note, when there is no training data for an epitome location, the distribution over the K possible values is uniform with probability 1/K.

It is also possible to create a joint epitome model of different features. Each feature is associated with each other by means of the mappings used. Let the epitomes representing F possible features be represented as $e^1,\ldots, e^F$. Then given the epitomes and mappings of the epitomes, the conditional distribution becomes

$$p(\{I^f\}|\mathcal{T},\{e^f\}) = \frac{1}{z}\prod_f p(I^f|\mathcal{T},e^f)^{\lambda_f} \quad (2.23)$$

where $0 \leq \lambda_f \leq 1$ is the preference for a feature. As done previously, the log of the probability of the observations is bounded and maximized.

The four types of features used were raw RGB pixels (as used in [22]), gist features, disparity maps, and local histograms. The gist features used build upon Torralba et al. [26]. The goal of using a gist feature is to define the location without having to specify objects within the location. The first step in obtaining the gist feature is to use a steerable pyramid [27] with six orientations and four scales. Next in order to keep limited spatial information, the images are broken into 4x4 local grids and the mean magnitudes of the local features are averaged over the grids. Finally the resulting gist features are scaled to have a zero mean and standard deviation $\sigma = 0.115$ [21].

A disparity map provides depth features that are more robust to changes in illumination [21]. The algorithm used to calculate the disparity maps is based on DP-based stereo matching and comes from [25]. This is a dynamic programming algorithm designed to create a synthetic image to correct for gaze in a teleconference situation. Their goal was to create an image that appears to be looking directly into a monitor during a conversation. Ni's system creates the final depth features by calculating the local histograms of the disparity maps found during that process.

The local histograms used are a combination of RGB and disparity features. The use of appearance and depth local histograms provided good generalization of the data adding invariance to small rotation, translation, and nonrigid deformations [21]. They project each image into a matrix with $B_N$ x $B_M$ cells. For the experiments performed, $B_N$ = 3 and $B_M$ = 2. Within each cell, the feature responses were quantized into B bins. For the RGB features B = 50, and for the disparity features B = 6. This means that the training image, from which a Gaussian epitome is learned, are represented as $B_N$ x $B_M$ x B vectors.

Once the location epitomes have been generated, the final step is to create a location map for each epitome. This map defines the distribution $p(L|\mathcal{T})$ of each location's labels for the positions in the epitome. When a new image I is presented, the location is found by computing $p(L|I)$. This is done as follows

$$p(L|I) = \int_{\mathcal{T}} p(L,\mathcal{T}|I) = \int_{\mathcal{T}} p(L|\mathcal{T})p(\mathcal{T}|I) \tag{2.24}$$

which is efficiently performed using convolution. The process is shown in Figure 2-16.

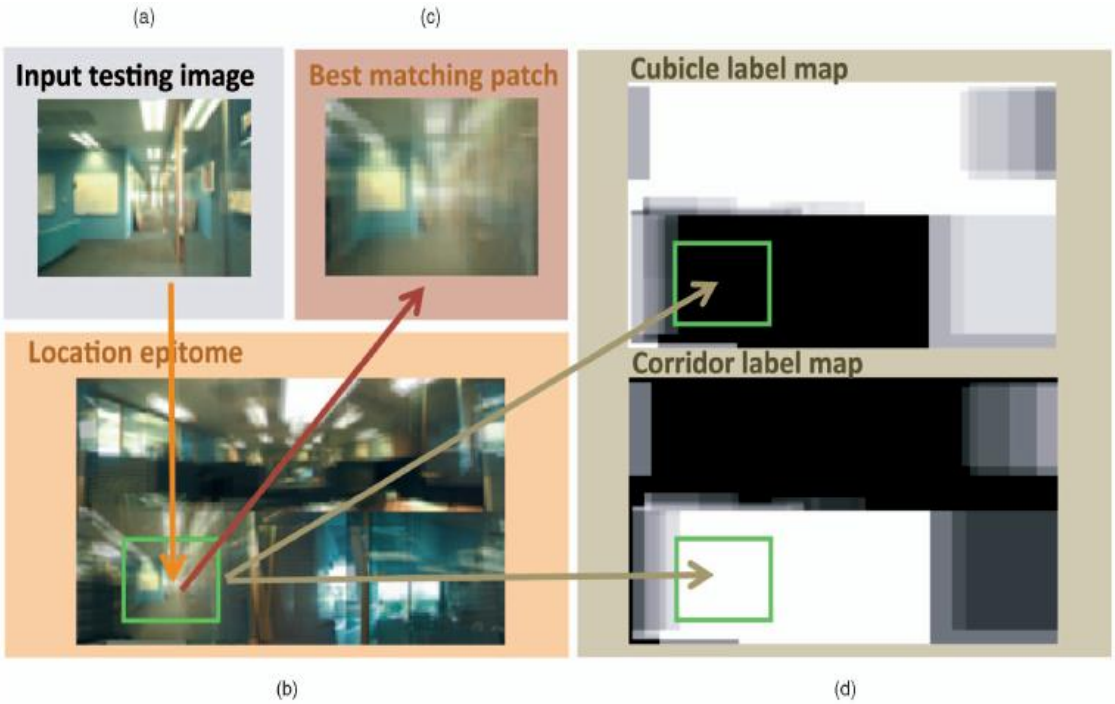



Figure 2-16. The recognition process. (a) The input testing image is convolved with (b) the location epitome. Then the best label is found as the one that maximizes (2.29). Note that the posterior of mappings $p(\mathcal{T}|I)$ tends to be very peaky, and the optimal label is usually decided by the best mapping position (the green rectangles in (d) the location map). In this example, the corridor class gets many more "votes" than cubicle. [21]

Figures 2-16(a), (b), and (c) show the location epitome that maximizes (2.29) through convolution. Figure 2-16(d) shows the location map indicating that the system matched best to the corridor class.

The experiments for the epitomes were performed on a data set from MIT. The data provides images of 64 different locations translating into 64 location epitomes. The results are shown in the precision-recall curve in Figure 2-17.

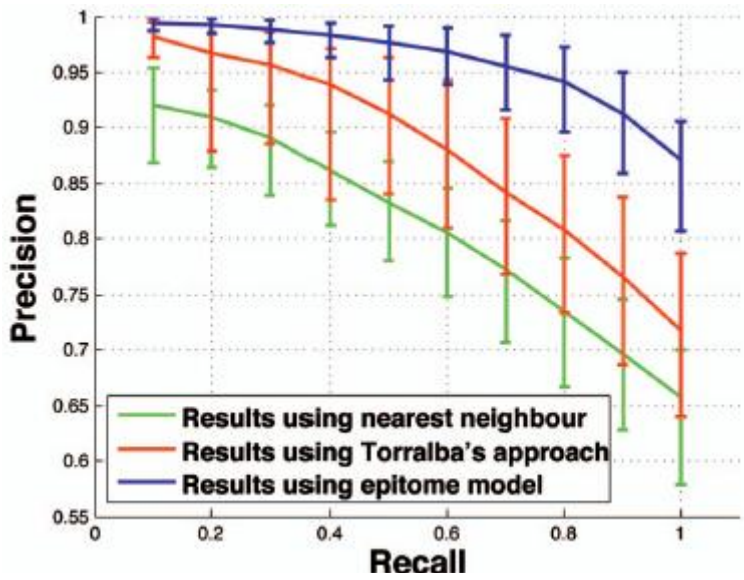



Figure 2-17. Location epitome versus GMM. Precision-recall curves illustrate the median recognition success for the nearest neighbor model, the GMM model (red) and the proposed epitome model (blue). The scale and translation invariance of the location epitome leads to more accurate recognition results. Following [26], the error bars indicate variability in accuracy across different image sequences. [21]

This graph shows that the use of epitomes provides a higher precision-recall curve, indicating that the translation and scale invariance that epitomes provide is valuable information. The next experiment performed incorporated the different features mentioned previously. It was also performed on a new data set gathered by the author that was significantly smaller. The results are shown in Figure 2-18.

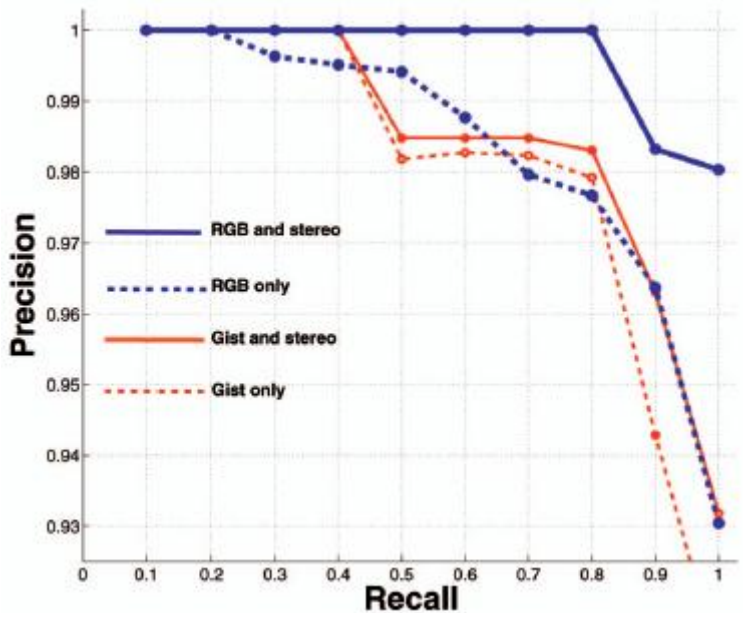



Figure 2-18. Comparing RGB, Gist, and Depth features. The precision-recall curves when using RGB or gist features, with and without stereo disparity features. [21]

Figure 2-18 implies that RGB and stereo is the best combination for epitomes, however because of the data set used this may be misleading. The MIT data set had large changes in illumination which made the gist features easier to distinguish. The new data set did not have as much change in illumination.

Epitomes are reasonably capable of localizing a system. However, they lack some features necessary for true spatial cognition. First the system lacks any sense of what it is looking at. Although enough information exists within an epitome to rebuild an image, there is no distinguishing any of the percepts. This means that if anything is moved within a room there is no means for this system to recognize that movement. Another criticism of this work is there was no break down of where it failed. This would be useful because the system obviously broke down where the differences between like locations were significant. This would have allowed for a gauging of how much change is necessary for the system to fail. For example, would the system be able to handle if a

wall were painted a different color in the hallway? The importance of this information comes from the system needing to know if it recognizes what it sees or not. Even with object segmentation the system should not necessarily recognize a newly painted wall. It should however recognize that it does not recognize the wall and can therefore remove that percept from the localization process and add the percept as a novel object. An epitome has no means of handling this type of situation.

## Final Statement

The goal of this system is going to be to solve some of the weaknesses of the three types of systems presented while still retaining the information necessary to make it possible to solve the other issues.

# CHAPTER III

# PREVIOUS IMPLEMENTATIONS

## Overview

The visual system used for this work has gone through numerous stages of development. It was originally developed by Dr. Mert Tugcu [1]. In [1] the original architecture was proposed and combined with a working memory toolkit. The goal of the visual system was to create a very robust and reliable segmentation of an unmodified environment. Some of the main features of the architecture are that it:

- used a 10,001 dimensional data space with 10,000 dimensions from a HSV color histogram and one dimension of Laplacian texture to define the percepts based on 15x15 pixel sized patches of the image.
- used a three way K-means nearest neighbor search tree to speed up segmentation
- was trained using supervised learning.

The system was then extended by Dr. Amy Wang [2]. Although she contributed far more than what will be covered here, the parts of her work that are pertinent to this work are:

- the implementation of a minimum spanning tree classification as a means of unsupervised learning
- add novel object detection

The next work on the system was performed by Dr. Jonathan Hunter [3]. His work focused on using this visual system for human motion segmentation. This work resulted in:

- improved performance using normalized feature vectors
- improved autonomy of the minimum spanning tree
- demonstrating the autonomous system working in a natural environment

The final additions to this work were performed in [4]. The main contributions of this work that led to the current state of the system were:

- allow the K-means search tree to be updated in real time with additional training and new percepts
- add change detection to the system
- apply maximum likelihood estimation to the classification process

Through all of this work, the system, although not without its issues, still retains a great deal of potential for future applications.

## Vision System with Working Memory Toolkit

The work in [1] aimed to create a vision system that was capable of reliably segmenting percepts in an unaltered environment and then using that segmentation to learn a behavior using working memory. Working memory is defined as "a theoretical framework which refers to a temporal type of storage that retains elements that are active and being manipulated for a short period of time" [1]. Because the current work focuses on improving the visual system, the working memory aspect of [1] will not be discussed further.

Early on the decision was made to use a very high dimensional feature space for this system because it allows for a high capacity to learn. In an adequately large space

very subtle differences can be detected in segmenting similar objects. Unfortunately, this decision does not come without its problems.

The first issue to deal with is having enough data to adequately train a system with such a high dimensionality. According to [47], the amount of data should be five times the size of the feature vector space. Fortunately, a video sequence of images inherently contains a great deal of information. Therefore, extracting a data set of that size is relatively simple.

The next problem with very high dimensional feature spaces is that as the number of dimensions rise, parametric classifiers based on Eigen values, Eigen vectors, etc. become less useful [1]. Also, the calculations required for any of these techniques are very expensive and difficult. A nearest neighbor (NN) classification technique, which only uses distance calculations, was used instead.

The NN technique is a very powerful technique for getting accurate results. Unfortunately, it is also very slow. In order to deal with the speed issue here an approximate NN search tree was used. The benefits of this search tree were two-fold. First, it greatly reduced the processing speed while maintaining robust performance. Second, it was far easier to train in real time than parametric approaches.

The final problem associated with the very high dimensional feature space was the storage and calculations of such large vectors. This was dealt with by using a sparse vector representation. Because this system used a color histogram extracted from 15 x 15 pixels patches in the image, a sparse vector greatly reduced the size and calculations required for each vector.

A flowchart representing the implementation of this visual system is shown in Figure 3-1.

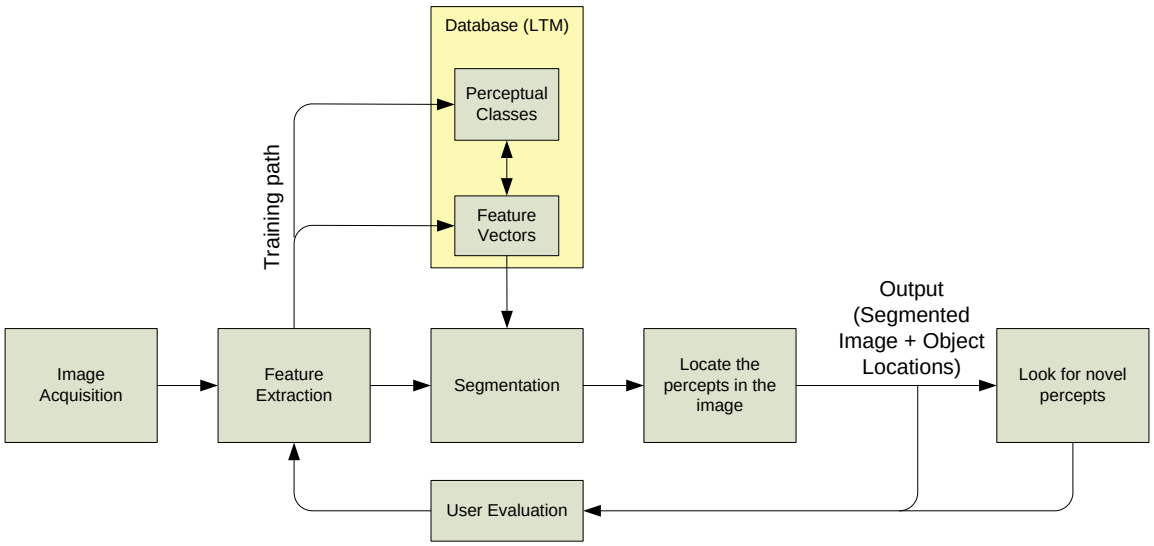



Figure 3-1. Flowchart of the perceptual system [1]

Figure 3-1 shows that the process begins with the acquisition of an RGB image. The RGB image is than converted into an HSV image in order to represent the image intuitively as the hue, saturation, and value of a color. The hue value represents the color of the pixel and represents an angle from 0° to 360° although it is usually a number normalized between 0 and 1. The saturation is the purity of the color and is represented from 0 to 1. The value parameter defines the brightness of the color, or the grayscale of the pixel represented. This parameter is also represented from 0 to 1. Figure 3-2 shows the HSV color map represented in a three dimensional space.

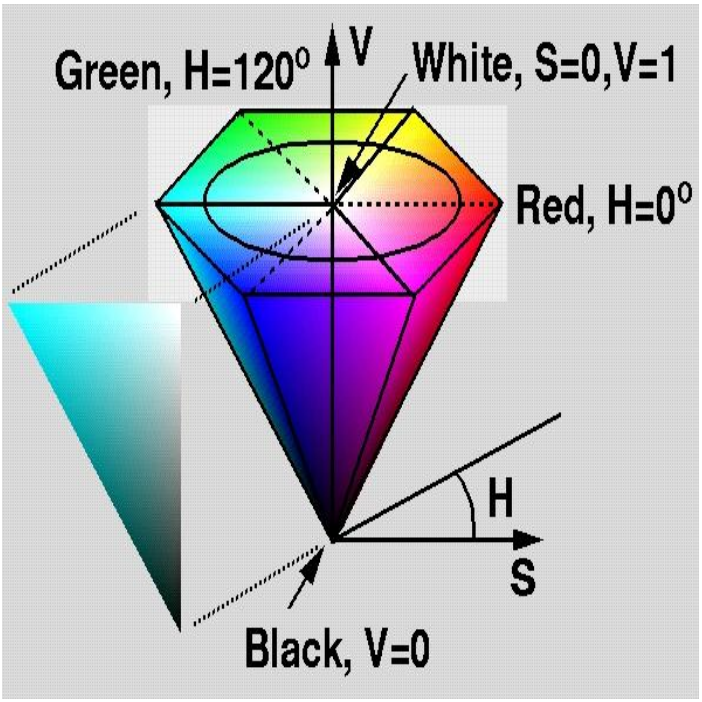


Figure 3-2. HSV color domain. [1]

Once the HSV values are obtained the feature extraction is performed. The image is broken into 15x15 patches that have a 10 pixel hop in both the vertical and horizontal directions. This means that the first patch, starting in the top left corner of the image, will begin at pixel coordinate (0, 0) and the second patch will begin at pixel coordinates (10, 0). Then when the first row is completed there will be a 10 pixel vertical hop downward. The overlap is used to help blend the boundaries of objects. Then a probability density function (pdf) of the distribution of the HSV colors in a patch is found. The pdf is computed from a histogram of the HSV colors that have been quantized into 10,000 bins. This process is performed by first evenly distributed the hue into 100 bins, ranging from 0 to 1. Then the saturations and values are distributed into 10 bins each, also ranging from 0 to 1. Finally these three values are combined resulting in the 10,000 different possible color features. Because of the 10,000 possible color features, and that, in the worst case scenario, the 15 x 15 patches can only provide 225 different potential color

features, a highly sparse representation is used here. Therefore each patch is represented by a feature vector that contains two vectors. The first vector holds the index of each color feature detected, and the second vector holds the value of the color feature. This representation provides numerous benefits.

The first benefit is that there is no computational cost for increasing the dimensionality of the feature vectors [1]. In this work, the Euclidean distance measure is used. Therefore given two vectors X and Y the equation to find the distance between them can be given as:

$$D = \sqrt{(X-Y)^T(X-Y)} = (\|X\|^2 + \|Y\|^2 - 2X \cdot Y)^{1/2} \quad (3.1)$$

Because the norm of the vector only requires the non-zero elements, and the inner product only requires the non-zero elements that exist in both vectors, this representation is immune to increase computational costs due to increased dimensionality. The only way to increase the computational cost is to change the size of the patches used. So, if an N x N patch size is used and $N^2$ unique color feature indices are found, then the worst case distance calculation would require $2N^2 + 1$ index fetches. The additional one comes from a texture feature added to the feature vector.

The second advantage of the sparse feature vector representation is the amount of memory preserved. In Tucgu's work it allows the feature vectors to fit in the virtual memory of a computer. However the real benefit of this reduced size is demonstrated when the feature vectors can fit on a general purpose graphical processing unit (GPGPU). This will be discussed in greater detail in Chapter IV.

As mentioned previously, a texture measure or "roughness" of the region is used. This is found using a Laplacian operator. The Laplacian operator is commonly used for

edge detection where the areas of the image that have quick intensity changes are highlighted. The Laplacian is defined as:

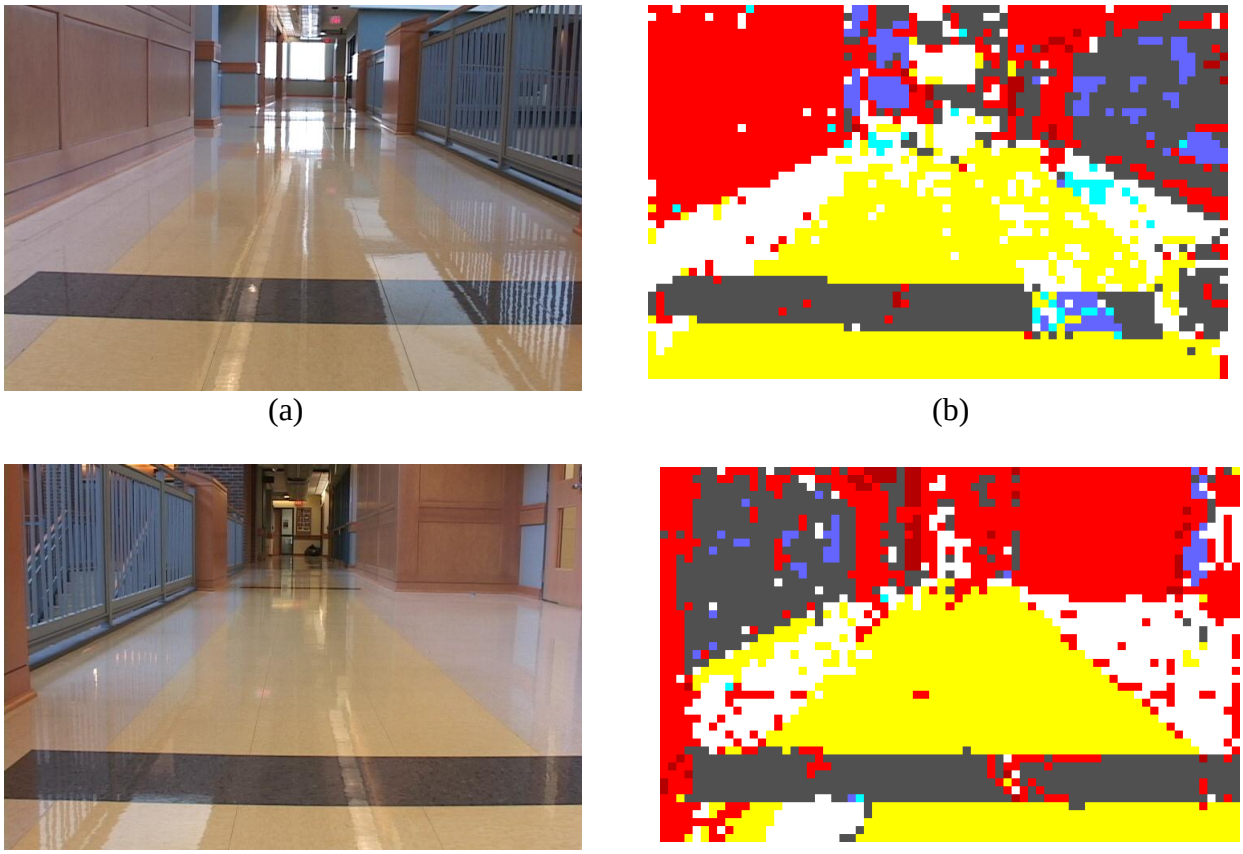
$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{3.2}$$

where I represents the pixel intensities. With the texture measure found, the size of the extracted feature vector is 10,001.

Now that the feature vectors have been extracted they need to be classified using a trained database for the segmentation of future images. In Tugcu's work the classification was performed by the user, making this a supervised learning system. However once the feature vectors in the database were given their appropriate labels, there were two issues to deal with. The first was how are the similarities between the training database and the new feature vectors going to be measured. Based on [48], the best means of determining the relationship between the data is through the Euclidean distance. The second issue was, because the feature vectors represent a 10,001 dimensional data set and according to [47] the amount of data collected should be five times the number of dimensions, the database was extremely large and inefficient to process. An exact nearest neighbor search would take too long to be useful. Because of this, an approximate 3-way nearest neighbor search tree was constructed. The search tree was constructed as follows: The first node or root node of the tree was created by randomly selecting three feature vectors from the training database. The rest of the database was then clustered into three child nodes corresponding to whichever centroid they were closest to, based on the Euclidian distance, in the root node. Then for the three new nodes in the second level of the tree, three new centroids were selected in the same way and the data segmented. This process continued until one of three conditions

stopped it. The first condition that can stop a node from propagating into child nodes is if all the feature vectors in that node represent the same percept. In this case the node is considered a pure leaf node. The second reason a node will cease to expand is if the number of feature vectors in that node is below a preset threshold. In this case it is considered an impure leaf node. Finally, the last reason the tree will cease to expand is that the preset maximum number of levels has been reached. This too results in impure leaf nodes.

Once this tree has been created, it is then used to segment the current image presented to the system. The segmentations work as follows: Once the feature vectors are extracted, the distance from the current feature vector to each of the three centroids in the root node are found. The child node of the centroid that provides the shortest distance to the feature vector will be used next. This will continue until a leaf node is reached. If that leaf node is pure then the label for the percept will become the label that represents the leaf node. If the leaf node is impure then an exact NN search will be performed between the current feature vector and the entirety of the feature vectors represented in that leaf node. The current feature vector will then be labeled by whichever feature vector in the node that it is found to be closest to. Figure 3-3 shows two image segmentations performed by this system.



(a) (b)

(c) (d)

Figure 3-3. Typical segmentation results of the system. (a) West side of the hallway. (b) Segmented image of (a). (c) East side of the hallway. (d) Segmented image of (c). [1]

These results show that in an unaltered real world environment the large objects are segmented well. The areas with large reflections have difficulty, but that is to be expected from a system that has no understanding of the concept of a reflection.

The final step in this iteration of the system is the ability to update the training information on the fly. Building a single tree requires hours of processing time and it is not desirable to have to go through that every time the system is updated. Therefore, the system was built so that anytime new training data is added it will propagate through the tree, as if it were being processed for segmentation, and then added to the leaf node that it is closest too. This brings up the problem of the leaf node exploding in size and

rendering the tree virtually useless due to processing time. That problem was addressed in [4] and will be addressed later.

This work formed the platform for all the following systems. Although it still needs work, it shows the ability to segment real unaltered environments while maintaining enough information about the training data to use those segmentations to complete tasks.

## Autonomous Visual System

In [2], Wang used the visual system created in [1] and upgraded it. She then used the system along with the same WMtk to create a landmark detection system. This form of landmark detection was used to help the robot get from one location to a target. Because the scope of this landmark detection was limited to a single area of a hallway, and the use of WMtk does not focus on location detection this section will focus on the updates and modifications [2] made to the visual system.

The first change [2] made was using a minimum spanning tree (MST) to autonomously classify unlabeled training data. [2] defined a MST as follows:

”The minimum spanning tree method is a graph analysis of arbitrary point sets of data. In a graph, two points can be connected by either a direct edge or a sequence of edges called a path. A loop in a graph is a closed path. A connected graph has one or more paths between any pair of points. A tree is a connected graph without closed loops. A spanning tree is a tree that contains every point in the data set. If a value is assigned to each edge in the tree, the tree is called a weighted tree. For example, the weights for each edge can be the distance between the two points. The weight of a tree is the total sum of

edge weights in the tree. The minimum spanning tree (MST) is the spanning tree that has the minimal total weight among all possible spanning trees for the data set. The minimum spanning tree has the following property that can be used for clustering if the weight associated with each edge denotes the distance between the two points. That is, the weight associated with every edge in the minimum spanning tree will be the shortest distance between two sub-trees that are connected by that edge. Therefore, removal of the longest edge will theoretically result in a two-cluster grouping. Removal of the next longest edge will result in a three-cluster grouping, and so on. These correspond to choosing breaks where maximum weights occur in the sorted edges. When the tree is built, after sorting the edges in decreasing order, the edges can be cut to form clusters."

An example of a MST was provided in [3], and is shown in Figure 3-4.

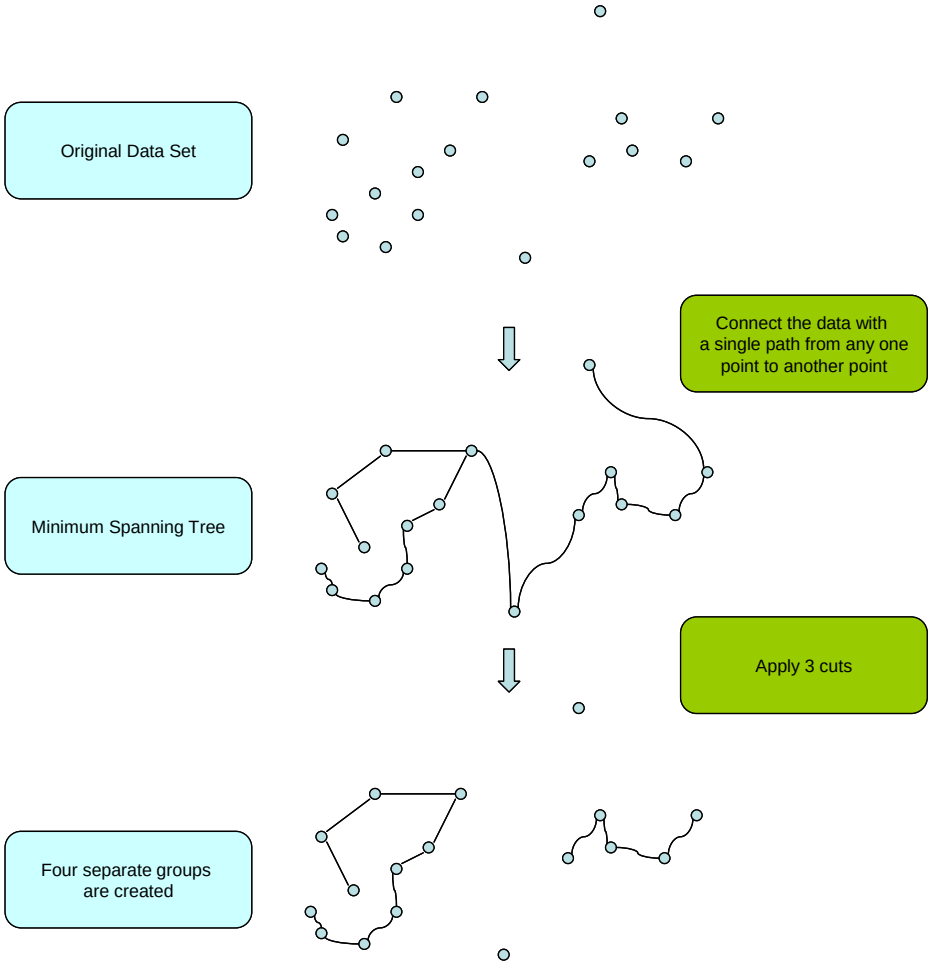



Figure 3-4. An example of a minimum spanning tree. [3]

Figure 3-4 shows how a simple MST will work. First all the data points are connected while maintaining the rule that only one path can exist to connect any two points. Then a preset number of cuts are made on the data. The cuts are made between the data points that have the longest path between them. Finally the clusters fall out. In the example above, four groups fall out. In [2], the number of cuts was preset by the user. This method yielded the following segmentation shown in Figure 3-5.

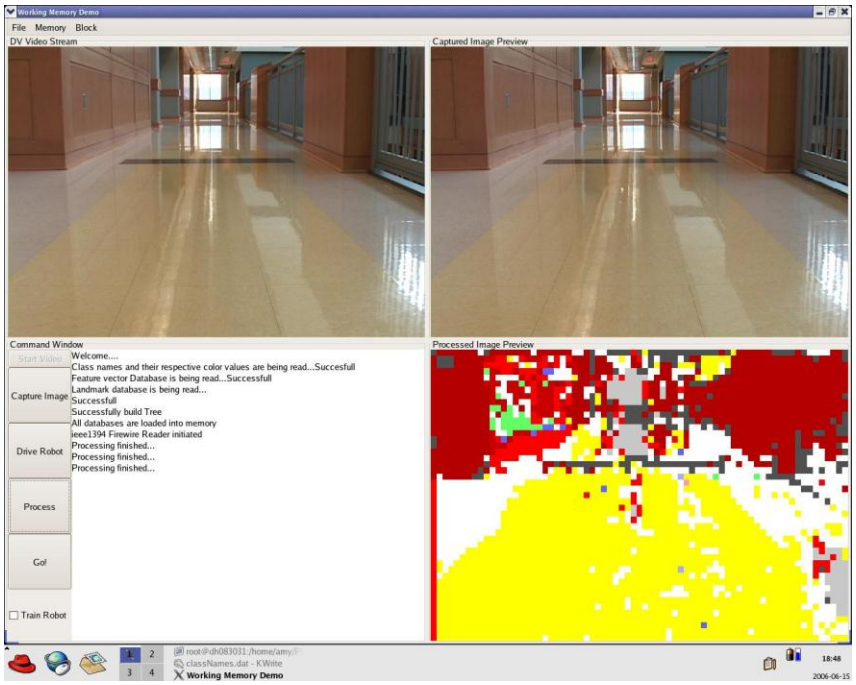


Figure 3-5. Example of image segmentation using MST classification [2]

Figure 3-5 shows the large percepts successfully segmented and the quality of the segmentation is comparable to that of Figure 3-3.

The second contribution [2] made that pertains to this work, was providing the system with novel object detection. In order for a system to be functional in the real world it not only needs to recognize objects, but it needs to know when it doesn't recognize an object and then be able to add the new object to its database. This operation is based off of a calculated threshold using 80 images without the novel object present. The median of the distances of each patch from the feature vector it is closest to in the approximate NN search tree is found. The standard deviation for each patch from the median of the set of medians is then found. The threshold T finally comes from adding the median of the standard deviations to the median of the medians. This is shown in equation 3.3

$$T = median(d_{median1}, \cdots, d_{medianN}) + median(std_1, \cdots, std_N) \qquad (3.3)$$

Now that the threshold has been calculated the robot is driven through the same environment, but with a novel object present. An image of percept distances is formed and segmented using the threshold. In order to determine that the object is not, noise a binary image is created. The bottom half of the image is eroded twice by an 8-connected structure element. Finally the largest group of connected patches remaining is selected as a potential novel object. As the robot gets closer to the object, the size of the connected group should continue to grow. If the number of patches exceeds 100, they are stored and added to the training database. This process is shown in Figure 3-6.

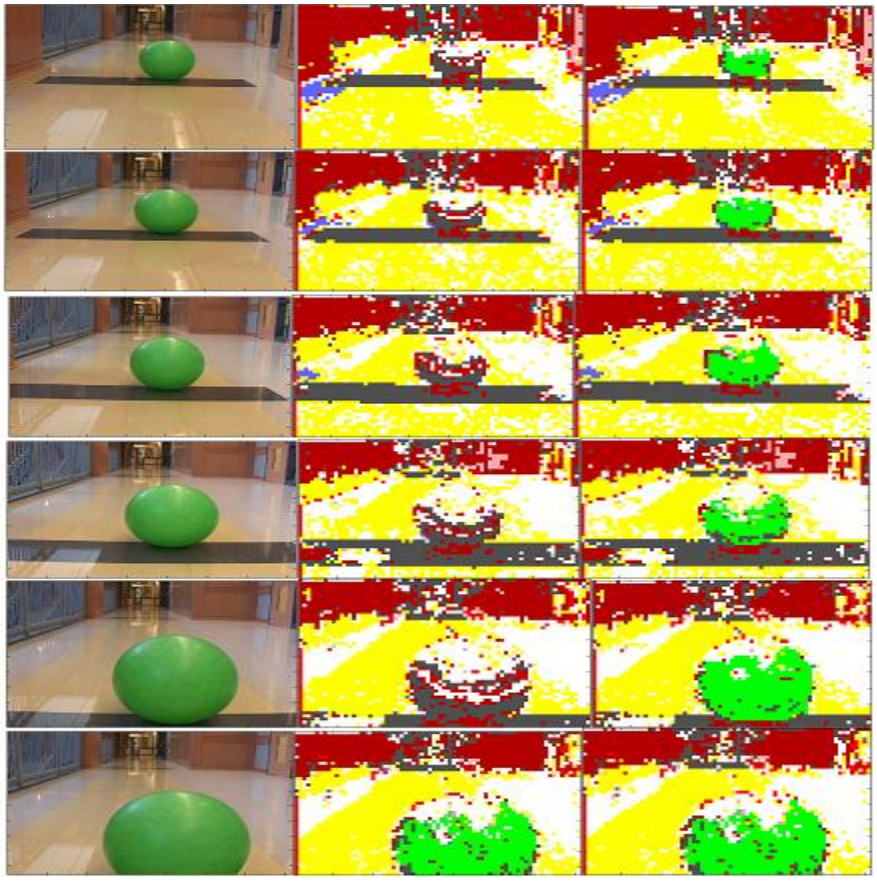


Figure 3-6. (left) the original images, (middle) processed images, (right) processed images after learning. [2]

This work showed two very important abilities of this visual system. The first is the ability to classify clusters without the assistance of a human user (unsupervised learning), and the second is the ability to add new objects to the training database. Some of the weaknesses still present in the system are the time it takes to train the system and process the images, the need of a human user to predetermine the number of cuts for the MST and determine the quality of the search tree created, and the novel object detection can only detect one novel object at a time.

## Vision System for Human Motion Segmentation

In [3], the vision system was used to classify human motion in order to determine what task was being performed. This required the system to learn and classify the motion of the objects in the videos. For more information about human motion segmentation, please refer to [3] as this section will focus on the contributions to the visual system.

The first contribution [3] made to the system was recognizing the value of normalizing the feature vectors. The results of this are shown in Figure3-7.

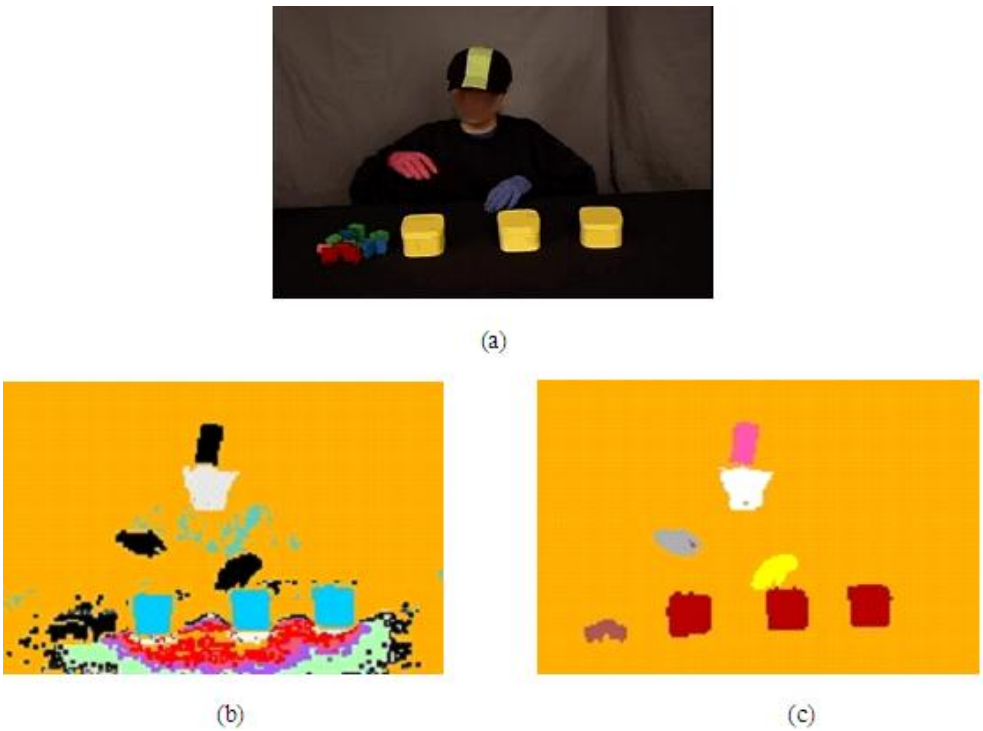


Figure 3-7: (a) Original Image, (b) Segmentation without normalization (c) Segmentation with normalization [3]

The difference in performance comes from the distance measure used and the high dimensionality of the system. Because of the high dimensionality, the non-normalized vectors were forced toward the origin as a result of being on the L1 hyperplane. With the distance from the origin to the center of the L1 norm being $\frac{\sqrt{N}}{N}$ in the N dimensional case, as N grows larger the separability of the feature vectors drops. This issue can easily be viewed in two dimensions and is shown in Figure 3-8.

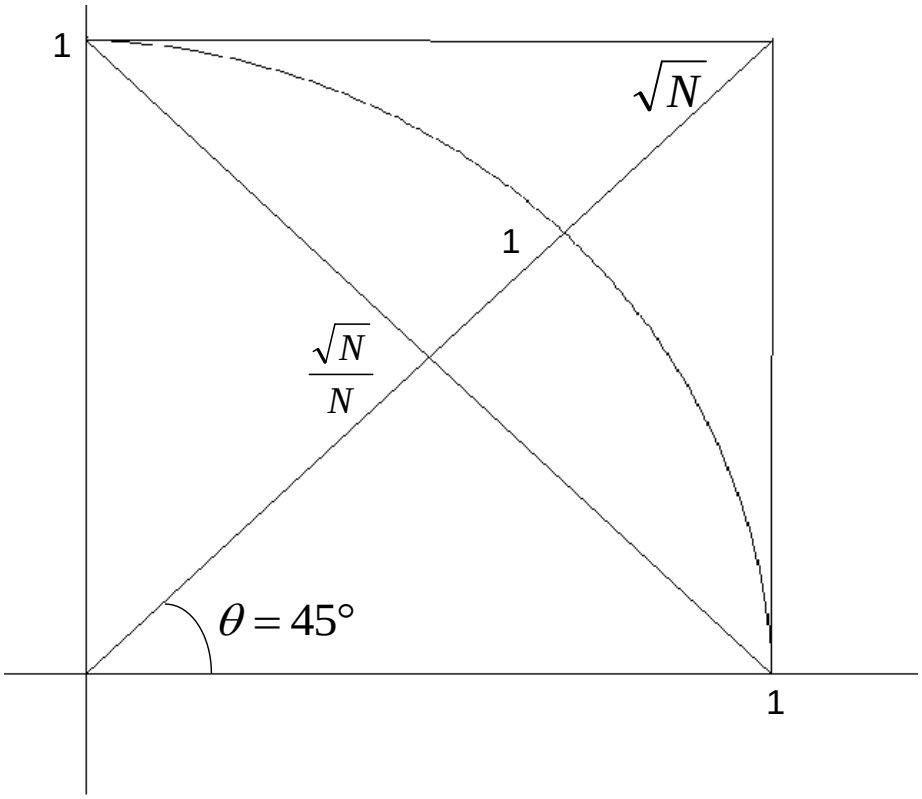



Figure 3-8: Two Dimension Projection Example [3]

In Figure 3-8 the plot is 2-D and thus N=2 and the distance is $\frac{\sqrt{2}}{2}$. Because the actual data used is N=10,001 dimensions, the L1 hyperplane passes significantly closer to the origin. This means that the discriminating power of the L2 norm as a distance measure is greatly weakened. Therefore by normalizing the feature vectors, they are projected out to the hypersphere and the distance measure will work consistently regardless of the dimensionality of the data.

The second major contribution [3] made was the increased autonomy in creating the minimum spanning tree and approximate nearest neighbor search tree. In [2], the human user had to provide the training images and the number of cuts the MST tree used. The problem with selecting the number of cuts was that the number changed for every dataset. This means that the user had to use a trial and error approach to find an optimal

number of cuts each time a new MST search tree was to be constructed. Once the MST had classified the data, [2] created an approximate nearest neighbor search tree. Because the centroids were selected randomly, the quality of the tree's segmentations was always random. If the tree did not perform well the user determined this and created another tree.

The first issue [3] addressed was having the system select the training images from the video presented. This was done by selecting 20 images from the first quarter of the video and selecting 20 images from the remaining video. This resulted in a total of 40 training images that had all objects present. This type of image selection was appropriate in this case, because of the nature of the videos having all objects present at the start of them.

After the 40 images are selected, the feature vectors are extracted from them. Because the human motion segmentation needed a higher resolution than 15 x 15 could offer, a 7 x 7 patch size was used resulting in 21,004 feature vectors from each image and 840,160 vectors total. Due to the number of feature vectors, the database was thinned. This was done in two steps. In the first step any feature vectors that were within 0.0000001 of each other were reduced to a single representation. Then a threshold is found by using the mean distance of the vectors of the first two images. After the threshold is set, any vectors whose nearest neighbor is further away then the threshold is removed from the database. This resulted in a database of 48,283 vectors.

The next issue was to resolve the trial and error method of determining the proper number of cuts used in [2]. This was done by looking at the distances across the MST from largest to smallest. A plot of this is shown in Figure 3-9. Ref [3] found that most of

the databases behaved in a similar fashion with a sharp drop in distance and then settling into a nearly linear decrease of distances. Through experimentation he found that the best number of cuts was at the beginning of the approximately linear tail as shown in Figure 3-9.

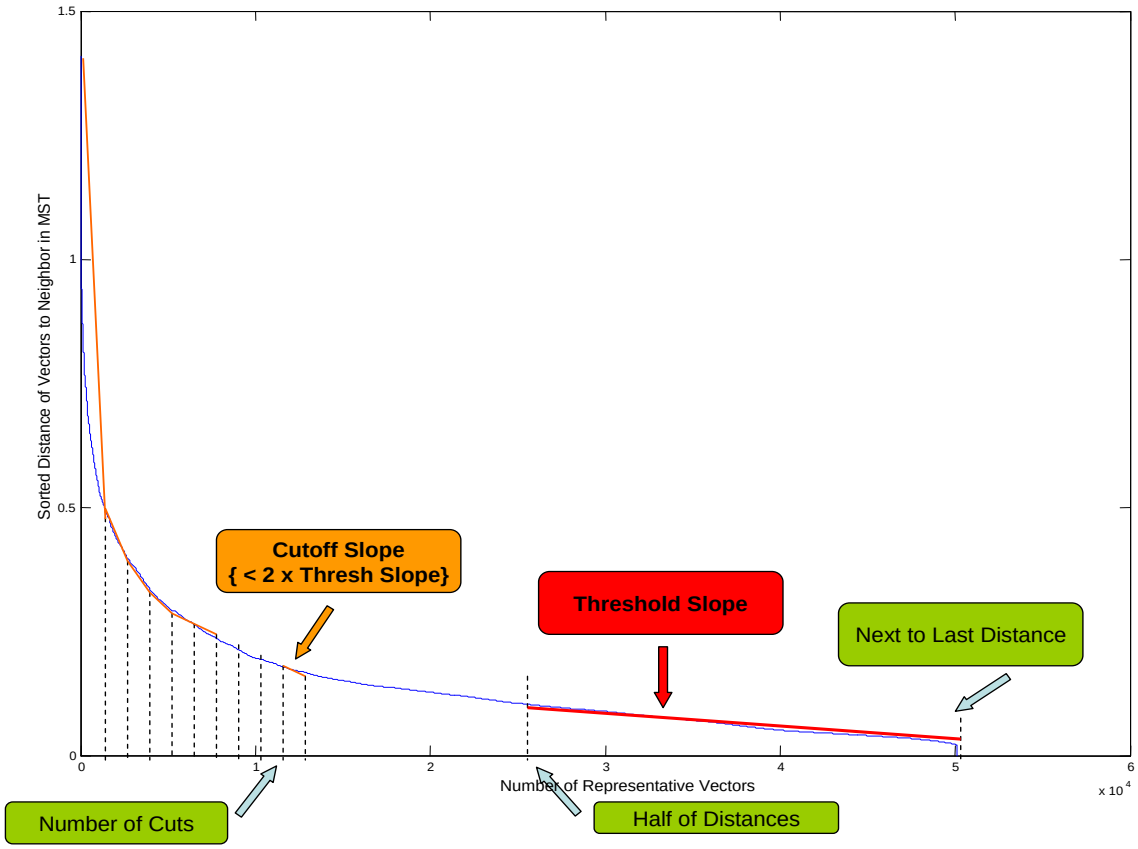



Figure 3-9: Number of Cuts Algorithm [3]

The number of cuts is then determined by first calculating the threshold slope. The threshold slope is the slope of the last half of the distance values. The cutoff slope is then determined by backtracking until the cutoff/threshold ratio was less than two. The number where this occurs is then determined to be the number of cuts applied to the MST.

From here the approximate nearest neighbor search tree was created. Although the tree was created in the same way as [2], a tree validation process was added. During this process the created search tree results of an image segmentation were compared to an exact NN's segmentation results. If the tree provided the same labels as the NN on 98% of the patches then the tree was saved. If none of the trees were capable of 98% accuracy after 20 trees were made then the tree that had the highest accuracy was saved.

The final contribution [3] made regarding the visual system was to demonstrate the performance of the autonomous system in real world environments. The environments tested were the third floor hallway of Featheringill hall on Vanderbilt University campus and the walkway outside Featheringill hall. The results are shown in Figure 3-10.

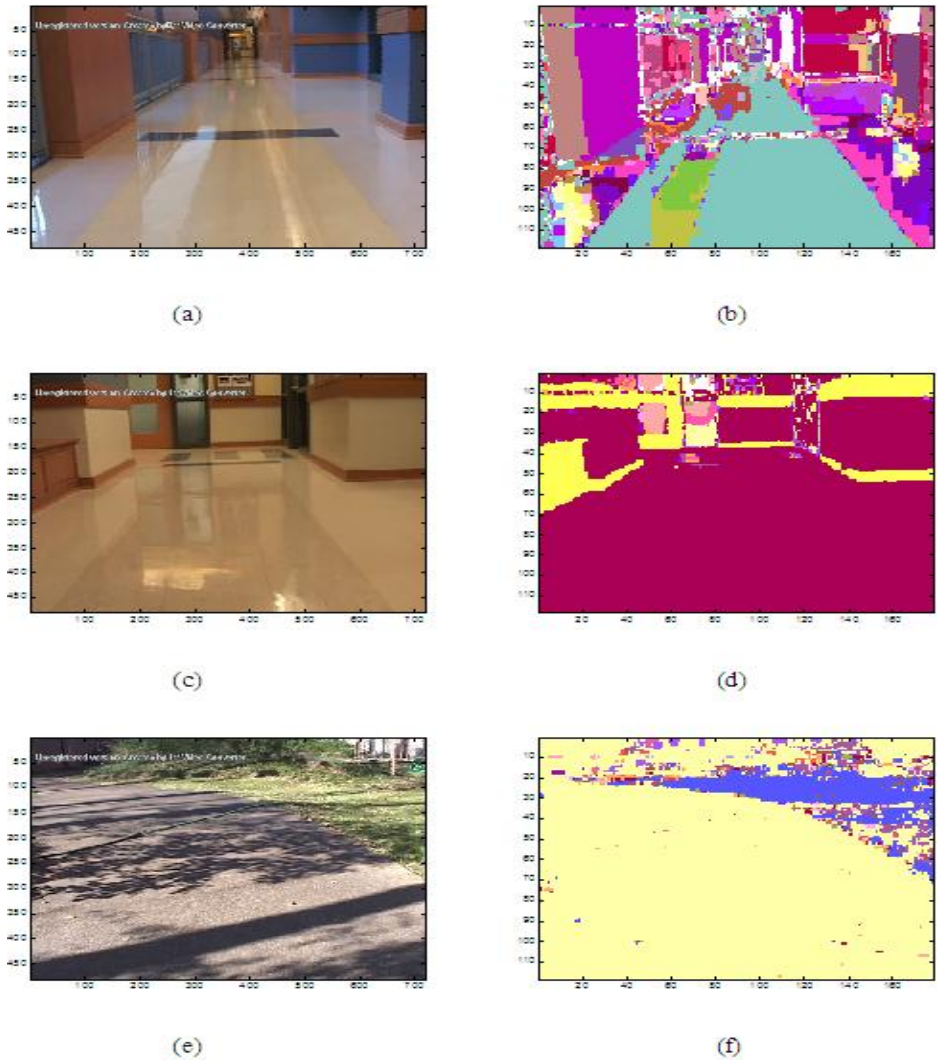


Figure 3-10. Natural Scene Segmentation Examples; (a) Indoor Atrium, (b) Indoor Atrium Segmentation, (c) Indoor Jacob Hall, (d) Indoor Jacob Hall Segmentation, (e) Outdoor FGH, (f) Outdoor FGH Segmentation [3]

Figure 3-10 shows the system adequately performing in multiple real world settings. Although noise is present, the large objects are well segmented in all three images.

This work showed the effectiveness of normalizing the feature vectors which will greatly affect the works presented in the rest of this document. This work also shows that the system is able to behave in a largely autonomous manner and still provide meaningful results. The greatest weakness of this process is that it takes an extremely long time to process any information. The times reported for each step are as follows:

- Thinning of feature vectors ~ 1 day
- Creating MST ~ 4 hours
- Creating approximate nearest neighbor search tree ~ 1.5 days

According to these times it will take more than two days just to classify and set up the training data for use. This time does not even include the processing time of each new image. So clearly time is a serious issue that must be addressed.

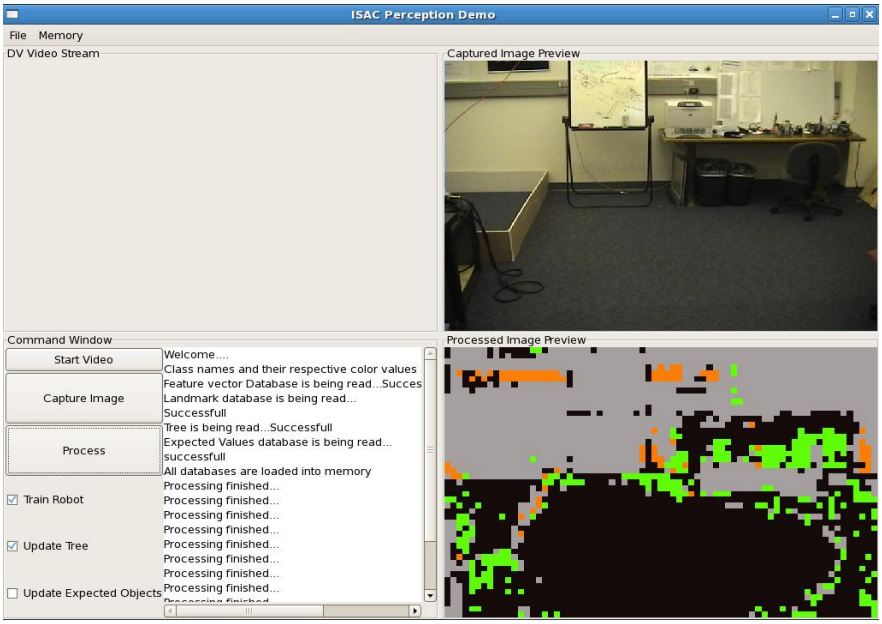
## Implementing Change Detection into the Visual System

The final modifications to the visual system, before the current work began, were done in [4]. The focus of [4] was to solve some of the previous issues with the system as well as add change detection to the system. The two issues this work aimed to address were to allow the tree to expand in real time instead of collecting all the new training vectors in their respective leaf nodes, and to try and speed up the processing time of each image using maximum likelihood estimation at the leaf nodes of the approximate nearest neighbor search tree. The change detection made use of the novel object detection implemented in [2] while allowing more than one novel object to be detected at the same time, additionally a means of determining when objects were moved in a room was developed. The assumption for this system was that it was used on a stationary humanoid robot. In this case the system was run on the humanoid robot ISAC at Vanderbilt University. It should also be noted that this work was done at the same time as the human motion segmentation work so those results were not implemented in this system.

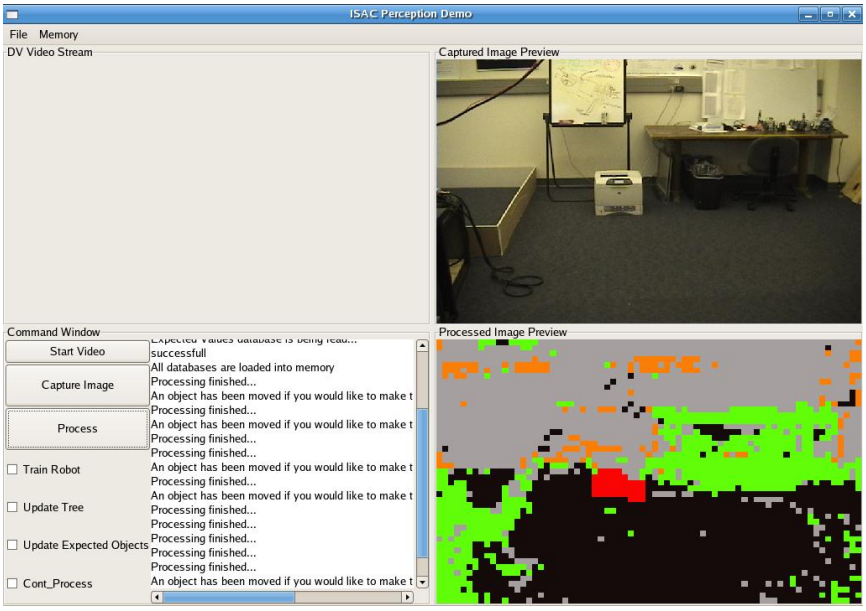
In both [1] and [2], the tree could only be created after the training database had been developed and classified, whether by supervised or unsupervised means. Then when

new feature vectors were added, to improve the current system or to add novel objects, they were all added to the leaf node in the tree that they were closest to according to the distance measure previously described. There are two problems with this method of generating a tree and updating the training database. The first problem is in the supervised training mode. Since the user has to create the entire database, they need to guess at which feature vectors in the training images will help produce the most useful tree. The second problem with this comes from the exact NN procedure used on each leaf node. As the leaf node grows, the performance of the system will suffer. A better way to allow training on the fly is to permit the tree to continue to expand in real time. This means that if new training vectors are added to the system the leaf nodes can simply expand. This ability also solves the supervised training problem as the user can continually check the performance of the search tree and train on the objects that are not performing as well as desired.

As mentioned multiple times, the processing time of the system is of great concern. After testing it was found that a significant amount of the processing time was devoted to the exact nearest neighbor searches at the leaf nodes. Because of this a quasi maximum likelihood estimation (MLE) approach was taken to speed up the processing. This was done by finding out which percept was represented most in each leaf node and classifying that leaf node as that percept. This way the only processing performed was done while propagating through the tree. This approach provided results that were comparable to using the exact nearest neighbor, as scene in Figure 3-11.




(a)




(b)

Figure 3-11. (a) Processed image with MLE tree. (b) processed image using the original search tree and detecting the movement of the printer [4]

In Figure 3-11, the objects that have been trained on and their respective representative colors are presented in Table 3-1.

Table 3-1: List of percepts and representative colors [4]

| Object(s) | Color Representation |
|---|---|
| Wall, White Erase Boards, Printer | Gray |
| Floor | Black |
| Trash Cans, Chair | Green |
| Power Strip | Orange |

An issue with this environment was how similar all the objects were to each other. This resulted in combining multiple objects and made getting a good segmentation extremely difficult regardless of the search tree method. Even so the difference in segmentation quality can be seen in the chair and trash cans. In Figure 3-11(a) both of these objects are not segmented correctly while in Figure 3-11(b) they are. Although there is a difference in segmentation quality, depending on the application the difference in processing time may make it worth while. Using the MLE tree this image took five seconds to process. The same image with the exact NN search tree took 12 seconds to process. Unfortunately depending on the application, neither time may be acceptable which means further study into this issue is required.

The final addition to the system done in [4] was adding change detection. The two aspects of change detection added to this system were novel object detection and moved object detection. The novel object detection was implemented the same way as [2], but slightly modified. The first modification was in setting the threshold distance to determine if an object was novel. In [2] the method for calculating the threshold is shown in Eq 3.4. However in this environment adding the standard deviation of the mean feature vector distance to the mean of the feature vector distances did not result in very

good results. Therefore, two times the standard deviation was used to start with then the threshold was modified by trial and error until an acceptable threshold was found.

The second addition to the novel object in [2] was the addition of the ability to locate more than one novel object at a time. This was done by using a size constraint on the patches that exceeded the threshold instead of finding the largest group of patches that exceeded the threshold as in [2]. In [4] the requirement for a novel object was that there were seven connected patches that all exceeded the set threshold. Figure 3-12 shows an example of the novel object detection.

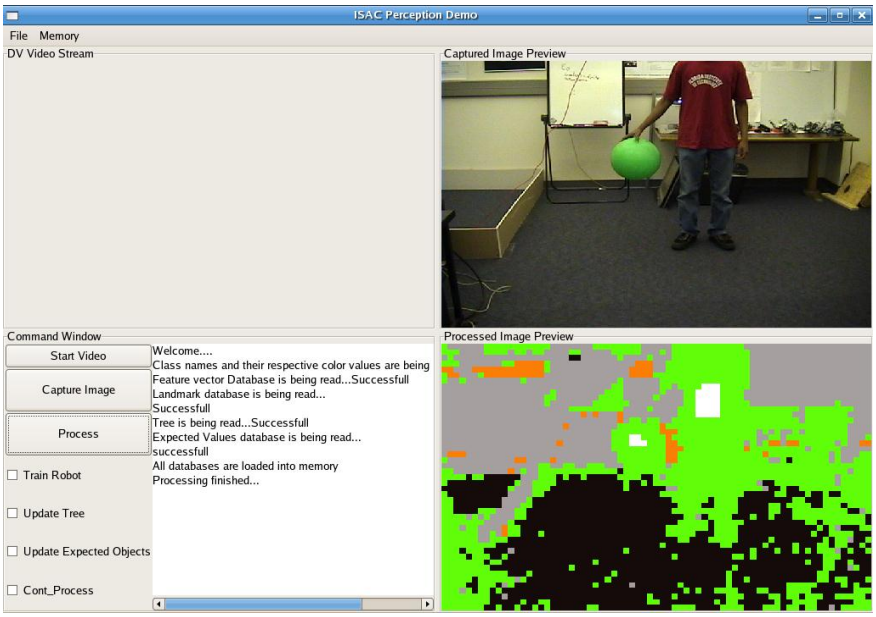



Figure 3-12. Example of novel object detection [4]

The novel objects are represented as white and the rest of the segmentations are according to Table 3-1. The reason the white blobs are so small relative to the objects is because of a patch erosion process performed around the white blobs to prevent noise from being considered. In the example both the ball and person are detected correctly as novel objects in the scene.

The next aspect of change detection implemented on the system was moved object detection. This is the ability of the system to determine if an object that it recognized had been moved in the environment. This was done using a look up table (LUT) to store the labels that the system would expect to see at a patch. The LUT was populated using 10 training images of the room. Then when the new image was processed the resulting labels were compared to the LUT. If seven or more connected patches had labels that were not represented in the LUT then the object was considered to be a moved object. This is shown in Figure 3-13.

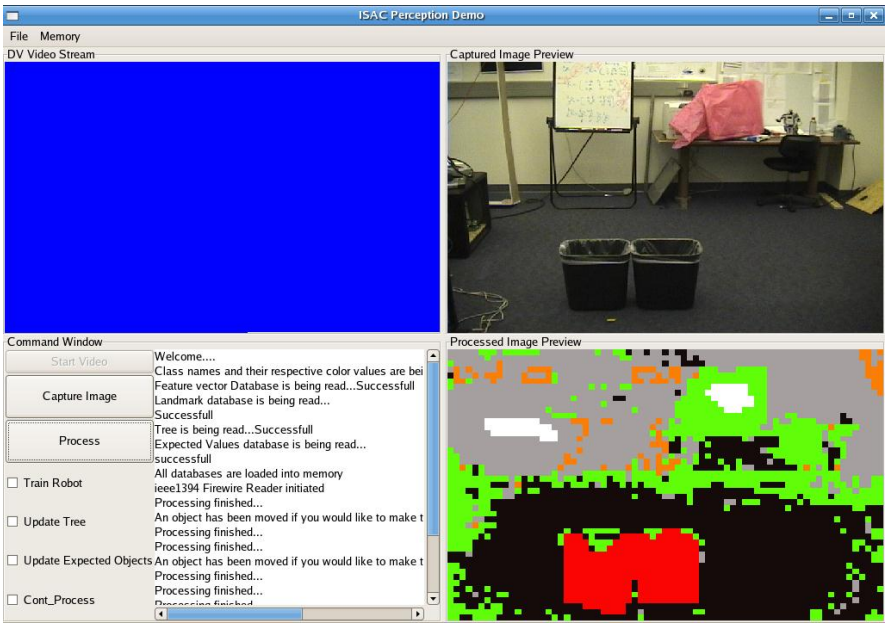


Figure 3-13. Example image of both moved object detection and novel object detection. [4]

Figure 3-13 shows that the system has recognized the trash cans as being moved out from under the table where they were expected. It also demonstrates the systems ability to detect a novel object at the same time. The false novel object in the image comes from the use of the HSV color space and the distance measure used in the system. According to Figure 3-2, white can appear with any hue so long as the value is high and the

saturation is low. This means that the distance measure will record distance measures greater than the threshold if the hue values are different at each patch. One possible solution to this problem would be to map the equivalent white values in the feature vector to one bin, however this was not implemented by [4].

This work showed that the system was capable of reliably performing more visual tasks than simple segmentation. It demonstrated the ability to learn an environment and determine when change had occurred, whether that change is a new object introduced to the environment or a known object being moved in the environment. This means the system has the information necessary to determine the context of what it perceives. It also showed that simple tricks using the search tree were not going to yield the speed desired for a real time visual system. Other techniques such as using a general purpose graphical processing unit (GPGPU) are going to be necessary to help speed the segmentation times up to acceptable speeds.

## Conclusion

All of these works combine to show the potential of using a very high dimensional feature space as a means of segmentation. They show that the segmentations can be autonomous, reliable, and retain a significant amount of information about the environment. A serious issue to address is the processing speed and solving that problem will be discussed in Chapter III. Beyond that draw back, this system shows promise to be able to perform numerous applications robustly and reliably.

# CHAPTER IV

## Manuscript I: Unsupervised Modeling of an Indoor Environment for a Developmental Location Recognition System

Christopher Costello and D. Mitchell Wilkes

Vanderbilt University

Nashville, TN

Submitted as a Regular Paper to the

IEEE Transactions on Autonomous Mental Development

## Abstract

This paper will focus on autonomously generating local models of regions of a larger more global world as seen through the camera of a mobile robot driven around a building. Subsequently, these local models are used to visually recognize the location the robot is currently in. The models are based on what the system determines to be the "dominant" percepts from the global region. These dominant percepts are also used as local percepts within each smaller region, although refined for each local region. These percept models are constructed via clustering in a very high dimensional space (e.g., 10,000 features). The global region is autonomously segmented into local regions using a relative perceptual difference measure between the current image and prior images taken take from a video obtained from driving around the building. Once the local regions have been segmented local representations of the global percepts that exist in each region will be created and used for the location recognition process.

## Introduction

The goal of this work is to create a developmental location recognition system that is capable of autonomously clustering percepts that provide highly useful information about the environment in a reasonable amount of time, autonomously segmenting a global region into local regions, and visually recognizing which local region the robot is currently in based on percepts seen. The percepts that are extracted will be representations of the large objects present in the global region, in this case multiple hallways on the third floor of Featheringill Hall (FGH) at Vanderbilt University. The significance of this type of segmentation is that it is aiming to extract general

information about an entire environment instead of focusing on providing an aesthetically pleasing segmentation of a single image. This type of extraction will allow for this information to be used in the autonomous development of a model representing each of the hallways in a location recognition system, thus demonstrating an incremental learning process.

There have been numerous robotic systems that have attempted to solve the localization paradigm [6,7,13,21,26,32,35]. Many have done so for the purpose of navigation [6,7,13,32,35]. Although navigating through a local environment is very important in mobile robotics, it is not the only step to spatial cognition. These systems still need the ability to reliably recognize where they are in a larger contextual sense. A common approach to localization is referred to as Simultaneous Localization and Mapping (SLAM) [6]. The objective of a SLAM system is to generate a map of the area while localizing the robot within that area. Two other means of localizing robots are through landmark detection and template matching. Landmark detection aims to robustly extract some features out of an image and use those features to determine where the robot is [13,35]. The third method of localizing a robot, template matching, attempts to use the information in an entire image [21,26]. All three approaches have been used successfully.

SLAM, at its roots, is typically based off of using a laser range finder or vision to map a new area and localize the robot within that area. It has been shown to be extremely useful in the area of navigation [6,7]. The weakness of using only a range finder with SLAM is that without any visual appearance information, it is not capable of determining the difference between two geometrically identical locations. Often, then is not much of

a weakness, however some problematic situations can occur. An example of this would be two floors of the same building. It is not inconceivable to think that a tall building would have multiple floors with the same geometric layout. So, even though a range finder-based SLAM will be able to determine exactly where the robot is on that floor, it has no means of determining which floor it is on. A number of systems have proposed to use visual SLAM [13,18,19] which would be likely to overcome this problem. This approach to the SLAM problem typically combines both a range finding sensor with a camera. The range finder will generate the maps while the camera is used for landmark detection to supply more information about an area [13,18,19].

Landmark detection has also been used extensively for navigation [13,18,19,35]. The idea behind landmark detection is to find unique features in the environment and use them to localize the robot. These features can include artificial landmarks in an environment [35] or natural features [13]. Once the robot understands where it is, it can plan its path for navigation. However this type of localization may suffer from several difficulties. The first is sensitivity in finding the landmarks. Extracting exact information from a scene can be very difficult. If the landmarks don't appear exactly as expected, the system may not detect some landmarks, which in turn may compromise the performance of the system. This implies that the system should have a high degree of robustness in landmark detection for successful application. The second difficulty in landmark detection is the localization dependency on the landmarks. Because the robot has no other means of interpreting its environment, failing to detect landmarks can render the robot lost. This dependency combined with the aforementioned detection sensitivity

may limit the robustness of this technique. This is why the combination of landmark detection with SLAM can help with navigation.

At this point, it should be noted that the authors consider these systems to be very good at navigation. Replacement of these techniques is not the intention of this work. The work presented in this paper seeks to add a new dimension to the way robots are able to perceive the overall context of the environment. The addition of SLAM for the purpose of navigation should be considered once the full potential of the proposed system has been explored.

There has also been some work in template matching or contextual based localization done using epitomes [21]. The epitome is created based on the probabilistic information in the training images, and is used to compare the current image to a known location. However the criticism of this work is the same as that of the previously mentioned systems, and that is that they all limit the information used. Although it is suggested that the epitomes will be able to differentiate between "my kitchen and "a kitchen", there is no segmentation of individual percepts performed. This means that they are essentially looking for a measure of difference. Although this is acceptable for location recognition it may limit what such a system will be able to accomplish. Once an epitome is created all of the other information about the individual percepts in the original image is lost. Because of this weakness, it seems that the epitomes would fail to recognize an area if a partial change occurred, such as painting the walls a different color.

The vision system used will be an extension of the work performed in [1,2,3,4]. This work uses very high dimensional sparse feature vectors extracted from overlapping 15x15 patches of in the image. These sparse vectors are then classified using various

methods. In [1] and [4], supervised learning was used to train the system, and in [2] and [3] an unsupervised minimum spanning tree (MST) was used.

These previous works have demonstrated the advantages of using the very high dimensional feature space while noting the training time and processing speed of the image segmentation as an issue. Therefore another goal of this work will be to focus on autonomous training methods that provide results in the fastest possible manner and reduce the processing times to functional real time speeds. Also because the potential of using this very high dimensional feature space representation remains relatively unknown, only the information inherent in this representation will be used.

In this work a training database of ~600,000-900,000 sparse feature vectors from 600 images was gathered. The images come from capturing video while driving the robot around a building. The database was over-segmented using a K= 40 K-means clustering algorithm. K was deliberately set to over-segment the data because the number of useful dominant percepts is unknown. Empirically, K=40 has been found to be much larger than the number of dominant percepts in a global region. After the first, over segmented, set of clusters were obtained, they were deleted or merged based on size and distance measures.

With regard to learning the dominant percepts, this method has shown the ability to extract information from the overall environment that was crucial for the system to recognize its location within the global region. Additionally, it reduced the training time for learning the dominant percepts from approximately 4 days in [3] to less than an hour. This reduction in time came with the penalty of a less aesthetically pleasing

segmentation. However, for the sake of location recognition the most visually aesthetic segmentation is not necessarily the best segmentation.

The global region that the 600 training images were gathered from will be segmented into local regions determined by the system. This is done using an overall image perceptual difference measurement based on the current image and four prior images. Once the global percepts and the local regions have been established the global percepts present in each local region are modeled within that region and used to recognize the local region on future trials.

This paper will be organized as follows. Section II describes the previous implementations of the visual system. Section III illustrates the current implementation of the system. Section IV will cover the results of Section III. Section V will provide a conclusion and future works.

## Background

### A. Overview

The vision system used for the current work is an extension of [1,2,3,4]. Each of these works applied the same general vision system to different applications [5]. The flowchart of the vision portion of these systems can be seen in Figure 4-1. The applications the vision system has been used for are: combining the

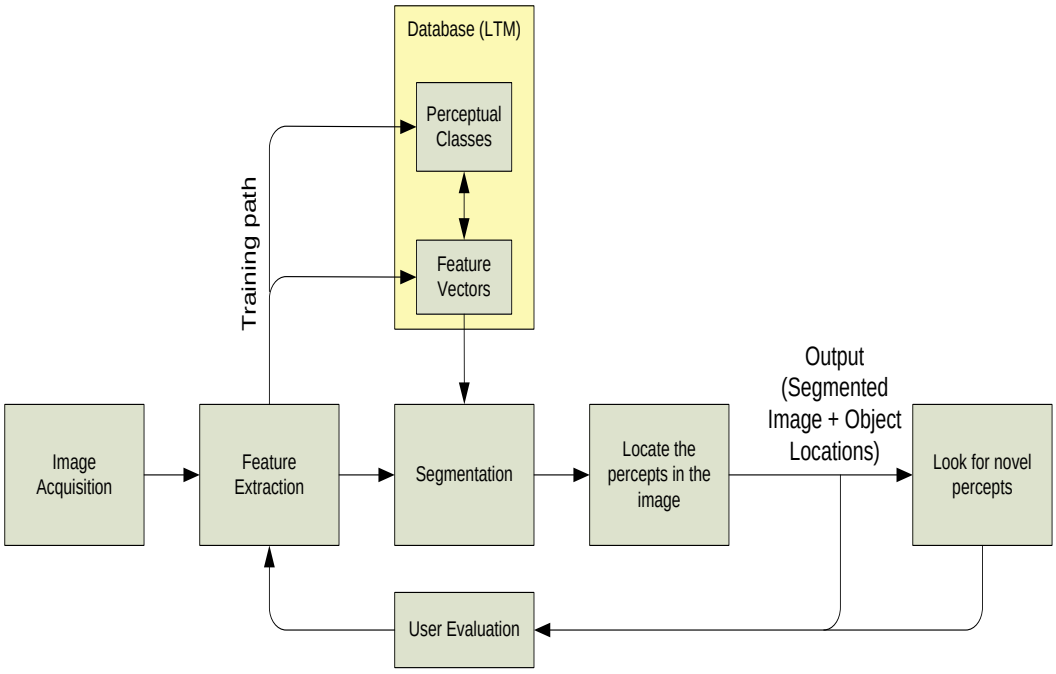


Figure 4-1.Flowchart of vision system. [1]

vision system with a working memory toolkit to find open space in the environment with supervised learning [1], learning the open space in the environment with unsupervised learning [2], human motion segmentation [3], and change detection [4]. Because each of them inspired different parts of this work, in the following background sections, only the most relevant system to the current work will be described in order to avoid repeating information describing all parts of each system.

*B.* Feature Extraction

The decision made in [1] to use a very high dimensional feature space has been the root of the entire visual system. This very high dimensional feature space system has shown good segmenting abilities in [1,2,3,4]. Each implementation has provided its own set of changes to the feature vectors and therefore this section will largely reference [1] while providing the necessary background on them.

The feature vectors that represent the data points in the feature space are 10,001 dimensions. The first 10,000 bins represent HSV color values as a very high dimension histogram of the colors in an NxN patch of pixels. The last bin is a Laplacian texture feature. The typical size for N is N=15.In the current work only the HSV-based features are used so please refer to [5] to learn more about the features.

Because the 10,001 dimensional feature vectors are large, slow and impractical to use, a sparse vector representation was adopted. This representation of the data allowed the system to retain all of the segmenting power while reducing the memory costs and fixing the worst case scenario calculation costs. Now instead of the size of the feature space determining the calculation costs, the size of the patches puts a ceiling on the worst case cost. If the patches are set at NxN pixels and assuming each pixel in a patch is a distinct color, the largest size a sparse vector could be is $N^2+1$, where the 1 comes from the Laplacian texture feature [1]. Therefore the worst case number of indices that need to be accessed by the CPU when calculating vector distances would be $2N^2+1$.

*C.* Training

In [1] and [4] the system was trained using supervised learning. In [2] and [3] the system used a MST for unsupervised learning. Although [2] initially pioneered this approach, [3] is the most up to date implementation. So this work will be the focus of this section.

The first step in training the system was thinning the database of feature vectors. The exact method for this can be found in [3] and resulted in 840,160 feature vectors being reduced to ~ 400,000 feature vectors in approximately1 day of processing.

The next step in training the system involved creating a MST [3]. Clusters are formed from the MST by cutting links between points sequentially, starting with the largest link. The difference between [2] and [3] in regard to the MST was in [2] the human user chose the number of cuts while in [3] this process was automated. The time it took to create a MST was approximately 40 hours. Also, even though this process worked well for the human motion segmentation environments, when used in the natural environments shown in Figure 4-2 some human intervention was necessary due to under segmentation [3].

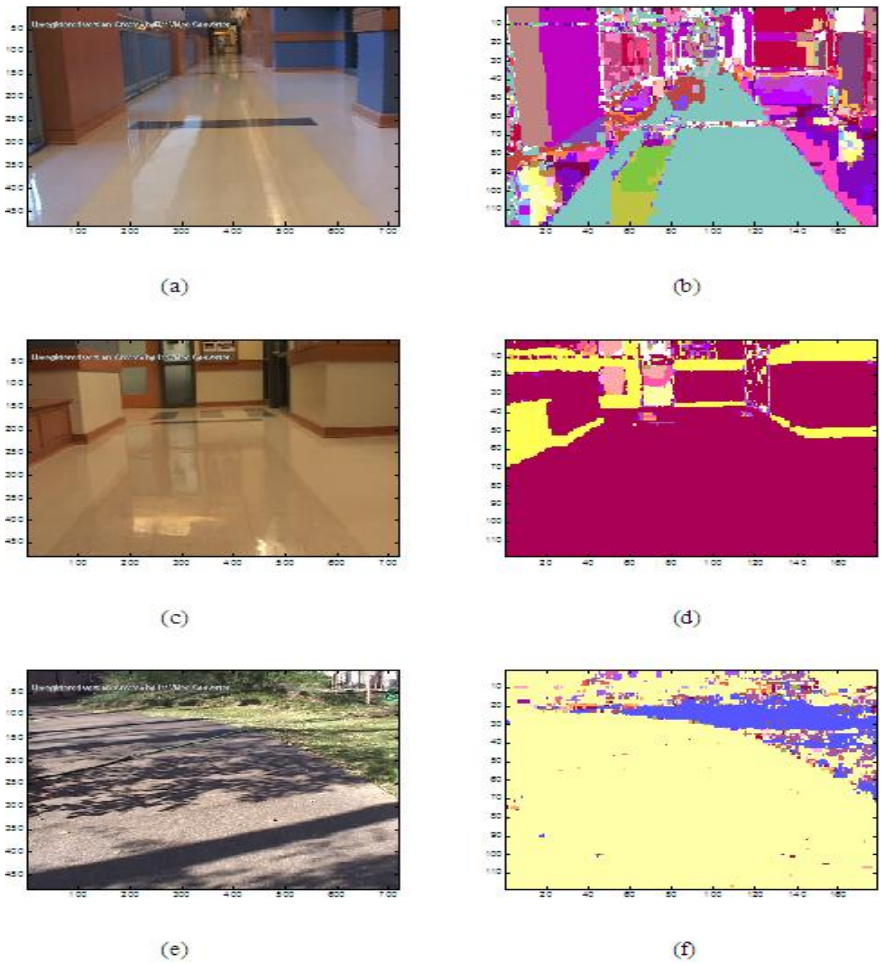


Figure. 4-2. Natural Scene Segmentation Examples; (a) Indoor Atrium, (b) Indoor Atrium Segmentation, (c) Indoor Jacob Hall, (d) Indoor Jacob Hall Segmentation, (e) Outdoor FGH, (f) Outdoor FGH Segmentation [3]

The final step in training the visual system is to organize the clusters found into a more meaningful representation. Reference [3] used a K=3 K-means search tree.

This process resulted in all the feature vectors being assigned to a percept class and organized in an approximate nearest neighbor (a-NN) search tree for image segmentation. This step of creating the tree took about 1.5 days to complete, and thus the technique was clearly not fast enough for practicality. As an alternative, a quasi-maximum likelihood estimation (MLE) tree was created as [4]. The MLE tree is constructed in the same way as the a-NN search tree except for the final leaf nodes of the

tree. In the a-NN tree, many leaf nodes contained training vectors from different classes, and were designated as an impure node. Upon arriving at such an impure leaf node a nearest neighbor search is performed over the vectors in the impure node in order to select the class decision. While the a-NN search tree keeps all of the feature vectors in the impure node for an e-NN comparison, the MLE search tree finds the class with the highest number of feature vectors present in the leaf node. Then during segmentation, if a new vector arrives at that node it will automatically be assigned to the class most represented in that node. During segmentation processing, the MLE tree will be somewhat faster than the a-NN tree.

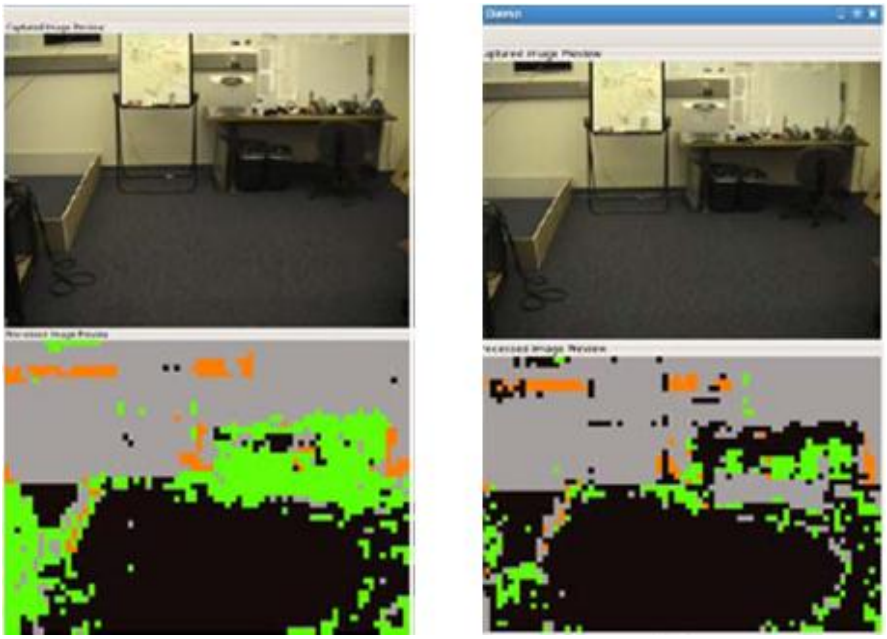
*D.* Image Segmentation

Now that the training data has been classified and organized a new image is ready to be segmented. This process begins with the image being broken down into 15x15 (i.e., N=15) patches and extracting the feature vectors. Each feature vector is then parsed through the search tree with a percept label assigned based on the leaf node reached.

*E.* Image Segmentation Results

This technique has resulted in the segmentation of large objects in natural environments, as shown in Figure 4-2. The environments used are a hallway on the third floor of FGH, Figure 4-2 (a) and (c), and a walkway right outside the building, Figure 4-2 (e). The segmented results are shown in Figure 4-2 (b), (d), and (f). These results show the ability of the methodology used to segment objects, but the problem is still the time required to segment an image. In [4], this was looked at more closely. It was found that using the a-NN search tree to segment six objects in an image required ~12 seconds [4]. This was far too long for any practical real-time vision system. So the MLE search tree

was created to solve this problem. The MLE tree sped the process up to ~5 seconds per image [4]. The segmentation of an image using the a-NN compared the MLE approach is shown in Figure 4-3. Unfortunately, this is still too slow for a real-time system. Therefore, the next step was to



(a) (b)

Figure 4-3. (a) Approximate nearest neighbor search tree segmentation of a lab. (b) MLE search tree segmentation of a lab. [4]

move this system to a general purpose graphical processing unit (GPGPU) such as NVIDIA's CUDA architecture.

*F.* CUDA

CUDA stands for "compute unified device architecture" and was designed by NVIDIA to manage parallel computations [15]. This architecture was designed in a single instruction multiple data (SIMD) format. The only requirement for this parallel architecture to improve a serial process is that the data must have a high arithmetic cost relative to the data fetching cost. This is due to the design of the processor. A typical

CPU has a large cache for data fetching relative to the number of arithmetic logic units (ALU) while the CUDA device was designed with a large number ALUs relative to a small cache. This means there is a relatively small amount of space to store the data in fast memory increasing the cost of having to fetch that data. Because each patch goes through the same set of instructions, in principle the design presented is well suited for this architecture. Although CUDA did help significantly, yielding segmentation times of 50 to 100 milliseconds per image, a number of changes needed to be made to the process in order to run under CUDA. These changes will be explained in the next section.

## Implementation of the System

### *A.* Overview

The goal of this work is to autonomously create a model of the world seen through the view of a mobile robot in a reasonable amount of time. The flowchart for training this system is shown in Figure 4-4.

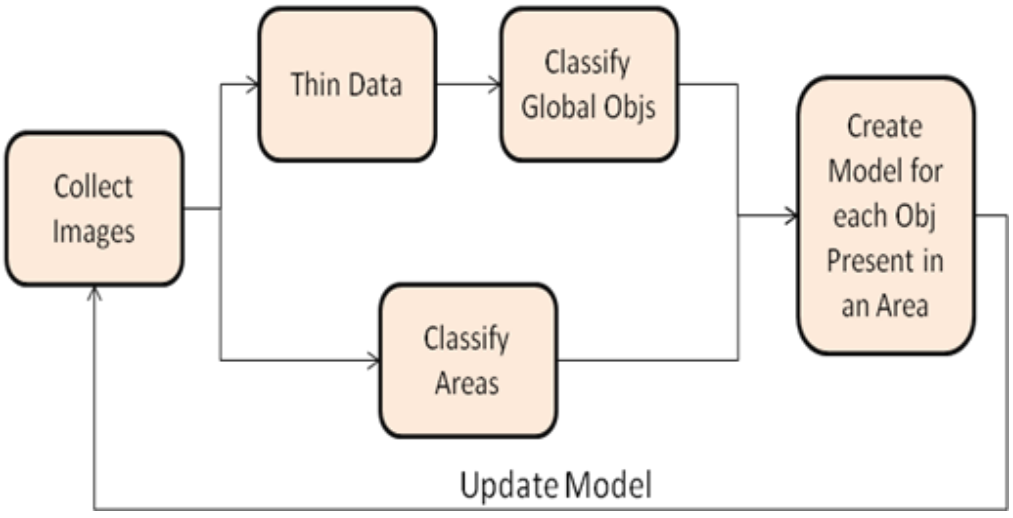



Figure 4-4. Flowchart for training the location recognition system

The process begins by first collecting the training images. Next the database of extracted feature vectors needs to be thinned and the global percepts classified. Then the local areas need to be determined. Once the global percepts and local regions have been segmented the percepts present in each local region are used to model that region.

The data for this work was collected from a video camera mounted on top of a Pioneer 2-AT mobile robot that was guided around the third floor of FGH. Although a wandering algorithm could have been implemented, this was deemed unnecessary instead adopting a philosophy more resembling a human being guided around a building. The path taken is shown in Figure 4-5. The path required about 6 minutes and 20 seconds to travel. From that video 600 evenly spaced images were collected resulting in an image about every foot of travel.

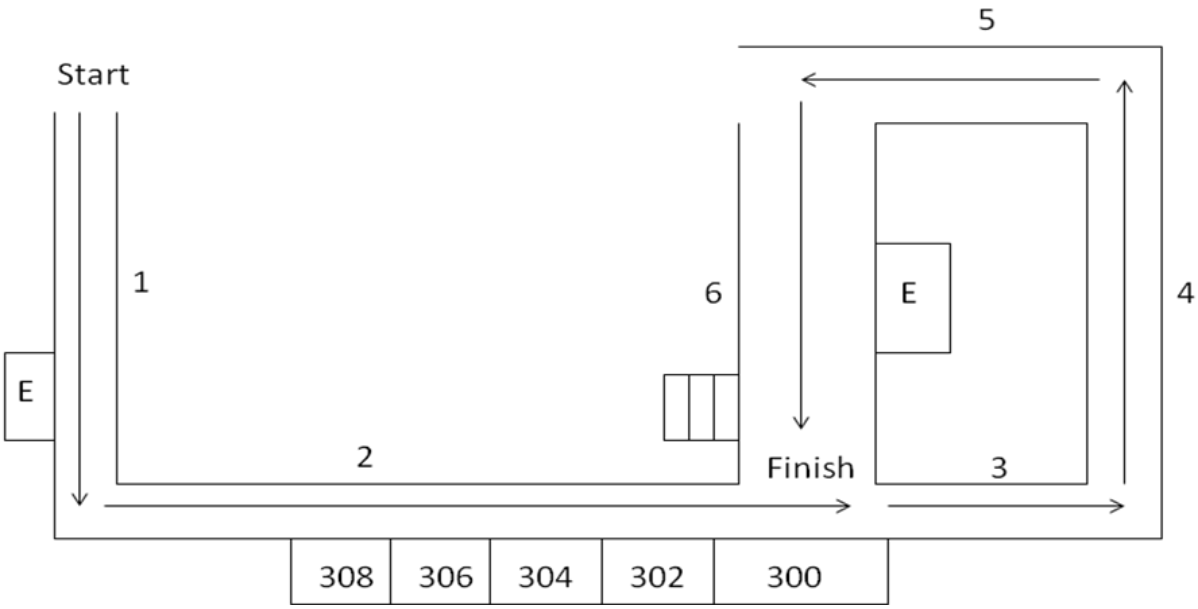



Figure 4-5. A diagram of the path taken by the mobile robot on the third floor of FGH at Vanderbilt University with the numbers providing a label for each hallway. E represents elevators and because this diagram is incomplete some rooms have been labeled for orientation.

This method will allow the system to learn a general understanding of the percepts common throughout the entire floor (i.e., wall color, floor color, etc.) so that when the global region is segmented into smaller more local regions, it will be able to understand that even if a blue colored wall looks slightly different locally in multiple regions, globally it is the same percept. The significance of this relates to the system's ability to connect percepts somewhat like humans do.

*B.* Feature Extraction

The current state of the vision system extracts a 10,000 dimensional HSV color histogram from a 15x15 patch. The texture feature was removed in order to simplify the processing. Also, it was found that by normalizing the feature vectors to have an L2 or Euclidean length of 1, they would retain more separation for segmentation [3].

*C.* Thinning

Because such a high dimensional feature space has not often been used before, the previous works focused on determining whether such a feature space was useful for vision applications. This means that the approaches taken were well known and very thorough in processing the data to provide the best results, rather than the most efficient processing. This is evident in the training phase taking more than 4 days in [3] and each image requiring approximately 5-12 seconds to process depending on the method [4]. Therefore, instead of focusing on the best possible results, this work will focus on getting good meaningful results in the least amount of time.

The first step in training the system is thinning the 2,002,200 feature vectors gathered from the 600 images. The previous method required gathering all of the data and then arduously finding the distances between every feature vector in the database. The

proposed method here will thin the database relative to each image gathered rather then to the entire database.

The purpose of thinning the data is to remove the more isolated feature vectors that do not clearly correspond to an object. Such isolated vectors occur very rarely in the training data and thus it is likely they do not represent robust dominant percepts. In fact, they may be interpreted as distracting noise. Therefore removing such noise from each image as that image arrives, will remove the noise from the overall database. Although this method will not remove as many feature vectors as processing the entire database as a whole, it will still thin the database substantially, keeping only tightly clustered feature vectors.

The first step in thinning each image is to extract the feature vectors and find each feature vector's shortest distance to any other feature vector from that image. Then all the minimum distances are averaged and divided by 20. The value of 20 was found empirically and repeatedly resulted in roughly a third of the feature vectors from each image being kept. This thinning process resulted in about 600,000-900,000 feature vectors retained and required only about 10 minutes total computation time.

*D.* Clustering the Data

The next step in training the system is to cluster the thinned data points. With about 400,000 feature vectors the MST took approximately 40 hours. Therefore, with the thinned database being almost twice as big as that in [3], the processing time of the MST was even less practical. So in order to decrease the processing time a K=40 K-means clustering was used. Because the number of objects is unknown, K=40 is meant to over-segment the data initially. The value of 40 was found empirically to perform well.

Once the number of clusters is set, the K-means algorithm will randomly select K feature vectors and make them the centroids. Next, the remaining feature vectors will be assigned to the cluster that has the centroid that feature vector is closest to. As soon as all the feature vectors have been assigned to a cluster, the mean of each cluster is calculated and the feature vectors are clustered again. This process continues until the clusters converge and the means no longer change.

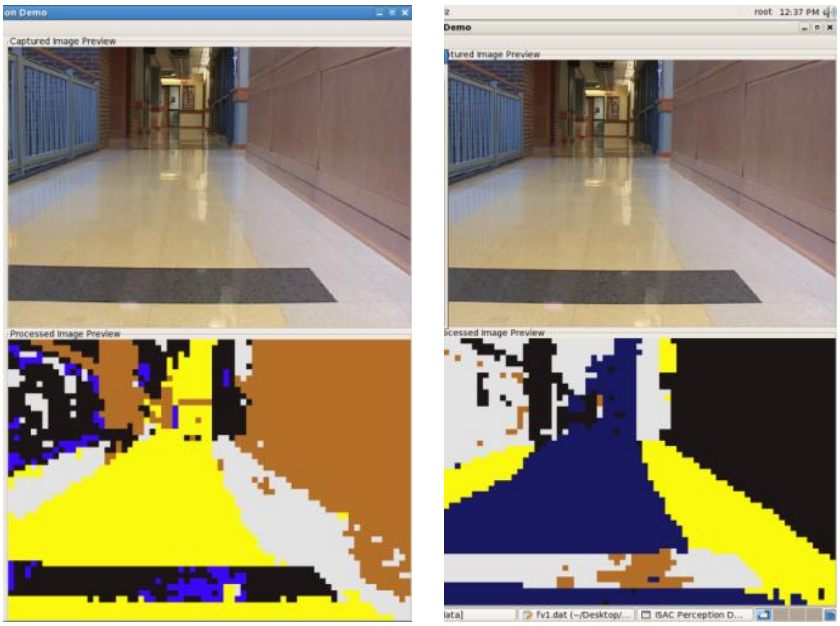
Next, the over-segmented clusters can be merged into a smaller number of significant clusters. This is explained in detail in Section III.F below.

*E.* Modeling the Clusters

At this point in the previous works, the next step would be to create a search tree of the database in order to optimize the structure of the data for processing. However, none of the search trees attempted were fast enough for real-time applications. The solution to this problem was to use CUDA. Because of the memory limitations of the GPGPU, it was not possible to port the entire search tree onto the GPGPU and efficiently access the feature vectors within it. So a nearest mean classification technique was implemented. Nearest mean classification is equivalent to assuming that each percept is modeled with a Gaussian distribution with different means and all covariance matrices equal to an identity matrix. Clearly, this is usually going to be a suboptimal assumption, however, the use of a very high dimensional feature space provides greater separation than low dimensional spaces, and we found the nearest mean approach to yield both efficient and good quality results.

The nearest mean classification calculates the mean vector of each of the clusters and uses that as a model of that percept. Then when a new feature vector is being

processed, it will be classified according to the percept mean vector that it is closest too. An example of an image segmented using the nearest mean technique compared to the same image segmented using an exact NN classification is shown in Figure 4-6.




(a) (b)

Figure 4-6. (a) An image of the hallway using an exact nearest neighbor classification. (b) The same image of the hallway using nearest mean classification. Note: The labels in the nearest mean were assigned randomly. The clusters represented were checked and found to be the same as the exact nearest neighbor.

In this example, the database gathered and labeled for Figure 4-6 (a) was done by a human user. The feature vectors representing each cluster were then averaged and those percept mean vectors were assigned a random label. Because of the random labeling algorithm used for display purposes, the percepts in Figure 4-6(b) appear in different colors, but the clusters that represent each object are the same.

*F.* Combining the Clusters

When the K clusters mean vectors have been found, some clusters needed to be merged or removed due to the over-segmentation. The first means of reducing the number of mean vectors is a simple threshold. If a cluster does not have at least 1,000

feature vectors represented by the mean vector, that mean vector is deleted. This is done to remove small clusters of feature vectors that do not occur very often in the training data. The interpretation is that these clusters do not represent dominant percepts. The value of 1000 was found empirically and represents a very tiny fraction of the thinned database of 600,000 to 900,000 vectors.

Next, the distances between all the remaining mean vectors relative to each other are found. If a mean vector is orthogonal to all the other mean vectors it is automatically kept as a distinct cluster. All the other mean vectors that do not have another mean vector less than a distance of 1 to it, are also kept. Those mean vectors that have other mean vectors that are < 1 from them are merged by averaging with the mean vectors within that distance. The distance of 1 was experimentally found to provide 16-19 percepts for the global area which is roughly the same as the human user found prior to this technique.

This method takes about 10-15 minutes to provide the mean vectors of the large objects where the MST approach reported in [3] took about 40 hours.

*G.* Processing an Image

When segmenting an image, the goal is to find the nearest percept mean vector to each feature vector computed from the patches in the image. The mean vectors are represented by $x_i$ for i=1, …, M, thus M designates the number of percepts. The vector $y_p$ is the feature vector from the $p^{th}$ patch in the image. The minimum exact distance, $D_{p,}$ between the set of normalized mean vectors $\{x_i\}$ and the normalized feature vector $y_p$ would be calculated as follows:

$$D_p = \min_i \left( \sqrt{\|x_i\|^2 + \|y_p\|^2 - 2(x_i \cdot y_p)} \right) \text{i=1,…,M} \qquad (4.1)$$

Since the feature vectors have been normalized, $\|\mathrm{x}_i\|^2 + \|y_p\|^2 = 2$. This leaves the dot product as the main computation remaining. Based on this equation, finding the largest dot product will result in the minimum distance. Therefore (4.2) is equivalent to the distance calculation used:

$$d_p = \max_i(x_i \cdot y_p) \tag{4.2}$$

The sparse vector representation, nearest mean criterion and dot product computation resulted in greater efficiency, significantly less storage space on the GPGPU, and far fewer memory fetches. In fact, the entire dot product calculation is based on only the indices in $x_i$ and $y_p$ where both elements in the vectors are nonzero. This means that only these nonzero elements need be multiplied.

The time results based on this implementation improved the system greatly. As reported, previously it took 12 seconds and 5 seconds for the a-NN and MLE search trees respectively [4], the parallelized GPGPU system was able to segment images at ~10-20 images/second (i.e., 50 to 100 msec). Whereas in the search trees the time was based on how deep the tree was and how many feature vectors exist in a leaf node, the new system is dependant on how many potential percepts are present and how many nonzero indices there are in a feature vector. For the estimated times reported there were between 12-18 objects in any given segmentation.

*H.* Finding the Local Regions

Now that the global percept mean vectors have been found the next step is to classify the large global area shown in Figure 4-5 into smaller areas that are determined by the system. This is done by comparing what the system currently sees to what it saw in its recent past. If the change in the environment is significant, a border will be created.

In order to do this, first assume that N (N = 600 in this case) images have been gathered and Q (Q= 3337 in this case) feature vectors have been extracted from each image. Then each image is represented as $I_n$ (n = 1,…, N), and each feature vector is represented as $\underline{x}_{n,q}$ (q = 1,…, Q). Now the average perceptual change $r_n$ between image $I_n$ and images $I_m$, where m represents the indices of four images that occurred prior to $I_n$ (m = {n-10,n-15,n-20,n-25} in this case), can be calculated.

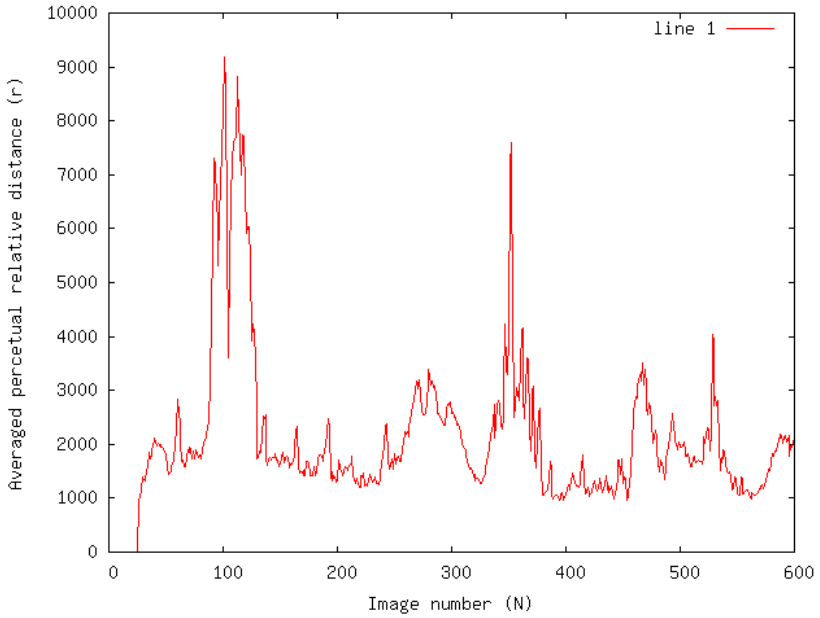
The first step in finding $r_n$ is to find the individual relative perceptual distances $\rho_{n,m}$ between $I_n$ and each of the four prior images used. This value is calculated as the symmetrical sum of the sum of distances between each feature vector in $I_n$ and its closest feature vector in $I_m$ and the sum of distances between each feature vector in $I_m$ and its closest feature vector in $I_n$. This is shown in (4.3).

$$\rho_{n,m} = \sum_{q=1}^{Q} \min_{q'} \left\| \underline{x}_{n,q} - \underline{x}_{n,q'} \right\| + \sum_{q=1}^{Q} \min_{q'} \left\| \underline{x}_{m,q} - \underline{x}_{n,q'} \right\| \quad (4.3)$$

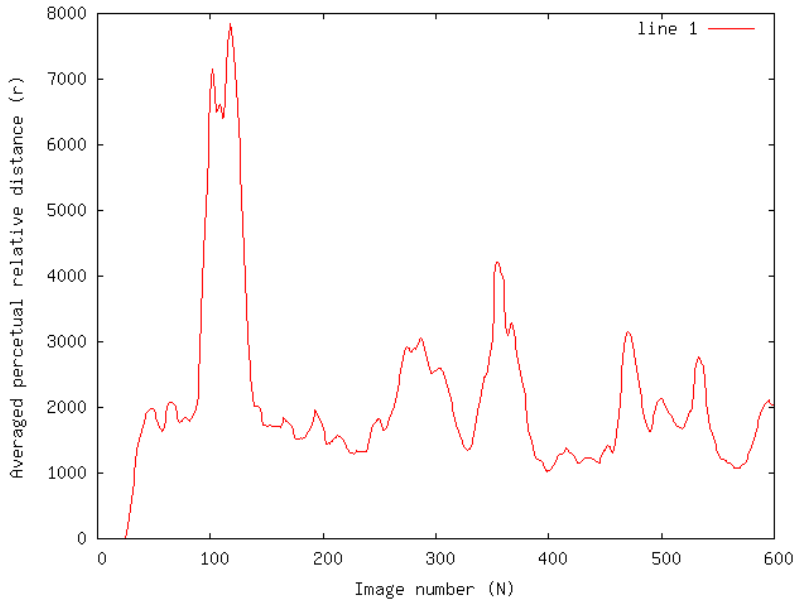
Once $\rho_{n,m}$ has been found for each value of m they are averaged together to find the average perceptual change between $I_n$ and each image in $I_m$. This is shown in (4.4)

$$r_n = \frac{1}{4}\left(\rho_{n,n-10} + \rho_{n,n-15} + \rho_{n,n-20} + \rho_{n,n-25}\right) \quad (4.4)$$

The results of (4.4) are plotted in Figure 4-7(a). The horizontal axis indicates the image number and the vertical axis is $\rho_{n,m}$. The plot is smoothed using an averaging filter, and only the peaks in areas of the plot that have an average perceptual change that is trending up followed by a downward trend are required. This results in the image shown in Figure 4-7 (b).




(a)




(b)

Figure 4-7. (a) A plot of the average perceptual change over 600 images. (b) (a) after being smoothed by an averaging filter.

Figure 4-7 (b) shows multiple large peaks representing different points of change. The first set of peaks at roughly image 110 represents the left turn going from area 1 to area 2 in Figure 4-5. This change is very sharp because of the turning motion performed. The changes at images ~360, 475, and 535 are from turning as well. The change at image

~280 was due to passing from an area with a sky light into a hallway with no natural light sources. In a separate experiment, the system was driven from a hallway straight into another room and the context change from the hallway to the room was enough to indicate a new environment. These results were consistent over multiple trials, indicating this method can effectively detect context changes in this environment.

The current problem using this method is finding the best way to determine which image is the cutoff point, or border, between two regions. Currently, the criterion for creating an area is if there are 10 images with consecutively rising distances prior to the current image and 10 images with consecutively falling distances after the current image then that peak image is the cutoff. This technique does a reasonable job of locating the peaks, but is not robust enough to work without some minimal user assistance.

*I.* Modeling Local Regions

The final step for training this system is to create the local models. This is done by calculating the mean vectors of the percepts that are local to each area. These are called the local percept mean vectors. To do this, first assume that there are M percepts, J local regions and that the global percept mean vectors are represented as $\underline{x}_i^{(g)}$, where i= 1,…,M. Then the local percept mean vectors are represented as $\underline{x}_{i,j}^{(l)}$, where j=1,…,J. Next, the local percept mean vectors, $\underline{x}_{i,j}^{(l)}$, for an area are found by first segmenting all the images found to be in that area using the global percept mean vectors, $\underline{x}_i^{(g)}$ . Then all the feature vectors, from all the images in an area that are associated with each percept M, are averaged. These averaged vectors are then used as the local percept mean vectors modeling an area.

*J.* Recognizing Locations

Now that the system has a model of each area, we are ready to segment new images and determine the area the robot is in at each new image. The flowchart to explain this process is shown in Figure 4-8.

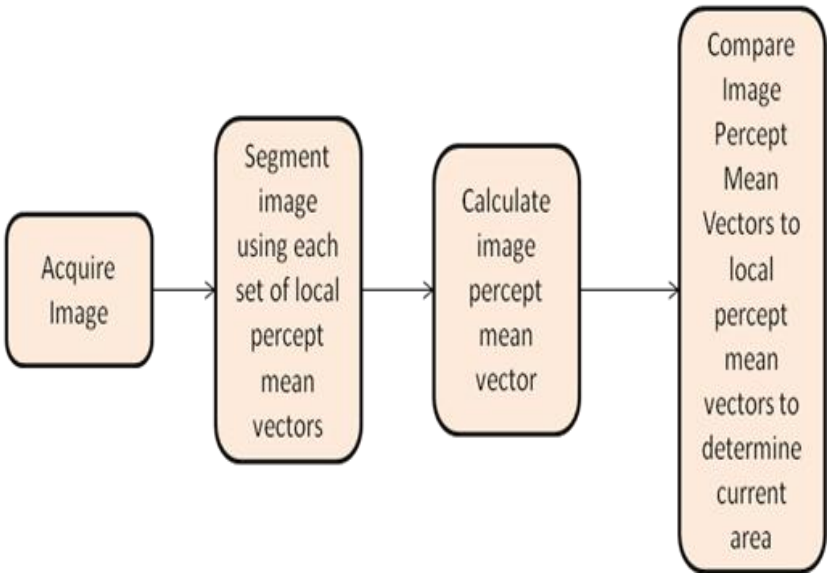



Figure 4-8. Flowchart for processing each new image seen by the system.

This process begins with simply acquiring an image from the camera. That image is then segmented for each assumed area j' using each assumed area's local percept mean vectors, $\underline{x}_{i,j'}^{(l)}$. In the case of the hallway described previously J=6 areas. After the segmentations are performed the feature vectors associated to each percept are averaged. This step provides J new sets of mean vectors for the current image called the image percept mean vectors, $\underline{y}_{i,j'}$. The new mean vectors represent what the system believes it sees if the robot is in the area of the assumed local percept mean vectors used for the segmentation. The six new sets of image percept mean vectors are then compared to $\underline{x}_{i,j'}^{(l)}$. The distance metric used is the sum of the distance of each percept in $\underline{y}_{i,j'}$ to its respective

percept in $\underline{x}_{i,j'}^{(l)}$. Whichever area, j' , provides the shortest local area distance $A_{j'}$, is determined to be the current area. The equation for $A_{j'}$ is shown in (4.5).

$$A_{j'} = \sum_{i=1}^{M} \left\| \underline{x}_{i,j'}^{(l)} - \underline{y}_{i,j'} \right\| \quad (4.5)$$

Obviously, it is very desirable to have a system that can be easily updated after the initial training. The means of currently updating the training data base is to collect a new set of images and create new local percept mean vectors for that set of images. Finally, average the new set of local percept mean vectors with the current set of local percept mean vectors.

## Explanation of Results

### *A.* Processing Time Results

The reason very high dimensional feature spaces are not used frequently is because of the cost of processing high dimensional data. The amount of data quickly overwhelms the CPU and renders solutions far too slow for practical use in both training the system and processing new data. This system offers a solution to that problem. By expanding the dimensionality to such a high level, most of the mathematical complexity other systems need to use for segmentation become infeasible. Therefore, simple methods with proper implementation, such as GPGPU, have been able to largely reduce the time issues.

The first place this can be seen is in the training phase. Table 4-1 shows the processing times comparing the training phase of the systems presented. The total time to train the system in [3] is ~100

Table 4-1: Training phase times

| Training Phase | Time (hrs) |
|---|---|
| Previous System | |
| Thinning | ~24 |
| MST | ~40 |
| Creating Search Tree | ~36 |
| Total | ~100 |
| Current System | |
| Thinning | ~0.16 |
| K-means | ~0.16 |
| Total | ~0.32 |

hours or ~4 days while the current implementation requires ~0.32 hours or ~ 20-25 minutes. This is a significant increase in training time that comes without significant loss of segmentation quality, with quality defined as providing information to the system rather then aesthetics to humans.

The second improvement regarding processing time is seen in the image segmentation phase. The processing times comparing image segmentations are shown in Table 4-2. The previous methods using a-NN

Table 4-2: Image segmentation times

| **Technique** | **Time (msec)** |
|---|---|
| Approximate Nearest Neighbor | 12,000 |
| MLE | 5,000 |
| Nearest Mean | 50-100 |

and MLE required 12 seconds and 5 seconds respectively segmenting an image. This is not fast enough for any system to operate in real time. By changing the model of the data to mean vectors and implementing the distance calculations using CUDA this time was reduced to 50-100 milliseconds. It is also believed that optimizing the implementation on CUDA will improve this processing speed even more.

*B.* Image Segmentation Results

Upon first inspection of all the segmented images in Figure 4-9, it would be perfectly reasonable to conclude that the segmentations retain little if any useful information. However that is not the case.

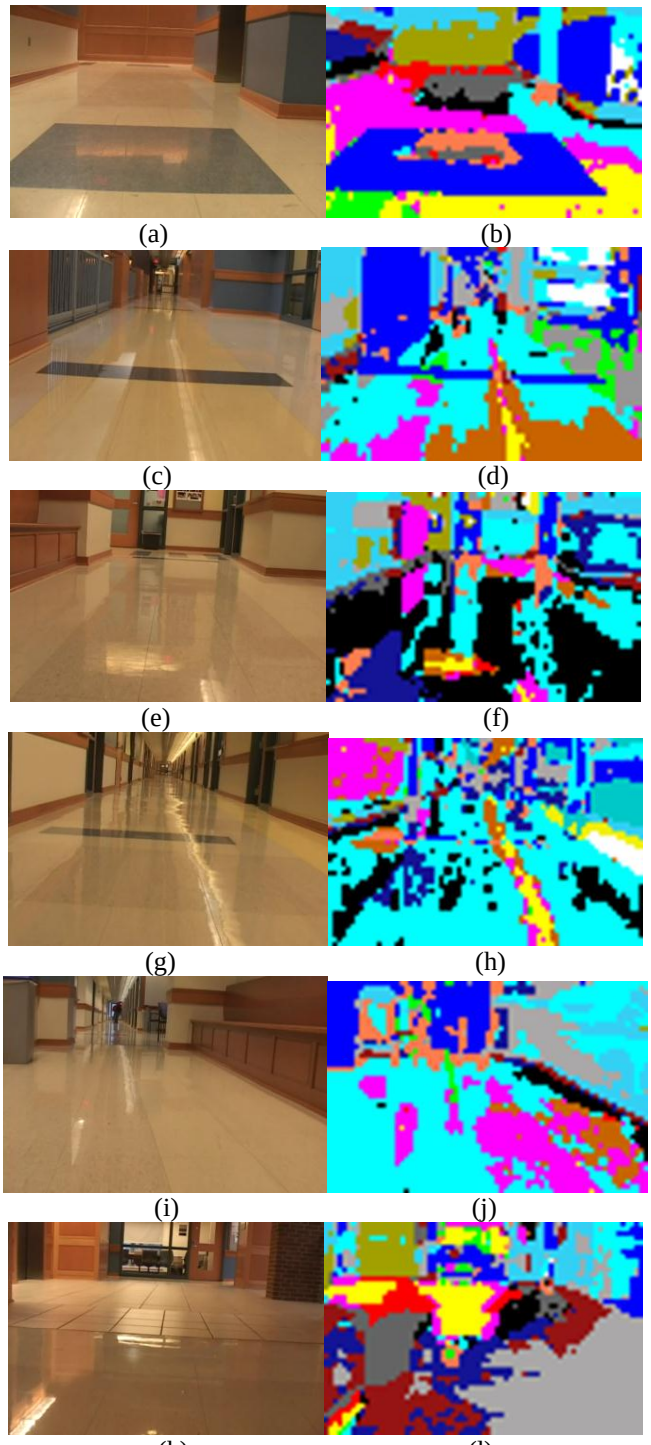


Figure 4-9. (a) image of hallway 1 in FGH. (b) segmented image of (a). (c) image of hallway 2. (d)segmented image of (c). (e) image of hallway 3. (f) segmented image of (e). (g) image of hallway 4. (h) segmented image of (g). (i) image of hallway 5. (j) segmented image of (i). (k) image of hallway 6. (l) segmented image of (k)

Although these segmentations do not directly reflect the original image, they do retain a good amount of information about the original image. Comparing the segmented images of Figure 4-9 to those in Figure 4-2, Figure 4-9's images do not appear to be segmented as cleanly. There are two reasons for this. The first is that the patch size in Figure 4-2 is 7x7. This means those images are segmented at a much higher resolution and single errors do not stand out as much. This resolution is not used here because high resolution information is not necessary for location recognition and increases the number of feature vectors per image from 3337 to 21,004. The second reason is because the number of cuts in the MST was selected by the human user in the case of natural environments. The autonomous method worked very well for the human motion segmentation work in [3], but when tried in a natural environment the algorithm under-segmented the images. The clustering method presented here required no human interaction. It should be noted that the reason the proposed system's segmentation is not aesthetically pleasing is that the K-means clustering selects different percept clusters from those selected under the MST clustering technique. However, the K-means derived clusters still contain a great deal of useful information and result in much more efficient processing.

The real significance of this work is what the segmented images mean and what information they provide. Frequently when images are segmented, the goal is to find very specific objects that stand out from the environment. In this case the goal is to recognize the environment itself. Following that approach, it was shown in [2] that this system is capable of novel object detection. By observing the entire environment the novel object would then stand out as what it does not know, i.e., it doesn't belong to the known percepts. Then once the object is learned, it too will be recognized and used as

necessary, whereas an approach of looking specifically for that object disregards all the other information from the world around the robot.

Another implication of this type of processing is the relation of objects that are classified as the same but are not actually the same singular object. An example of this would be the white wall seen in Figures 4-9 (e) and (g). Although they are both white wall, they are not both the same white wall. As it stands, with just the global percept mean vectors they are both considered the same thing. However the single global percept will develop local percept models of each object present thus providing the system with the philosophical understanding that even though these are the same object they may appear differently in different environments.

*C.* Location Recognition Results

The results for the location recognition are presented in Table 4-3. These results were obtained from a

Table 4-3: Results of Location Recognition on Untrained Images

| | 1 Training Set % Correct | 3 Combined Training Sets % Correct | 4 Combined Training Sets %Correct |
|---|---|---|---|
| Area 1 (97 images) | 74% | 91.8% | 92.8% |
| Area 2 (183 images) | 59.3% | 86.8% | 92.9% |
| Area 3 (69 images) | 52.9% | 91.2% | 91.4% |
| Area 4 (101 images) | 96% | 98% | 98% |
| Area 5 (66 images) | 3% | 50.8% | 69.2% |
| Area 6 (84 images) | 0% | 71.1% | 53% |
| Overall | 48.6% | 81.6% | 83.3% |

series of untrained images extracted from a video of the hallways. Four such videos were made over the same area and on different days, resulting in four training sets. Training sets 2 through 4 are used to update the system initially trained on set 1. As seen in Table

4-3 the system is able locate itself with respect to autonomously predetermined areas. The performance in areas 1-4, after updating the database, indicates that the system is capable of recognizing what local area it is present in based solely on the percepts that are found in that area.

The poor performance in areas 5 and 6 relative to the rest of the areas was due to the initial training. The first video was taken on a cloudy day and both area 5 and 6 have part natural lighting and part non natural light. The result of the clouds reduced the size of the peaks in Figure 4-7(b) at images ~500 and 600. When the other training videos were taken there were prominent peaks at those locations indicating that areas 5 and 6 should have been split into two more areas. Because the system is not currently capable of handling such a situation on its own at the current time the decision was made to keep the results as they are and use this event to devise a means of dealing with this situation as part of future work.

## Conclusion and Future Work

This work has shown an efficient means of representing the world for a mobile robot. With the model created, the robot is able to quickly segment the world around it and use that information for useful purposes. The models of the objects in the world are 10,000 dimensional pdfs, or histograms, of the HSV color space mean vectors. These are then used to segment a new image using nearest mean classification. The results showed practical segmentations and significantly improved processing times.

This model is then used to semi-autonomously partition the hallways in Figure 4-5 into 6 smaller regions, develop local percept models of the global percepts, decide

which of the 6 smaller regions the robot is currently located in, and incrementally learn more about each area of the hallway, with each new pass through it improving the location recognition system.

The next step to be taken in this work will be to implement methods to detect both novel objects and novel areas. This will include using the method of novel object detection from [2] and [4], detecting completely novel areas, detecting a new area even if the percepts are recognized (i.e., further extensions of the hallways in Figure 4-5 that complete the entire floor plan), and determining if the current model of the region is the best model. Once this work is completed the system will then be able to learn and recognize more areas.

Also a few more additions will be made with respect to the vision system. Obviously understanding of functionality can not come through vision alone. However recognizing behaviors through vision is possible. With this in mind the system will be able to start recognizing what it believes to be reflections based on their behavior as the robot travels. Also when objects are seen at a distance, due to various lighting conditions, they can be segmented incorrectly. Therefore a method of tracking how areas are segmented is being developed in order to track whether or not mean percept vectors should be combined, split, or marked as similar.

# CHAPTER V

## Manuscript II: Percept Classification, Novel Object Detection, and Novel Area Detection for a Vision Based Developmental Location Recognition System

Christopher Costello and D. Mitchell Wilkes

Vanderbilt University

Nashville, TN

To be Submitted as a Regular Paper to the

IEEE Transactions on Autonomous Mental Development

## Abstract

This paper will focus on two new aspects of the vision system described in [59] which focuses on creating a developmental location recognition system. The first addition is a further classification of the autonomously learned percepts. Because they were learned in an unsupervised manner through vision alone, it is necessary for the system to error check itself and determine if the segmented percepts are in fact actual percepts. This includes deciding if a percept is a reflection, a different aberration of light, or an actual percept. This stage of processing will be done by tracking the percepts as the robot moves and classifying each percept blob based on its behavior. The second addition presented here will be a means of detecting both novel objects and novel regions. The implementation of the novel object detection will be similar to the work in [2,4], but has been modified to only detect objects of a certain size. The novel area detection will use a threshold and consistency measure to determine if an area should be considered a novel region. This method resulted in accurately reducing previously learned regions into more accurate regions, detecting novel regions on the same floor as the initial training, and detecting novel regions in different areas.

## Introduction

Numerous location recognition systems have been and are currently being developed [6,7,13,18,21,26,32,33,38]. As described in [59], three general types of location recognition used are SLAM, landmark detection, and template matching. These types of systems have all had success to varying degrees, but still lack certain aspects of location recognition that are important. SLAM has been found to be extremely useful,

and will be added to this vision system as a future work, but SLAM by itself is entirely based on geometric shape. Because of this numerous visual SLAM techniques have been developed [13,18,19], and although these systems tend to use vision to improve the exact location of the robot relative to its surroundings, a system using SLAM that aims to provide a general description of the robots location, e.g. by the elevator, was not found. Landmark detection has also had a good amount of success. This technique involves using specific objects or qualifying features to define a region. The problems with landmark detection has been selecting robust enough landmarks and finding those landmarks in application [13,35]. Also landmark detection has been largely used for navigation purposes versus defining an actual location. Template matching is the matching of a current image to a set of training images and, based on some set of features, determining the location of the robot. This technique has been used to define areas through the use of epitomes [21]. The issue with this method is that it does not provide any type of percept segmentation.

With these works in mind, the goal of this location recognition system is to create a system that is capable of segmenting the percepts in an image and, based on the percepts observed, determine what area the robot is currently in. The vision system used was created in [1], and has been used for multiple works [2,3,4,5]. This vision system has shown the ability to reliably segment trained percepts using both supervised and unsupervised training methods.

As discussed in [59], the problem with the previous unsupervised methods was the length of time required to train the system. Therefore a fast K-means classification with K reduction was performed in [59]. This resulted in segmentations that were useful

for location recognition, but were not very good at object segmentation. Therefore this work will also include methods of further classifying the percepts segmented by determining whether a reflection is present, as well as determining if segmented percepts are actual objects, aberrations of light, or if the system is uncertain whether it sees an object or not. This will be done by tracking the percepts as the robot moves and classifying them based on their behavior.

The location recognition system introduced in [59] was able to autonomously learn the percepts within a training region, segment the training region into local regions, create models of the local regions based on the percepts recognized from training, and finally visually recognizes each local region with a high degree of reliability based on the local models. Now that the system is able to learn an initial set of regions, it needs to be able to recognize if it is in a new region. This process involves multiple aspects.

First, the system must be able to determine if novel objects are present. A novel object detection scheme was introduced in [2,4], and will be included in this work. Next, the system needs to be able to determine if it is in a region that it recognizes that simply has a novel object present. Then, the system needs to determine if it is in a novel region. There are three types of novel regions that need to be determined. The first type of novel region is determining if a known region was originally trained incorrectly and should have been learned as two separate regions. In this case, the system should then separate the known region into two new regions and recognize them as such. The second case is a novel region that has the known percepts present, but is in an untrained area. This could be different hallways in the same building. The final type of novel region that needs to be recognized is a truly different area, such as a different building.

The implementation of these additions and the results will be discussed in the following sections. The rest of the paper will be organized as follows. Section II will describe the location recognition system and provide all of the background information necessary. Section III will explain how the reflection detection and percept classifications were implemented and the results. Section IV will explain the novel object detection and novel area detection scheme along with their results. Section V will give conclusions as well as future works.

## Related Works

### *A.* Vision System

The vision system used in [59] uses color information extracted from patches of 15x15 pixels. The feature vectors used are 10,000 dimensional feature vectors representing the pdf, or the histogram, of the hue, saturation, and value (HSV) color space present in each patch of pixels. The primary focus is on the true colors present in each patch and therefore the hue is quantized into 100 bins with the saturation and value being quantized into 10 bins each. As stated in [1] the high dimensionality is used because of the robustness it offers for classifying percepts.

Using this level of dimensionality presents its own challenges. First, gathering enough training data. Second, all $2^{nd}$ order calculations have become extremely expensive and difficult to calculate. Third, the size of the data can quickly overwhelm the memory of a system. Finally, segmenting the images at frame rate speeds is difficult.

The first issue is discussed more in [1], and in summary there is no issue. Images are full of information. So much so that systems often try to limit the information used.

Therefore, collecting enough information for properly training the system is not an issue for this work.

Next, due to the size of the feature vectors calculations such as covariance matrices, Eigen values, Eigen vectors, etc. become impractical. This is dealt with by using a Euclidian distance measure and a nearest mean classification. Although distance measures in such high dimensionality are known to have problems [23,24], they were not largely experienced in this system [63].

The size of the feature vectors was dealt with by using a sparse representation of the data. Because the information was extracted from a patch of 15x15 pixels, in the worst case scenario a sparse vector would have 225 bins. Therefore using this method affords the segmenting power of a 10,000 dimensional space while only having to deal with vector sizes of 225 nonzero bins. It is worth noting that most feature vectors have only 30-40 bins reducing the memory requirements even more. For more information about the benefits of this space see [1,2,5,63].

As far as processing the data, frame rate image segmentation has been the most challenging aspect of the high dimensionality. Multiple attempts have been made including nearest neighbor (NN) searches, approximate nearest neighbor (a-NN) search tree, and maximum likelihood estimation (MLE) search tree [5]. None of these methods were able to reduce the segmentation times to the desired level. As described in [59], a nearest mean approach has been implemented on the CUDA architecture which has sped the image segmentation up to acceptable speeds while retaining acceptable quality.

### *B.* Percept Classification

The training database of feature vectors for the vision system has been clustered using three different methods. The first method used supervised learning [1,4]. This method provided robust and reliable results. Because of the success of the supervised method, unsupervised techniques were implemented. The first unsupervised clustering technique used was a minimum spanning tree [2,3]. This technique also resulted in robust and reliable segmentations. The problem with this technique was that it took days to train the system and if a problem arose the training had to be restarted. So because of the time required to get the results, the decision was made to see how the system would perform using a simpler and faster clustering method, K-means [59]. This method was implemented as follows.

After the feature vectors were extracted from the images, the database was thinned to be more manageable. Then a K = 40 K-means classification was performed on the data. This resulted in over segmenting the database. As a result the mean feature vectors that were close to each other in the feature space were combined to represent a single percept. This resulted in a single mean percept vector representing each percept classified. The mean percept vectors were then normalized resulting in a set of normalized mean percept vectors.

When segmenting an image, the goal is to find the nearest percept mean vector to each feature vector computed from the patches in the image. The mean vectors are represented by $x_i$ for i=1, …, M, thus M designates the number of percepts. The vector $y_p$ is the feature vector from the $p^{th}$ patch in the image. The minimum exact distance, $D_{p,}$ between the set of normalized mean vectors $\{x_i\}$ and the normalized feature vector $y_p$ would be calculated as follows:

$$D_p = \min_i \left( \sqrt{\|x_i\|^2 + \|y_p\|^2 - 2(x_i \cdot y_p)} \right) \text{i=1,…,M} \quad (5.1)$$

Since the feature vectors have been normalized, $\|x_i\|^2 + \|y_p\|^2 = 2$. This leaves the dot product as the main computation remaining. Based on this equation, finding the largest dot product will result in the minimum distance. Therefore (5.2) is equivalent to the distance calculation used:

$$d_p = \max_i (x_i \cdot y_p) \quad (5.2)$$

For further explanation please refer to [59]. Examples of the resulting segmentations can be seen in Figure 5-4.

Accepting that the clustering was going to yield errors meant that methods of dealing with such errors would become necessary. It is accepted that without being able to truly interact with the environment the robot would be limited in how much it could visually correct, but a method of classifying each percept has been developed.

Because of the simplicity of the clustering, reflections and other aberrations had a great deal of influence on the results. Therefore schemes for classifying these events were focused on. There does not exist a lot of research in this type of classification. Therefore, the goal of furthering this vision system has been divided into two tasks, reflection detection and determining if a percept is actually a percept or light.

### *C.* Location Recognition System

As mentioned, numerous methods of location recognition exist. What makes the method presented in [59] different is that it focuses on visually defining the environment based on the overall percepts present. It also autonomously defines each local region based on overall visual differences in the images.

For a full description of the system see [59]. To summarize the location recognition system it begins with first finding the global percept mean vectors. This is done as described in section B. The next step is to divide the global region into local regions. The segmentation of the area is performed using a relative perceptual difference measure, $r_{n,}$ as follows copied from [59]:

In order to do this first assume that N (N = 600 in this case) images have been gathered and Q (Q= 3337 in this case) feature vectors have been extracted from each image. Then each image is represented as $I_n$ (n = 1,…, N), and each feature vector is represented as $\underline{x}_{n,q}$ (q = 1,…, Q). Now the average perceptual change $r_n$ between image $I_n$ and images $I_{m,}$ where m represents the indices of four images that occurred prior to $I_n$ (m = {n-10,n-15,n-20,n-25} in this case), can be calculated.

The first step in finding $r_n$ is to find the individual relative perceptual distances $\rho_{n,m}$ between $I_n$ and each of the four prior images used. This value is calculated as the symmetrical sum of the sum of distances between each feature vector in $I_n$ and its closest feature vector in $I_m$ and the sum of distances between each feature vector in $I_m$ and its closest feature vector in $I_n$. This is shown in (5.3).

$$\rho_{n,m} = \sum_{q=1}^{Q} \min_{q'} \|\underline{x}_{n,q} - \underline{x}_{n,q'}\| + \sum_{q=1}^{Q} \min_{q'} \|\underline{x}_{m,q} - \underline{x}_{n,q'}\| \tag{5.3}$$

Once $\rho_{n,m}$ has been found for each value of m they are averaged together to find the average perceptual change between $I_n$ and each image in $I_m$. This is shown in (5.4)

$$r_n = \frac{1}{4}\left(\rho_{n,n-10} + \rho_{n,n-15} + \rho_{n,n-20} + \rho_{n,n-25}\right) \tag{5.4}$$

An averaging filter is then applied to $r_n$ and the resulting peaks in the data are used to define the lines between regions. Figure 5-1 shows how part of the 3rd floor of Featheringill Hall at Vanderbilt University was segmented.

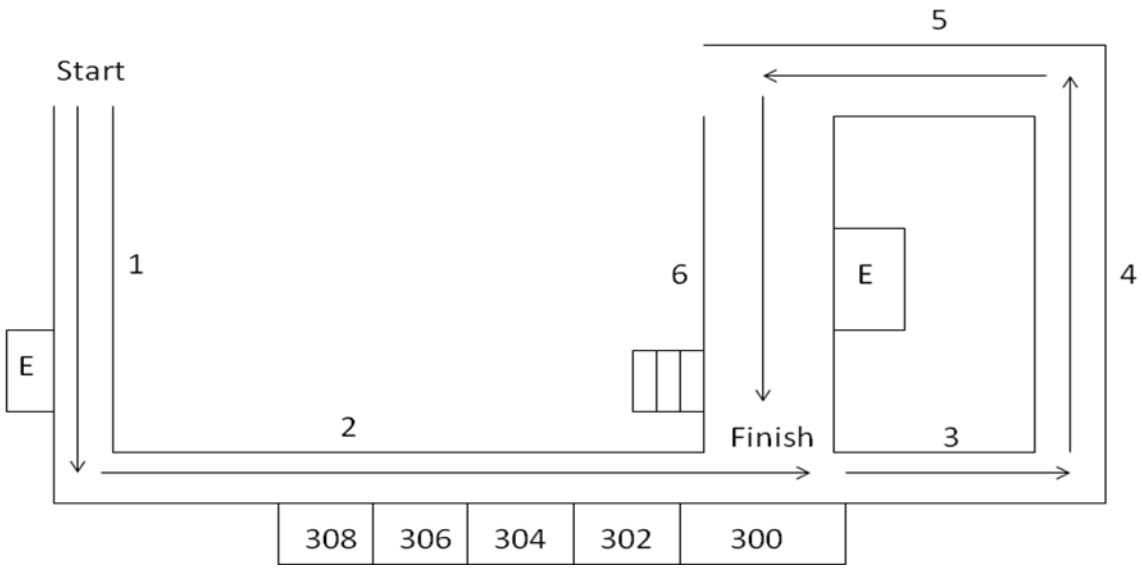



Figure 5-1. A diagram of the path taken by the mobile robot on the third floor of Featheringill Hall at Vanderbilt University with the numbers providing a label for each hallway. E represents elevators and because this diagram is incomplete some rooms have been labeled for orientation. [59]

The final step for training this system is to create the local models. This is done by calculating the mean vectors of the percepts that are local to each area. These are called the local percept mean vectors. To do this, first assume that there are M percepts, J local regions and that the global percept mean vectors are represented as $\underline{x}_i^{(g)}$, where i= 1,…,M. Then the local percept mean vectors are represented as $\underline{x}_{i,j}^{(l)}$, where j=1,…,J. Next, the local percept mean vectors, $\underline{x}_{i,j}^{(l)}$, for an area are found by first segmenting all the images found to be in that area using the global percept mean vectors, $\underline{x}_i^{(g)}$ . Then all the feature vectors, from all the images in an area that are associated with each percept M, are averaged. These averaged vectors are then used as the local percept mean vectors modeling an area.

With the models of the local regions generated the next step is to recognize the locations that each image was taken from. This is done as follows:

This process begins with simply acquiring an image from the camera. That image is then segmented for each assumed area j' using each assumed area's local percept mean vectors, $\underline{x}_{i,j'}^{(l)}$. After the segmentations are performed the feature vectors associated to each percept are averaged. This step provides J new sets of mean vectors for the current image called the image percept mean vectors, $\underline{y}_{i,j'}$. The new mean vectors represent what the system believes it sees if the robot is in the area of the assumed local percept mean vectors used for the segmentation. The six new sets of image percept mean vectors are then compared to $\underline{x}_{i,j'}^{(l)}$. The distance metric used is the sum of the distance of each percept in $\underline{y}_{i,j'}$ to its respective percept in $\underline{x}_{i,j'}^{(l)}$. Whichever area, j' , provides the shortest local area distance $A_{j'}$, is determined to be the current area. The equation for $A_{j'}$ is shown in (5.5).

$$A_{j'} = \sum_{i=1}^{M} \left\| \underline{x}_{i,j'}^{(l)} - \underline{y}_{i,j'} \right\| \tag{5.5}$$

The next step for this location recognition system to take is to be able to handle novel objects and novel regions.

*D.* Novel Object Detection

Object detection is a field that has been studied extensively [52,53,54,55]. The implementation as performed in [2,4] will be used for the location recognition system. This operation is based off of a calculated threshold using a series of images without the novel object present. The median of the distances of each patch from the feature vector it is closest to is found. The standard deviation for each patch from the median of the set of medians is then found. The threshold T finally comes from adding the median of the standard deviations to the median of the medians. This is shown in (5.6).

$$T = \text{median}\left(d_{\text{median}_1}, \cdots, d_{\text{median}_N}\right) + \text{median}\left(\text{std}_1, \cdots, \text{std}_N\right) \tag{5.6}$$

Now that the threshold has been calculated the robot is driven through the same environment, but with a novel object present. An image of percept distances is formed and segmented using the threshold. In order to determine that the object is not noise, a binary image is created. The bottom half of the image is eroded twice by an 8-connected structure element. Finally the largest group of connected patches remaining is selected as a potential novel object. As the robot gets closer to the object, the size of the connected group should continue to grow. If the number of patches exceeds 100, they are stored and added to the training database. This process is shown in Figure 5-2.

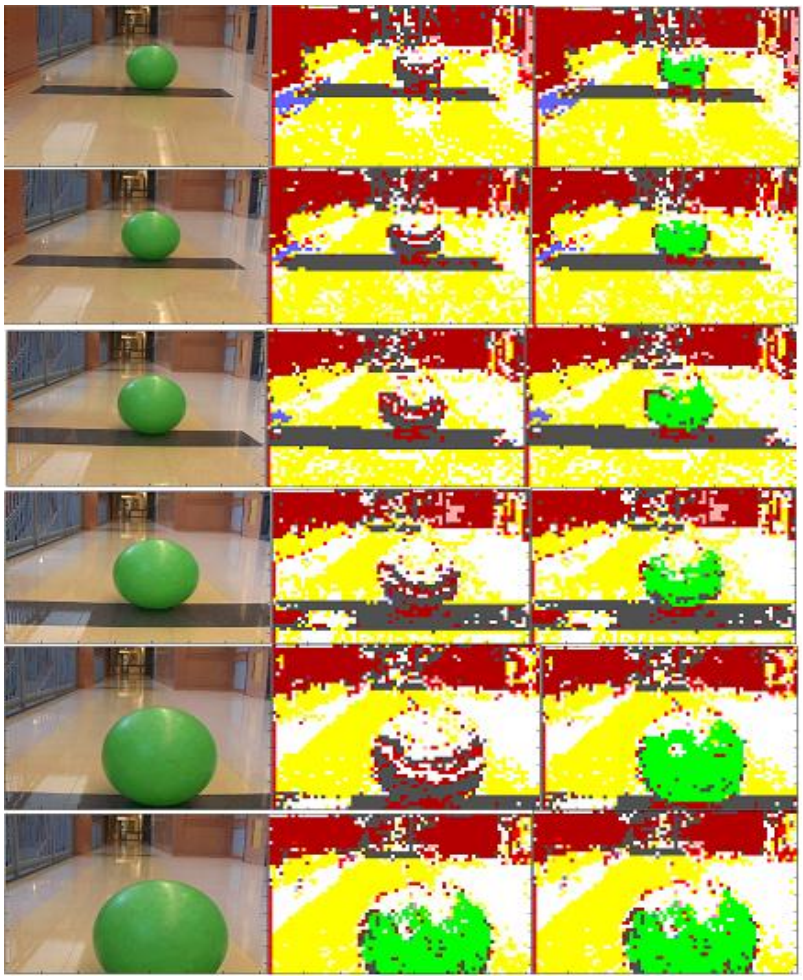


Figure 5-2. (left) the original images, (middle) processed images, (right) processed images after learning. [2]

As shown this method worked well in discovering a single novel object in the environment. Unfortunately it was limited to a single object. In [4], this scheme was expanded to find multiple novel objects at the same time. This was done by using a size constraint on the patches that exceeded the threshold instead of finding the largest group of patches that exceeded the threshold as in [2]. In [4] the requirement for a novel object was that there were seven connected patches that all exceeded the set threshold. This allowed for multiple novel objects to be found simultaneously.

### *E.* Novel Area Detection

Novel area detection, for our purposes, is defined as the recognition of areas not previously trained on. This is similar to novel object detection only this is the study of recognizing entire areas as new. SLAM is capable of recognizing novel areas, but in a philosophically different way. SLAM is able to detect that it does not recognize the geometric shape of an area it is in, but as far as it is concerned the map that it is using is one large region and it will simply add another piece to that map. This work aims to recognize each region as a perceptually distinct region. This aspect of location recognition has not garnered much attention largely because systems performing this type of location recognition typically provide the regions already segmented and then devise a means of telling them apart [21,26]. In this work the boundaries of the regions were found by the system. Therefore the system must also be capable of defining new boundaries for novel regions. There are three incidences of novel areas that this work focused on; correcting previously defined areas, detecting novel areas where the percepts are largely known, and detected completely novel areas where nothing is recognized.

Whenever a system makes decisions it is important to have error correcting methods. In this case, it is the recognition that an area defined in the initial training may be better off as two separate areas. This incidence occurred in [59] and it is the aim of this work to deal with this type of error.

The second type of novel area detection is recognizing that an area is novel even when the percepts are recognized. This can often occur inside a building. The floor initially used in [59] was the third floor of Featheringill Hall at Vanderbilt University. The second floor appears to be largely the same.

The third type of novel area detection is recognizing an area that is completely unknown. This involves recognizing that the percepts are entirely unknown and therefore the region is completely unknown.

## Additions to the Vision System

### *A.* Reflection Detection

Based on observations of the percepts in the image sequence as the robot moves forward, it appears as though there are three distinct behaviors of reflections:

1. Reflections caused by lights directly overhead
    a. Tend to be long static reflections
    b. Look like a distinct objects in the environment
2. Moderately distant objects reflecting in the floor
    a. Moves within the image as the robot moves forward
    b. Looks like actual object
3. Reflections from a distant light source
    a. Small reflections

b. Don’t move much within the image as the robot moves forward

Each type of reflection can be seen in Figure 5-3. The Type 1 reflection shown is due to

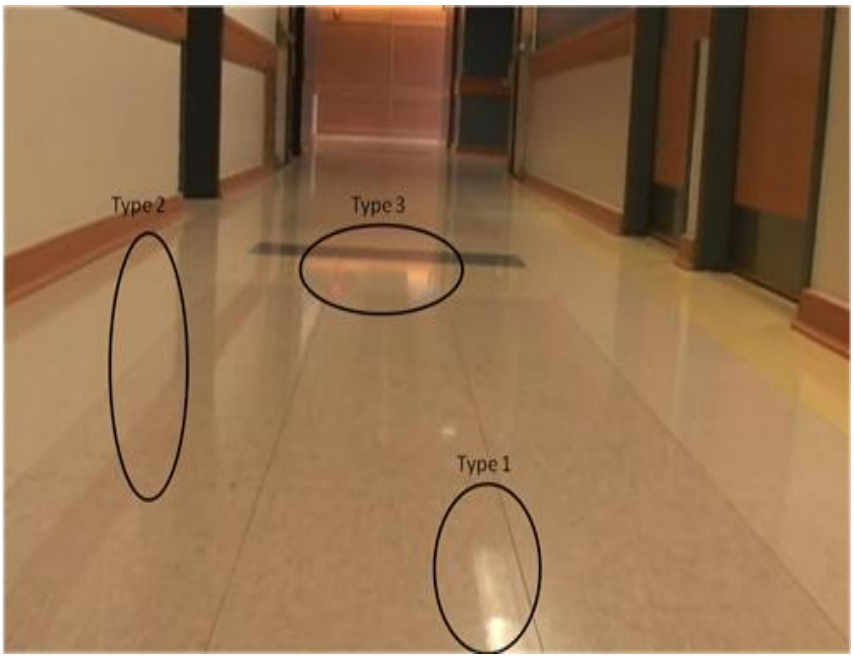



Figure 5-3. Demonstrates each type of reflection. Type 1 is a result of overhead fluorescent lighting. Type 2 results from the wall reflecting off the tiles. Type 3 results from distant light sources

the fluorescent light source directly above the floor. This type of reflection is very consistent and appears to behave as any other percept in the image. Therefore it will be considered an acceptable object. The second type of reflection is labeled as Type 2 and comes from the actual percepts reflecting off the tiles. This type of reflection closely resembles the actual percept that it reflects and is often segmented as that percept. This type of reflection can be detected due to its behavior, because it does not move as an actual percept would, but that will be left as future work. The final type of reflection is labeled Type 3. This reflection comes from a light source a long distance from the robot that generates a reflection due to its angle of incidence off the floor. These reflections are the type of reflection that will be detected. The method used to detect these reflections

will be based on the observed behavior of this type of reflection. As the robot moves in a straight line down the hallway, this type of reflection will appear to remain static in the processed image while all the actual objects appear to get closer to the robot. Therefore, a reflection can be detected by tracking the segmented percepts and observing that a percept is not moving while still persisting over an extended period of time.

The obvious argument against this method would be that the percept being tracked could just as easily be an object moving at the same speed as the robot. While that argument is true, as described in [59], the purpose of this system is to be a developmental robot. Therefore, it is assumed that the robot is in its initial stages of learning its surroundings. This means that the system is still operating under the assumption that everything present is still a part of the static environment. Dealing with more complex dynamic environments has been left as future work.

Detecting this type of reflection is important for the robot to begin to understand that what appears to be seen is not always accurate. As a robotic system it has no concept of object permanence, therefore by recognizing that it can never actually reach these visually present percepts, the association can begin to be made.

The hallway labeled 2 in Figure 5-1 will be the example used to discuss this implementation. The video of the hallway was taken in the reverse direction of the training arrow. This was done because there is a window at the end of the hallway providing a very clear reflection on the floor. An example of an image from hallway 2 and its segmentations are shown in Figure 5-4 (a,b). It is the goal of this part of the work to have the system determine that the windows reflection is in fact a reflection and

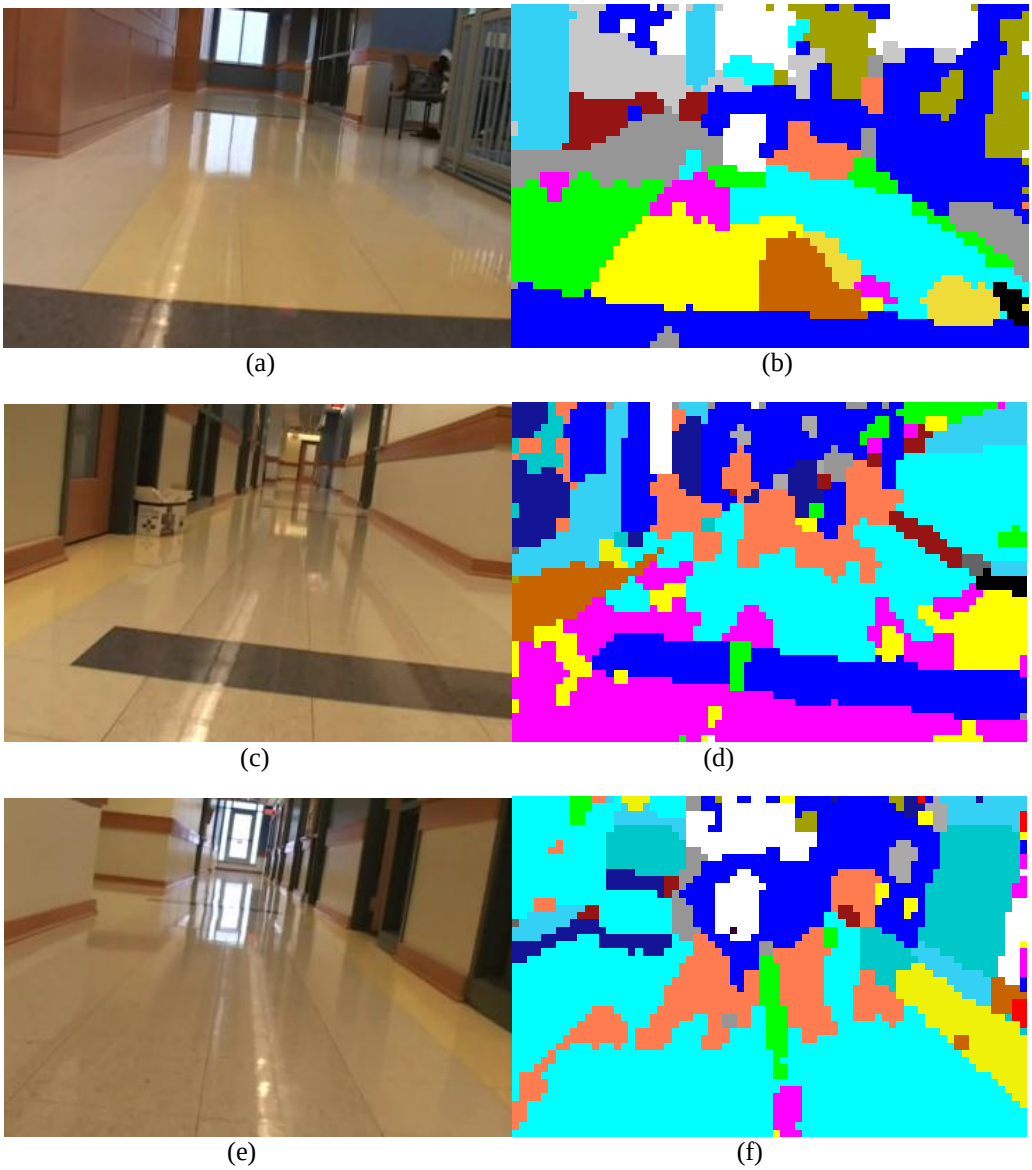


Figure 5-4. (a)Hallway 2 in Featheringill Hall (b) (a) segmented (c) Hallway 5 going from left to right (d) (c) segmented (e) Hallway 5 going from right to left (f) (e) segmented.

not the percept itself.

After the video was taken, the images were extracted at 20 images/sec. This was roughly equivalent to 20 images/ft. The next step in this process is to extract the percept blobs and track them. As stated, only the reflections coming from a distant light source are being sought. Therefore, only the region of the image in which this type of reflection is likely to be found in will be searched. This means that of the 47x71 pixels that make

up the segmented image the top 15 rows, left 20 columns, and right 20 columns will not be looked at in this process.

The method of extracting the most relevant percept blobs was first done in [3]. This method begins with creating an object image for each percept label. The object image is a binary image with a 1 at each pixel determined to be that object and a 0 every where else. In order to reduce noise in the process an averaging filter is applied. This is done as follows:

1. An *n x n* filter size is chosen by the user. The general rule is to use a size equivalent to the patch size or smaller for the application.
2. In order for a group of pixels to be kept as a relevant blob, that group must fill a percentage (P1) of the filter.
3. If the condition from 2 is met, the largest value (M) from the filtered object image is found and all values in the filtered object image that are greater than a percentage (P2) of M are changed into a value of 1 while the others are stored as 0 creating a filtered binary object image.
4. This new image is grouped using a connected component labeling algorithm. Statistics about each group are stored into the object descriptor class (i.e., width, height, and (x,y) centroid of each group)

For this process the averaging filter is set at 15 x 15.

Once the percept blobs have been extracted their centroids are tracked from one image to the next. In this case only the centroids that do not move within the image are of interest. The most a percept blob's centroid can move for it to be considered a reflection is 10 pixels up/down and 10 pixels left/right relative to the centroid's initial

position. The up/down measure is based on the motion of the robot as any actual percept was experimentally tracked to move far more. The left/right motion largely indicates that the robot is not driving perfectly straight, but it also measures variances in the shape of the segmented percept blobs. After tracking each percept if any of them persists long enough without movement it is then considered to be a reflection. The reason that object permanence is important here is because of the length of multiple percepts. For example, consider the yellow floor tiles. The only time they are broken up is when the black stripe is present, so the minimum distance of changes in the tracked environment sets the threshold for the duration that the percept blob must remain. In this case the distance from one set of tiles to another is ~20 feet or 400 images. So if a percept's centroid remains steady and persists for longer than 400 images continuously then that percept is considered a reflection. The results going down multiple hallways are shown in Table 5-1. Refer to Figure 5-8 for the corresponding hallway.

Table 5-1: Reflections detected in hallways

| Hallway | # of Images | # Clusters Tracked | # of Reflections Detected | # of Reflections Detected That Were False |
|---|---|---|---|---|
| 2 | 1041 | 21 | 2 | 0 |
| 5 (lt to rt) | 1561 | 18 | 1 | 0 |
| 5 (rt to lt) | 1801 | 24 | 3 | 2 |

This method applied to the hallway used for Figure 5-4 (a) resulted in two objects being tracked. The first is the white blob in the middle representing the reflection of the window in the floor, and the second percept blob is the orange blob next to the white blob. The second percept blob represents the reflection of the wall that is next to the

window, therefore this is an accurate reflection detection. The details of these reflections and the reflections detected in the other hallways are shown in Table 5-2.

Table 5-2: Details of the reflections detected

| Percept Color Tracked | Hallway | # of Cons. Images Observed In | Total Centroid Motion (in pixels) | Up/Down Pixel Motion | Left/Right Pixel Motion |
|---|---|---|---|---|---|
| White | 2 | 483 | 9 | 5 dn | 4 lt |
| Orange | 2 | 627 | 8 | 0 | 8 rt |
| Orange | 5(lt to rt) | 1232 | 18 | 8 dn | 10 rt |
| Green | 5(rt to lt) | 702 | 3 | 3 dn | 0 |
| Purple | 5(rt to lt) | 559 | 1 | 1 dn | 0 |
| White | 5(rt to lt) | 401 | 2 | 1 dn | 1 rt |

The total centroid motion tracked in Table 5-2 is based on the number of pixels the centroid of the tracked percept blob moves from its initial position. Therefore if the centroid of a percept moves down 5 pixels and left 4 pixels relative to its initial centroid pixel location, the percepts total centroid motion will be 9 pixels. As stated if either the up/down motion or the left/right motion exceeds 10 pixels in any single direction then percept blob will not be considered a reflection. Table 5-2 shows that all six of the percept blobs presented meet the criteria to be considered reflections. Figure 5-4 (c-f) shows images taken from hallway 5 in both directions as well as a segmentation that contains the percept blobs tracked as reflections in that hallway.

The results show for hallway 2 that both reflections detected persisted for some time. The only reason the number of images is not longer is because the video ended.

Also the results show that the percept blobs did not move in the manner expected of true percepts based on the motion of the robot. In the case of the white percept blob the centroid of the blob moved a total of 5 pixels down and 4 pixels to the left while the orange percept blob moved a total of 8 pixels to the right. Over the course of that many images any natural stationary object should have moved more.

While the robot was going from left to right in hallway 5 only one reflection percept was detected. That percept is the orange blob in the middle of the hallway shown in Figure 5-4 (d). This percept represents the reflection created from the light at the end of the hallway. The relatively large amount of motion recorded for this percept blob was largely due to the changing shape of the percept blob. The shape was not consistent because this reflective percept blobs is made up of reflections from multiple objects (e.g. door, wall, light source) and based on the changing light can be classified as the actual objects.

With the robot going from right to left in hallway 5, three reflective percepts were detected. The first two are the green and purple segmented percepts in Figure 5-4 (f). The reason these percepts were detected is because they are a reflection from the light directly above the floor. So it is accurate to call this a reflection however this is not the type of reflection this system is aiming to find. Therefore this is considered an error. The white percept blob representing the window however is correctly labeled as a reflection.

*B.* Percept Classification

Based on the segmentations shown in Figure 5-4 (b,d,f) the need for further classification of the percepts is obvious. Some of the objects from the original image can be seen in the segmented images, but the actual objects are not what are segmented.

Therefore if there is a method of combining the known percepts to represent the objects that exist in the original image, that would be ideal. It is understood that different initial classification methods could be used to improve the segmentations, but no matter what method is used, in an area with as many reflections as are present here, errors will occur and the need to deal with them is present. Therefore instead of going back to simply reclassify the data the decision was made instead to improve the original classification. The first step in the process is to develop a means of determining whether a percept is in fact a percept or an aberration of light. This will be dealt with by tracking the motion of the percepts.

To start with, because the segmentations at the top of the image are largely unreliable due to their distance from the robot, the top 20 rows will not be tracked. Next, a similar tracking method used in the reflection detection will be used here. The method for extracting the percepts blobs will remain the same except because the goal is to track all the blobs that appear, the averaging filter will be 5x5. Also, the (x,y) pixel locations defining each group will be stored. This is needed because previously the assumption was made that the only reflections to be tracked were light sources from a distance. So it could be inferred that the reflection percept blob would persist and not be separated or undergo significant changes as the robot moved forward. Unfortunately, this is not the case while trying to track objects that have to deal with changes in lighting conditions. Therefore, a method for detecting when a single percept blob separates and a method for determining when separate percept blobs merge was also implemented and required knowledge of all the pixels in a group to do so.

Once the statistics of all the percepts blobs have been calculated, tracking the centroids as done before is the next step. Because of the rate of motion of the robot, a single centroid should not move greater than three pixels in any direction. This is the basis for tracking each percept blob from one image to the next. In order to track if a percept blob had separated into multiple percept blobs with the same label, the pixel locations of the percept blob from the prior image are compared to the pixel locations of the current percept blob. If the percept blob from the prior image shares 20% of the same pixel locations and that prior percept blob had already been matched to a different current image percept blob then a split has occurred. The same process is used to detect if two percept blobs merge, only instead of checking if two current percept blobs match to a single prior percept blob the check is for one current percept blob that matches to two separate prior percept blobs. Again this check requires the current percept blob to match the location of the prior percept blob's centroid or 20% of each of the prior percept blobs pixel location's must match to pixel locations of the current percept blob.

Now with the ability to accurately track the blobs as the robot moves, four different classifications for the percepts blobs have been created. The 4 classifications then break down into 10 overall classifications. The break down is as follows:

1. Percept
    a. Normal percept

        Percept blob starts above the 28$^{th}$ row and moves through the image as expected.

    b. Long normal percept

Percept blob starts and moves as expected, but lasts longer then a normal percept would

2. Probably a percept
    a. Late starting percept

        Percept starts lower in the image than expected (greater then row 28 and less then or equal to row 33) and moves through the image as expected.

    b. Long late starting percept

        Percept starts lower in the image than expected (greater then row 28 and less then or equal to row 33), but lasts longer then a normal percept would.

3. Probably an aberration of light
    a. Very late starting percept, but then moves as a percept should

        Percept starts greater then row 33 but moves as a percept should

    b. Very late starting and long lasting percept, but moves as a percept should

        Percept starts greater then row 33 and lasts for a long time but moves as a percept should

    c. Far away starting and ending percept

        Percept begins less then row 28 and ends less then row 28

    d. Starts as a normal percept but disappears suddenly

    e. Starts late and moves as a percept would but suddenly disappears

4. Aberration of light

Percept starts greater then row 28 and moves in ways that static percepts cannot move (e.g., up in the image)

These classifications covered all the possible outcomes of tracking the percepts. This tracking method was tested using the same images from the hallways as the reflection detection and the results are shown in Table 5-3.

Table 5-3: Percept Classifications

| Hallway | Percepts tracked | # Percepts | # Probably Percepts | # Probably Light | # Aberration of light |
|---|---|---|---|---|---|
| 2 | 34 | 6 | 6 | 20 | 2 |
| 5(lt to rt) | 42 | 11 | 6 | 22 | 3 |
| 5(rt to lt) | 42 | 8 | 6 | 26 | 2 |

The results in Table 5-3 show that the majority of percepts detected behave in an odd manor. Based on the amount of reflectivity observed in the images in Figure 5-4, these results are not surprising.

*C.* Interpretation of the Results

At this point the system is able to identify when percepts are not behaving in the manner expected. It can identify reflections caused by distant light sources, percepts behaving as expected, percepts that are behaving almost as expected, and percepts not behaving at all as expected. With this foundation the next step is to develop a means of using this information.

The first use could be taking the obvious percepts and using them to improve the segmentations. An example would be of the black stripe in Figure 5-4(a) being segmented blue. The same label is then used to segment out multiple other locations in

the same image. By recognizing that the black stripe segmentation behaves as a percept should, the system could possibly use all the other feature vectors that are classified by the same blue label and create a new centroid mean vector from them. Another use of this information could come from tracking the 10 subclasses of the percept blob. An example here would be if a blob were classified as 3.d. and then a new blob started where it disappeared and was tracked as either 2.a. or 3.a. then this could indicate that a single percept was dealing with refractions of light and based on further investigation by the system either the percept labels can be combined as a single percept or be marked as indicating aberrations of light or actual percepts.

## Novelty Detection

### *A.* Novel Object Detection

Novel object detection using this vision system has already been implemented in [2,4]. Therefore, it has only been modified for more practical use in this implementation. Figure 5-5 demonstrates the

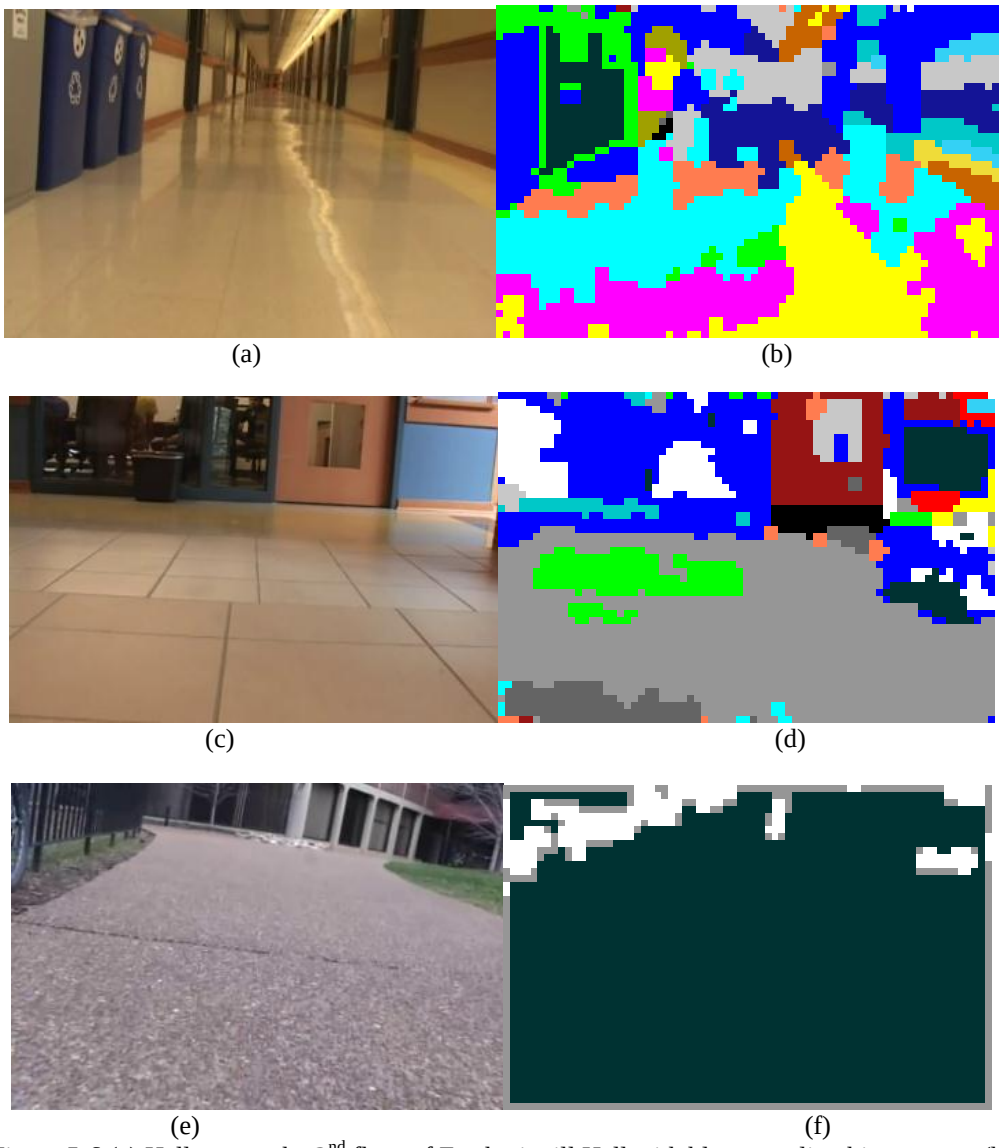
Figure 5-5 (a) Mobile robotics lab at Vanderbilt University (b) Segmented image of (a)

general effectiveness of this method in detecting multiple untrained objects in a single image. This image was taken from room 304 represented in Figure 5-1. The room has

not been trained on, but the floor which has been globally learned from the training for the hallway is largely recognized. As demonstrated, after the erosion of the segmentation, all three cones, the ball, and the poster are recognized as novel objects.

As stated, the goal of this system is to be a developmental location recognition system. This means that only large percepts within the environment are sought after. With that in mind it was empirically found that the desired distinct percepts from the environment were made up of at least 50 connected pixels. Therefore, only novel object percept blobs that exceed 50 connected pixels in a single image are kept for further tracking. Also the threshold as described in (5.6) was increased to be sure that only objects that appeared significantly different were found.

As of writing this paper, the tracking of the novel objects from one image to the next has not been implemented. That aspect of this work will be included in future works. At this point this system has demonstrated the ability to accurately detect novel percepts in both regions that have been trained on and regions that it has not been trained on. Examples of this are shown in Figure 5-6.



(a) (b)

(c) (d)

(e) (f)

Figure 5-6 (a) Hallway on the 2nd floor of Featheringill Hall with blue recycling bins present (b) segmentation of (a) with the recycling bins represented by the color indicating novel object. (c) Region 6 of the 3rd floor of Featheringill Hall. (d) Segmentation indicating the blue wall in the top right corner in a novel object. (e) Outside of Featheringill Hall. (f) segmentation of (e) indicating the image is largely composed of novel objects.

Image (a) of Figure 5-6 shows hallway 9 from Figure 5-10. By comparison to Figure 5-4 (c) and (e) this hallway looks largely similar to the 3rd floor of Featheringill Hall. The blue recycling bins in the top left corner however are novel objects. In Figure 5-6 (b) they are segmented with the dark green color randomly selected to visually indicate that a novel object is present. In Figure 5-6 (c) the image is taken from the

hallway labeled 6 in Figure 5-1. According to the segmentation in Figure 5-6 (d) the blue wall in the top right corner is a novel object. Although the blue wall was learned previously, because of the direct sunlight from the sunroof on the wall it is currently considered a novel object. Finally an outdoor image was used in Figure 5-6 (e). Figure 5-6(f) shows virtually the entire image as a novel object.

These are just three examples of thousands of processed images. They do however cover the three situations most commonly encountered. Figure 5-6 (a,b) demonstrate an actual novel object present and accurately found in a region, Figure 5-6 (c,d) represent a false novel object recognized due to a change of lighting, and Figure 5-6 (e,f) represent an entirely novel area being observed.

It is because of these three situations that it is not possible to simply find the mean feature vector of the novel object blob and add it to the known percepts. Therefore as mentioned, the novel objects will be tracked over a series of images. If they persist then the mean vector of all the feature vectors gathered will be found and used to represent the new object. When the majority of the image is considered a novel object the system will have two cases to deal with. The first is the possibility of multiple novel objects near each other in a known region, and the second is that the robot is in an unknown region. In the former case a classification technique will need to be used to segment the multiple objects and add their mean feature vectors to the database. In the latter event, the system should process all of the new information to model the new regions as performed in [59].

*B.* Novel Region Detection

Systems such as SLAM are capable of mapping novel areas and localizing a system within that area. The type of localization used is to pinpoint the robot's exact

location on a generated map. The difference between that type of system and the goal of this work is the ability to visually recognize the region in a perceptual sense. Therefore, the type of novel region detection used for SLAM does not apply here. As far as SLAM is concerned the entire map is one large region whereas in this system the area has been segmented as shown in [59]. The goal now is to have the system detect when it is not present in one of the known regions. The experiments will test the system in three ways.

The first problem is error checking the original classification of areas. As explained in [59], the models representing the areas labeled 5 and 6 in Figure 5-1 performed poorly when being used to classify new images from those regions. It was found the in the initial training both regions should have been split into two more regions. Therefore the ability to detect previous mistakes will be looked at.

The second problem will be the ability to detect when the robot is in a novel location that appears very similar to the trained area. An example of this type of similarity can be seen in Figure 5-7. The image

(a) (b)

(c) (d)

Figure 5-7 (a) Hallway 1 on the 3rd floor of Featheringill Hall. (b) Hallway 1 on the 2nd floor of Featheringill Hall. (c) Image from hallway 6 (d) Image from new area found in hallway 6.

in Figure 5-7(a) came from the 3rd floor of Featheringill Hall while the image in Figure 5-7 (b) came from the 2nd floor.

The final type of novel location recognition will be the ability of the system to determine when it is in completely novel regions.

A novel region will be declared if three criteria are met. The first is that a pre-set distance threshold set for the value found using (5.5) has been crossed. This threshold was first set using equation (5.6), but was modified by the user. The second is that a novel area must be detected for at least 25 images. Based on the frame rate extraction speed (~1 frame/sec) and speed of the robot (~1ft/sec) this equated to roughly 25 ft. This means that if an area is not at least 25 feet long it will not be considered a new area. The final criteria necessary to declare a novel area is that the dominant label classifying that

area must represent it in less than 70% of the images exceeding the threshold. The 70% threshold was empirically found to work well as trained areas have ~90% accuracy while untrained areas perform with significantly less accuracy.

The first experiment was performed on the 3rd floor of Featheringill Hall. Figure 5-1 shows the areas of the 3rd floor that were trained on. Figure 5-8 shows a map of the entire floor and the resulting

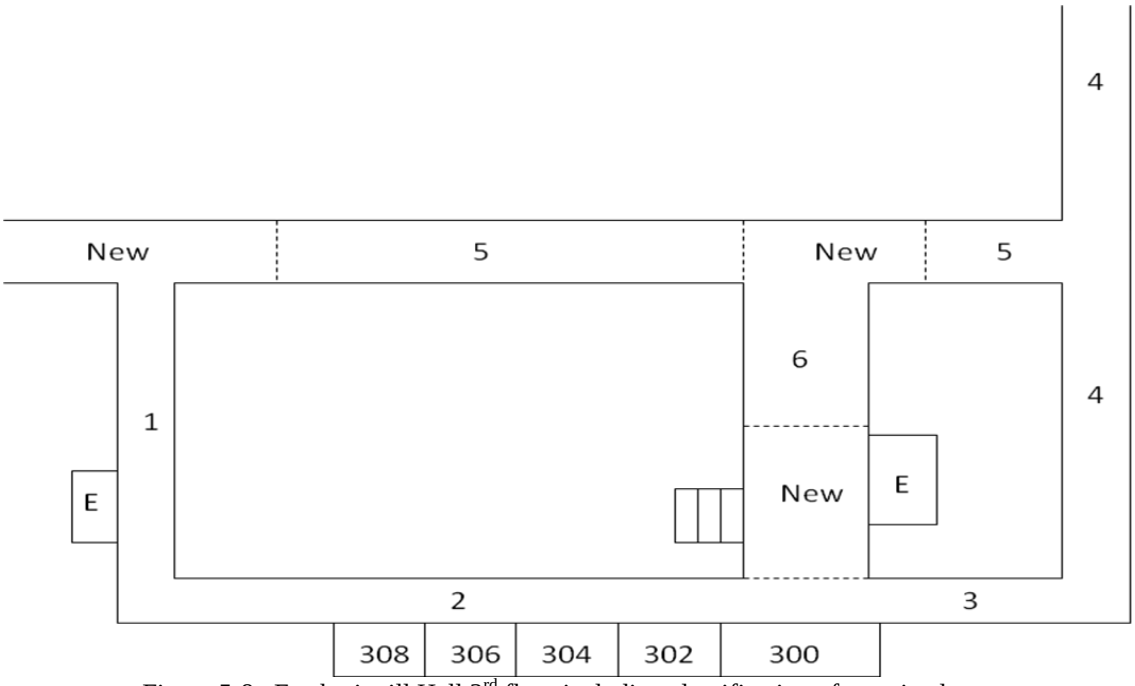


Figure 5-8. Featheringill Hall 3rd floor including classification of untrained areas.

segmentation of the floor. This shows that the system correctly identified the extended untrained area of hallways 4 and 5. This also shows that three novel areas were detected. Although we would consider hallway 5 to be one continuous hallway, based on the images the system is correct to identify them as new regions. Firstly the new region to the left of hallway 5 has a large window providing a very large reflection as seen in Figure 5-4(e). As the robot approaches that change, the hallway appears to be very

different. It is also windows in the middle of hallway 5 that create the novel area. The split in hallway 6 comes from a genuine change in appearance. This can be seen in Figure 5-7 (c) and (d) where Figure 5-7 (c) is what the top of hallway 6 appears as and Figure 5-7 (d) is what it looks like in the newly declared region. Table 5-4 provides the results of processing the images while only including the floors that had at least 25 novel area images found.

Table 5-4: Results from processing the 3rd floor of Featheringill Hall with novel region detection

| Hallway | # Images | % Labeled Correct | # Novel Area Images | % of Novel Area Images Labeled 1 | % of Novel Area Images Labeled 2 | % of Novel Area Images Labeled 3 | % of Novel Area Images Labeled 4 | % of Novel Area Images Labeled 5 | % of Novel Area Images Labeled 6 | New Region Created |
|---|---|---|---|---|---|---|---|---|---|---|
| **2** | 150 | 86% | 119 | 3% | 87% | 8% | 0 | 2% | 0 | No |
| **5 (trained region)** | 67 | 72% | 26 | 7% | 43% | 0 | 0 | 50% | 0 | Yes |
| **5 (untrained region)** | 180 | 66% | 43 | 60% | 0 | 6% | 0 | 12% | 21% | Yes |
| **6** | 92 | 66% | 25 | 36% | 0 | 0 | 0 | 48% | 16% | Yes |

Hallway 2 was the first hallway to have processed images exceed the set threshold. Of the 150 images taken from hallway 2 119 of them crossed the threshold. The reason for such a high number is because there is a skylight above this hallway providing a great deal of natural light and although it is still able to recognize the hallway, the light skews the distance calculations. This hallway is a good example of the importance of keeping track of the label changes as the hallway is processed. As seen in Table 5-4, although the images exceed the threshold, of the 119 that do, 87% are still classified as hallway 2. Because of this a new region is not declared.

The next hallway to observe a potential new area is hallway 5 after making a left turn from hallway 4. This new region is a result of an error in the original training that

was noted in [59]. The region labeled 5 in Figure 5-1 should have been split from the start. In this case 26 images are declared to come from a novel region with 50% of them being labeled as hallway 5. Therefore a new area has been declared.

The second novel area created in hallway 5 is due to the window at the end of the hallway as seen in Figure 5-4 (e). The light coming through the window changes the visual properties of the hallway to such an extent that this is considered an appropriate novel region.

The last new area generated on this floor is in hallway 6. This split is again due to the initial training. As shown in Figure 5-7 (c) and (d) the areas are visually distinct regions, and based on the criteria a new region has been created.

The second test of the novel region detection was performed on the 2nd floor of Featheringill Hall. The layout of this floor is different, but all of the color schemes are exactly the same as seen in Figure 5-7 (a) (b). Figure 5-9 shows the layout of this floor.

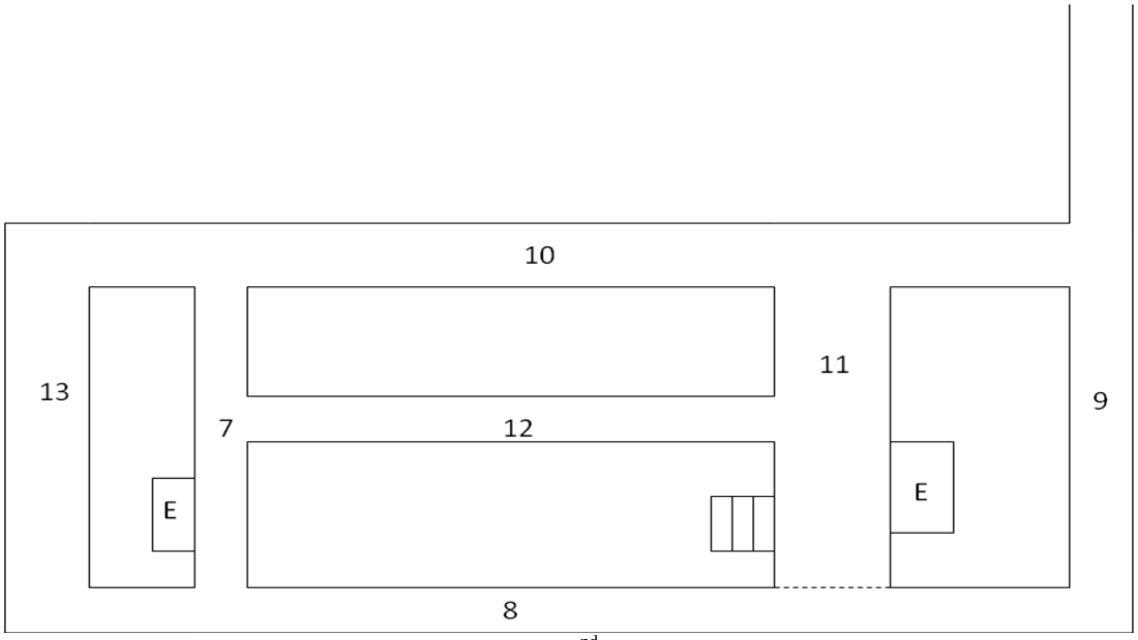


Figure 5-9. Floor layout of 2nd floor of Featheringill Hall.

Because the floors appear so similar only hallway 12 was considered a novel area. This is because it is the only hallway without an exact counterpart on the 3rd floor. An image of this hallway is shown in Figure 5-10. As a note, even though hallway 13 does not

(a) (b)

Figure 5-10 (a) Featheringill Hall 2nd floor, hallway 12 (b) Segmentation of (a)

exist on the 3rd floor, its entire make up matches hallway 4 on the 3rd floor and it is classified as such.

With such a high level of similarity it unreasonable to expect the system to recognize that it does not know these areas. By comparing Figure 5-7 (a) to Figure 5-7 (b), the only discernable difference is a single panel of wall that is blue on one floor and white on the other. Therefore the results are not surprising. What does indicate some measure of success with this technique is that hallway 12 was identified as a novel region. Of the 159 images comprising hallway 12, 131 of them exceeded the novel area threshold. Of those 131 images the known region that they were segmented most as was hallway 3 with 37% of the images. Also as shown in Figure 5-10(b), all the percepts in this image are known and the hallway is still identified as a novel region. Therefore the system is capable of detecting novel regions comprised of known percepts with the caveat that obviously there must be some discernable difference.

The final test of the novel region detection is testing the system on completely novel regions. This was done by driving the system around what the author classified as 11 novel regions starting on the 1st floor of Featheringill Hall and comprised of driving the system outdoors as well as through multiple hallways of another building. In total 2005 images were gathered and of them 1786 images were considered novel images with a high degree of variance over the regions that the system believed itself to be in. This means that the system successfully identified that it was is areas that it did not know.

*B.* Interpretation of the Results

The overall goal of this system is to develop a truly developmental vision system that is capable of learning its environment without any help or limited help from a human user. To that end the detection of novel objects and novel regions is a very important step. This system has demonstrated the ability to identify both novel objects and novel regions based on the methods presented. In the case of novel object detection, a threshold is set and if 50 or more connected pixels exceed that threshold then the novel object is counted. For novel region detection a similar threshold is used based on the distance the overall image is from the known region model that it is closest too. If the threshold is exceeded for more then 25 images and a single known region does not represent at least 70% of the images considered to be in a processed region, then that area is considered novel.

The results show that both systems work well at this stage. As shown in Figure 5-6 numerous novel objects were accurately identified. Although it is possible for light changes to cause false novel objects to be found, by adding a tracking system and making sure that the novel object persists over a series of images, this should be limited.

The results of the novel region detection presented show that the three criteria used will work. It was able to detect errors in previously trained data, a new hallway in a region with known percepts, and detect completely novel regions. Therefore the next step of this vision system will be to start incorporating the information gathered.

## Conclusions and Future Work

The goal of this work is to create a system that is capable of visually learning the environment around it in a manner that will allow it to visually recognize places at a later time. The system, as presented in [59], starts by first classifying the percepts and regions from a training region. It then develops a model of each region using the percepts present in that region. Therefore the next step was to improve the system, giving it the ability to start to interpret what it is segmenting as well as the ability to learn new objects and regions.

The first improvement to the vision system was the ability to detect reflections caused by distant light sources. This was done based on the behavior of the reflections created. It was observed that as the robot moves through the environment, these percepts tended to stay in the same pixel locations in the image. Therefore by tracking the centroids of the percept blobs and noticing when there was no movement relative to the robots movement, a reflection was detected. This was run using three hallways and in all three cases reflections were accurately identified.

The second addition to the vision system was a classification system for all the percept blobs as the system moved. This method classified the blobs into four main categories with 10 subcategories present. Again this classification was based on the

movement of the percepts as the robot moved. If the percepts moved as expected they were classified objects, and if they did not they were classified as probably a percept, probably an aberration of light, or an aberration of light.

The third addition to the system was the inclusion of a previously implemented novel object detection method. As shown this method was able to detect the presence of multiple novel objects while being tested over a large series of images. The next step in this process will be tracking the objects to reduce false identification and then incorporating the novel objects into the database.

The final addition to the system was the addition of novel region detection. The three criteria that a series of images must meet are first that a pre-set distance threshold set for the distance between the processed image and the model of the region that image has been classified as has been crossed. The second is that a novel area must be detected for at least 25 images. The final criteria necessary to declare a novel area is that the dominant label classifying that area must represent it at in at least 70% of the images exceeding the threshold. Based on these criteria the system was able to detect errors in the previously created models, find a new hallway in a region composed entirely of known percepts, and detect multiple novel regions where the majority of percepts were unknown.

As the focus of this work was to detect events and provide information about the environment, the next step will be to use this information in meaningful ways. The first step will be to track the novel objects in order to limit false detections. Once that has been completed the novel objects will need to be learned and added to the database. This will involve implementing a classification technique to determine if one or more novel

percepts were detected, and then adding the mean vector of the percepts found to the database. Because the number of percept vectors should be relatively low a technique such as a minimum spanning tree will probably work best in this scenario. Simply using K-means again will result in over-segmentation and further need for processing information.

The next step to take with the novel region detection will be to add the newly discovered regions to the database as well. This will be a little bit more involved as there are numerous cases to consider. First the series of novel images will need to be passed through the relative perceptual difference measures shown in (3) to determine if more than one novel region has been found. Then if novel objects are present they will need to be added to the database. Finally with all the objects and regions segmented, the models of the newly found regions can be generated.

With the addition of the novel regions and object detection implemented the information from the reflections and percept classifications can become useful. Making use of distant light sources can be used to aid the novel region detection. For example hallway 5 on the 3$^{rd}$ floor, as shown this new region at the end of the hallway was generated because of the reflection modifying the view of the hallway. By recognizing that a reflection was present a method of removing it could be developed to allow the system to recognize that as a continuation of the same hallway. It is recognized that numerous methods for this exist (e.g., SLAM), but it is one of the goals of this work to get as far as possible just using vision.

The final addition will be incorporating the classifications of the percept blobs. An example was given in section 3-C of how the combination of percept classifications could lead to combinations of percepts thus cleaning up the segmentations.

# CHAPTER VI

## Manuscript III: A Review for Efficient Use of a Very High Dimensional Feature Space for Percept Classification

Christopher Costello, Mert Tugcu, Xiaochun Wang, Jonathan Hunter, and D. Mitchell Wilkes

Vanderbilt University

Nashville, TN

To be Submitted as a Regular Paper to ISRN Machine Vision

## Abstract

It's no secret that knowing every detail of an environment will provide higher percept segmenting accuracy than a system with minimal information. The problem is that the curse of dimensionality makes it very difficult to operate in very high dimensional spaces. Because of this most systems aim to model features based on very specific data that requires more complex pattern recognition techniques oftentimes reducing robustness of the system. However proper modeling combined with the power of graphical processing units (GPUs) now allows for much higher dimensional data to be used in real time situations. This paper will describe the strengths and weaknesses as well as efficient methods of using a very high dimensional feature space. This work will use the hue, saturation, and value color space domain quantized into a 10,000 dimensional feature space, and explain the benefits and weaknesses of the various classification, and data processing methods used while designing systems ranging from human motion segmentation [3] to location recognition [59].


## Introduction

There are numerous methods for segmenting percepts in images using many different features. Some of those features are color [1,2], texture [1,2], shape [6,15], SIFT [14], and so-on. Often, multiple features are combined in the segmentation process [13,21]. These result in various sizes of dimensionality for processing occasionally resulting in hundreds of dimensions being used. However, the question of what if the feature space was made even larger has not largely been looked at. This has mostly been because of the inability to process such high dimensional data, but with appropriate

modeling and implementation it is possible to expand the dimensionality of the feature space to at least 10,000 features while still operating at real time speeds.

The feature vectors used in the work will represent a pdf of the histogram of the hue, saturation, and value (HSV) color space quantized into a 10,000 dimensional feature space with equal sized bins extracted from a 15x15 patch of pixels. Based on the size of this data set two problems need to be addressed. The first is the "curse of dimensionality". This applies to the difficulty of appropriately processing high dimensional data [56]. This will be addressed both in the modeling of the data and in the classification techniques used, and will be discussed later. The second problem with using this high of a dimensionality is the amount of training data required to properly train a system.

Because it is easier to answer, the problem of gathering enough training data will be addressed here. The answer is that for applications involving large amounts of data, such as video, there is in fact no problem. This is due to the systems use of images which are inherently loaded with information. In fact, most applications aim to reduce the amount of information extracted from an image to specific points so that it is easier to use [21,22,26,33,38]. Therefore extracting at least five times the number of dimensions as suggested in [47] will not be difficult.

Now that the main problems introduced by such a high dimensionality have been mentioned, it is time to mention the benefits. The most obvious benefit is the segmenting power of the feature space. The quantization process includes 100 hue bins, 10 saturation bins, and 10 value bins resulting in 10,000 bins. This puts the emphasis of segmentation

on the color spectrum allowing for very similar color objects to still be segmented. An example of this can be seen in Figure 6-1. The green ball on the right of Figure 6-1(a)

(a) (b)

Figure 6-1.The result of using a very high dimensional space. (a) The system is only trained for green ball located at the right section. (b) The system does not segment the ball on the left at all. [1]

was trained while the ball on the left was not. The segmented image in Figure 6-1(b) shows the correct ball was segmented while the other very similar ball was not.

The other benefit is the ease of segmenting the percepts. Although time is an issue that requires a great deal of consideration, the massive size of the data space means that only very similar percepts will have any overlap at all. This means that most clusters will be orthogonal to each other and will allow for the use of very simple pattern recognition techniques that typically do not perform very well in lower dimensions. For all the works done using this system, a simple Euclidian distance measure between feature vectors is the only calculation performed [1,2,3,4,5,59].

The next section will provide the required background information. Section III will provide a high level description of each of the implementations and a review of their performance. The final section will describe the strengths and weaknesses found in using very high dimensionality for percept segmentation.

## Background

### *A.* Curse of Dimensionality

The curse of dimensionality, introduced in [56], refers to the complications that arise by the exponential increase in volume that comes from increasing the dimensions of the feature space. The term now commonly refers to all the challenges posed when processing high dimensional data [50]. There have been numerous studies on the effects of the use of distance calculations with high dimensional data, more specifically the nearest neighbor (NN).

The work in [51] describes the faults in using NN calculations in high dimensions and how the results can begin to erode in as few as 10-15 dimensions. The underlying reason for this, as shown in [51], comes from the distances from a single point to all data points converging to the same distance. The exception to this problem is if the majority of the data points lie outside of some set threshold distance derived from the min NN distance. If this holds true then there is enough separation to provide meaningful results, and if it does not, then distance calculations on the data set as a whole will not yield useful results. Although the current work did not calculate a threshold distance from which to compare all of the data point distances, the quality of the results imply that they do meet this criterion, and that distance calculations can be used.

The next issue with very high dimensional data is the problem of both outliers and hubs. Outliers are data points that exist on the fringe of the expected cluster region potentially crossing into another cluster's feature space. The primary means of dealing with outliers is to thin the data set before use [50]. Numerous methods of performing this

have been studied [57], and a simple thinning method devised has been implemented [3,59] in the works using the very high dimension vision system.

Hubs are formed by points that appear in more query point k-NN lists than other points [49]. Although they have not been studied extensively they have been noticed in research using high dimensional spaces [60,61], in ref [49] it is suggested that "bad" hubs are caused by two factors. The first is violating the cluster assumption [58]. This roughly states that most pairs of points in a cluster should be in the same class. The second factor is the high intrinsic dimensionality of the feature space. This means that the high dimensionality will exacerbate the effects of violating the cluster assumption more so than in a lower dimensional feature space resulting in incorrect classifications. Again based on the assumption that our model creates enough separation of the recorded data, we have not yet experienced these effects.

The final aspect of the curse of dimensionality has to do with practical application. Although the effects on the calculations are important to consider, so too is the implementation. In this work a 10,000 dimensional feature space is used, and based on the rule of requiring a training database at least five times the dimensionality of data, memory quickly becomes a problem that needs to be addressed. Also, attempting to perform any type of vector calculations on 10,000 length vectors is very impractical. Therefore a sparse vector representation will be used and what it means will be discussed when explaining the vision system in Section III.

### *B.* K-Mean Clustering

The goal of K-means clustering is to partition the data into K clusters [36]. This is done by first selecting K mean vectors, $\mu_k$. This can be done randomly or some other

defined method [1,2]. The next step is to assign all other feature vectors to the $\mu_k$ vector that they are closest too. When all of the feature vectors have been clustered the mean of each cluster is recalculated and this process repeats until the data converges.

K-means is a powerful clustering algorithm commonly used in machine learning applications. Its simplicity makes it very robust and fast to process, especially when random means are used, for many applications. The weakness is that K must be preset. This means that the exact number of classes present must either be known beforehand or a method of determining K must be devised. Another problem with K-means is that it is very reliant upon the cluster assumption mentioned. If the data does not have very much separation then the clusters formed will not be very useful, or will be the equivalent of a "bad" hub. Because of this K-means is not suited for all applications. Given these issues with K-means, it is used here largely because the results of the segmentations were sufficient to provide enough information for further processing, *e.g.*, location recognition [59], and we wanted to compare a simple fast method of processing the data to a slow and precise method. In this case it was a minimum spanning tree classification.

### *C.* Minimum Spanning Tree (MST)

A MST is a graphical analysis of random point sets of data [3]. Each point in the data set is connected to another point in way that only one path can exist to connect any two data points. For the tree to be a minimum spanning tree the connections between each data point are minimized and the distances between all the data points are used to determine the weight or quality of the tree. Once the points are all connected a preset number of cuts between the points are made. This means that if N cuts are set to be made

then the longest N distances between any two points will be cut. The connected points that are left will then be considered a cluster of data points.

The first benefit of this method of classification is that the number of clusters does not need to be known before the process. Another benefit of this method is that it is very accurate. Due to the rigorous processing method the clusters retained are typically very useful for segmenting images.

The drawback to this method of processing data is the length of time it takes to generate a MST. Depending on the size of the database this can take from hours to days. The other problem with this method is that the number of cuts still needs to be predetermined, although this was addressed in [3].

*D.* Nearest Neighbor (NN) Classification

The NN classification method is a well known method that provides very accurate results. This method is considered to provide the best possible results with the drawback of requiring an unrealistic amount of time to process the information. This method calculates the distance between each untrained feature vector $x_i$ to every feature vector $y_j$ in a labeled data set. The label of the $y_j$ that provides the minimum distance D for each $x_i$ is then set as the label of that $x_i$. This is shown in Eq (6.1)

$$D = \min\left(\sqrt{\|x_i\|^2 + \|y_j\|^2 - 2(x_i \cdot y_j)}\right) \tag{6.1}$$

The results of this method are often very accurate for large amounts of training data. The drawback it the length of time it takes to perform this technique for large training data sets. Although it can be quite useful for systems that require great accuracy without time constraints, it is not very useful when time is a primary concern.

*E.* K-Means Search Trees

Anytime a large amount of data is present for a NN classification to be used, the data can be segmented into a search tree to speed the process up. It has been found that a k=3 K-means search tree was the optimal tree for the work. The proof as provided from [5] is as follows:

*C*=cost of calculating one distance between two vectors

*k*=# of distance calculations per node

*N*=total number of training vectors

*L*=# of levels in the tree

TC=total distance calculation cost

The number of levels in the tree, as a function of *k* and *N*

is given by

$$L = \log_k N \tag{6.2}$$

Thus, the total computational cost is

$$\mathrm{TC} = C \cdot k \cdot L = C \cdot k \cdot \log_k N \tag{6.3}$$

Assuming that *C* and *N* are constants and $k \in Z$ (i.e., *k* is an integer), the value of *k* that produces the minimum TC is empirically determined to be approximately 3. If *k* is allowed to be real-valued we obtain

$$\frac{dTC}{dk} = 0 = C \cdot \left(\frac{\ln N}{\ln k} - k \cdot \frac{\ln N}{\ln k^2} \cdot \frac{1}{k}\right) \tag{6.4}$$

which simplifies to

$$\frac{1}{\ln k} = \frac{1}{\ln k^2} \tag{6.5}$$

Thus

$$k = e \approx 2.7183 \tag{6.6}$$

With k set, the tree is easily generated as follows: The first node or root node of the tree was created by randomly selecting three feature vectors from the training database. The rest of the database was then clustered into three child nodes corresponding to whichever centroid they were closest to, based on the Euclidian distance, in the root node. Then for the three new nodes in the second level of the tree, three new centroids were selected in the same way and the data segmented. This process continued until one of three conditions stopped it. The first condition that can stop a node from propagating into child nodes is if all the feature vectors in that node represent the same percept. In this case the node is considered a pure leaf node. The second reason a node will cease to expand is if the number of feature vectors in that node is below a preset threshold. In this case it is considered an impure leaf node. Finally, the last reason the tree will cease to expand is that the preset maximum number of levels has been reached. This too results in impure leaf nodes.

*F.* Nearest Mean (NM) Classification

The NM classification technique is quite often not a very good means of segmenting data. This is because it does not have any information regarding the fringe area of the clusters It only has information on the core. The benefit of this method though is the small amount of data needing to be stored (one feature vector per cluster) and the time required to segment images using this method.

NM is processed the same as NN in (6.1) only $y_j$ represents the mean vector of cluster j. The typically small number of clusters (in our case about 18) explains the substantial reduction in processing time.

## Vision System

### *A.* Overview

The works using this very high dimensional feature space have been demonstrated in [1,2,3,4,5,59]. With the aim of this paper to consolidate all that has been learned about the use of such a high dimensional feature space, the systems it has been used with must first be addressed. Due to the fact that not all of the works provided the final implementation of a method used, only those works directly addressing the training classification and data classification methods will be described. This chapter will start by describing the representation of the 10,000 dimensional feature vectors extracted from each image. This process is the same across all implementations. Second, the original system created in [1] will describe the system as trained under supervised learning and using the a-NN search tree. Then the use of the MST in [3] will be explained. Next the implementation of the MLE search tree will be discussed. Finally the work in [59] will discuss the use of K-means classification and the NM image segmentation approach. The results of each of these systems and their implications on the use of the very high dimensional feature space will then be discussed in the next section.

### *B.* Feature Extraction

The process of extracting the feature vectors begins with the acquisition of an RGB image. The RGB image is than converted into an HSV image in order to represent the image intuitively as the hue, saturation, and value of a color. Once the HSV values are obtained the feature extraction is performed. The image is broken into 15x15 patches that have a 10 pixel hop in both the vertical and horizontal directions. This means that the first patch, starting in the top left corner of the image, will begin at pixel coordinate (0, 0)

and the second patch will begin at pixel coordinates (10, 0). Then when the first row is completed there will be a 10 pixel vertical hop downward. The overlap is used to help blend the boundaries of objects. Then a probability density function (pdf) of the distribution of the HSV colors in a patch is found. The pdf is computed from a histogram of the HSV colors that have been quantized into 10,000 bins. This process is performed by first evenly distributed the hue into 100 bins, ranging from 0 to 1. Then the saturations and values are distributed into 10 bins each, also ranging from 0 to 1. Finally these three values are combined resulting in the 10,000 different possible color features. Because of the 10,000 possible color features, and that, in the worst case scenario, the 15 x 15 patches can only provide 225 different potential color features, a highly sparse representation is used here. Therefore each patch is represented by a feature vector that contains two vectors. The first vector holds the index of each color feature detected, and the second vector holds the value of the color feature. This representation provides numerous benefits.

The first benefit is that there is no computational cost for increasing the dimensionality of the feature vectors [1]. In all works using the very high dimensional feature space, the Euclidean distance measure is used. Therefore given two vectors x and y the equation to find the distance between them can be given as:

$$D = \sqrt{\|x\|^2 + \|y\|^2 - 2(x \cdot y)} \tag{6.7}$$

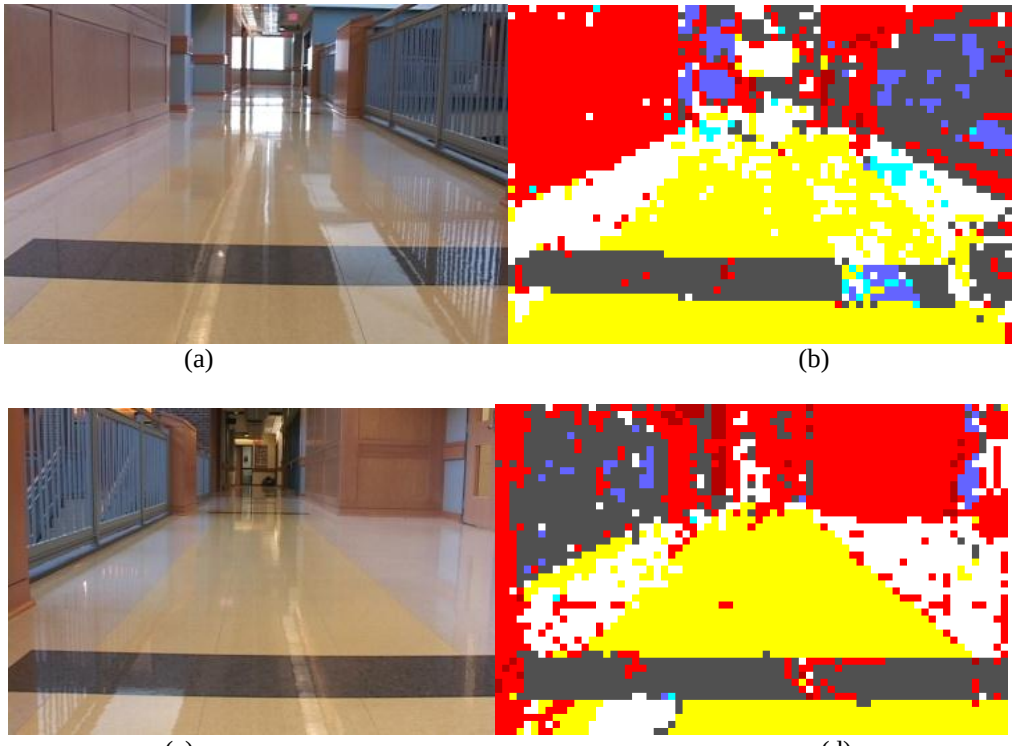
Because the norm of the vector only requires the non-zero elements, and the inner product only requires the non-zero elements that exist in both vectors, this representation is immune to increased computational costs due to increased dimensionality. The only way to increase the computational cost is to change the size of the patches used. So, if an

N x N patch size is used and $N^2$ unique color feature indices are found, then the worst case distance calculation would require $2N^2 + 1$ index fetches [1]. The additional one comes from a Laplacian texture feature added to the feature vector which was used in [1,3].

*B.* Under Supervised Learning and Segmented Using an a-NN Search Tree

The use of such a high dimensional feature space was originally implemented by [1]. The method of training the system was through direct supervised learning. In other words, a user selected regions of a set of images and provided labels for those regions. Because the database should be at least 50,000 feature vectors (according to [47]) the NN classification approach required too much time. So the approximate NN (a-NN) search tree as described was used.

With the data represented in this format when a new feature vector was introduced, it was processed as follows: Once the feature vectors are extracted, the distance from the current feature vector to each of the three centroids in the root node are found. The child node of the centroid that provides the shortest distance to the feature vector will be used next. This will continue until a leaf node is reached. If that leaf node is pure then the label for the percept will become the label that represents the leaf node. If the leaf node is impure then an exact NN search will be performed between the current feature vector and the entirety of the feature vectors represented in that leaf node. The current feature vector will then be labeled by whichever feature vector in the node that it is found to be closest to. Figure 6-2 shows two image segmentations performed by this system.



(a) (b)

(c) (d)

Figure 6-2. Typical segmentation results of the system under supervised learning using an a-NN search tree. (a) West side of the hallway. (b) Segmented image of (a). (c) East side of the hallway. (d) Segmented image of (c).[1]

*C.* Unsupervised Learning Using MST

In the work of [2] the use of the MST for unsupervised learning was introduced. However the work of [3] improved this implementation and will therefore be the basis for comparison in this review. To start, it is important to mention the use of this vision system was human motion segmentation. Therefore the resolution of the segmented images was increased by reducing the patch sizes to 7x7. This was the only change made to the feature extraction. The next step was to show that the system was able to be accurately trained using unsupervised learning methods.

The unsupervised learning technique used was the MST. Because the minimum spanning tree requires a preset number of cuts for the algorithm to work, a method for autonomously determining this number was devised [3]. Once the number of cuts

necessary was known the MST then clustered the data. Once the data was clustered an a-NN search tree was created for use in segmenting the images. An example of this method of segmenting the environment is shown in Figure 6-3. These images show this

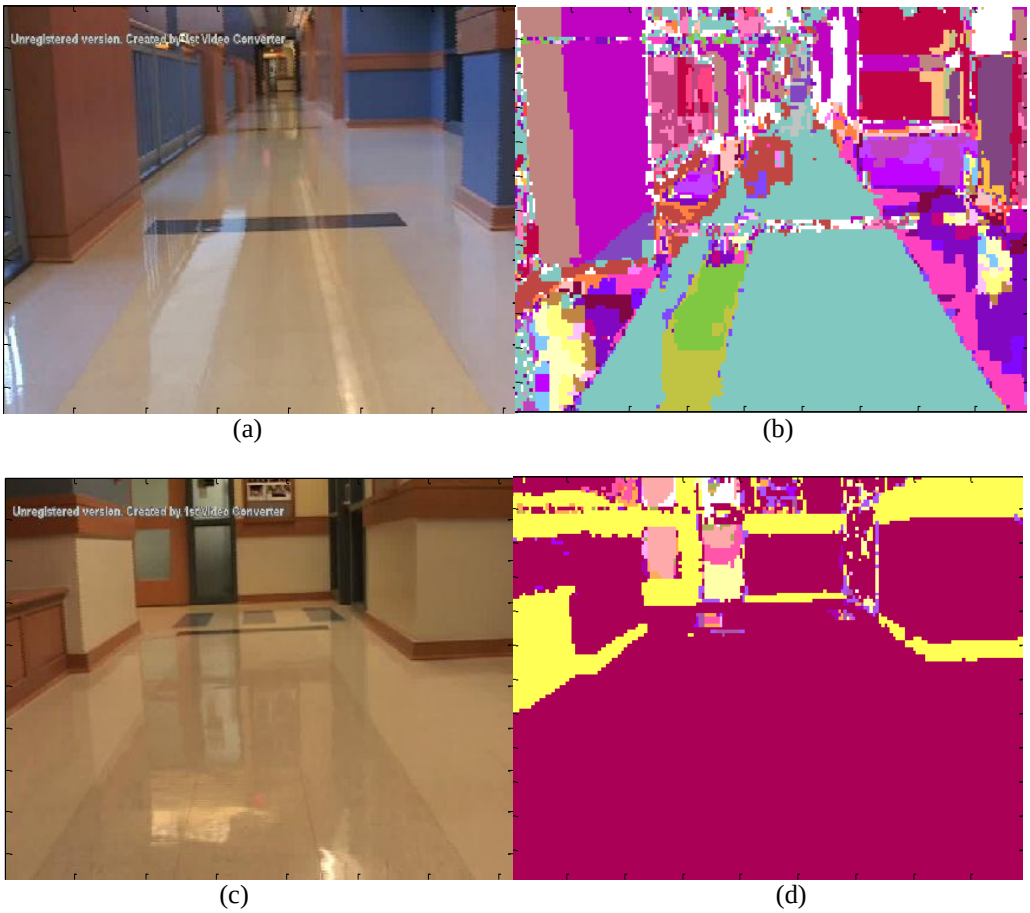



Figure 6-3 Natural Scene Segmentation Examples; (a) Indoor Atrium, (b) Indoor Atrium Segmentation, (c) Indoor Jacob Hall, (d) Indoor Jacob Hall Segmentation [3]

method being applied in the same hallway as the images taken in Figure 6-2. The segmentations are quite comparable with those in Figure 6-3 coming from a completely autonomous system.

*C.* Supervised Learning Using MLE Search Tree for Segmentation

The goal of the work in [4] was to speed up the image segmentation. Therefore supervised learning was used and a MLE search tree was implemented. This search tree is created in the same manner as the a-NN. The only difference is how the information in the tree is used. In the case of the MLE search tree, when an impure leaf node is reached, instead of performing an exact NN search, the label that has the highest number of feature vectors present in that leaf node is automatically assigned to represent the current patch. Figure 6-4 shows an example of an image being segmented by both an a-NN search tree

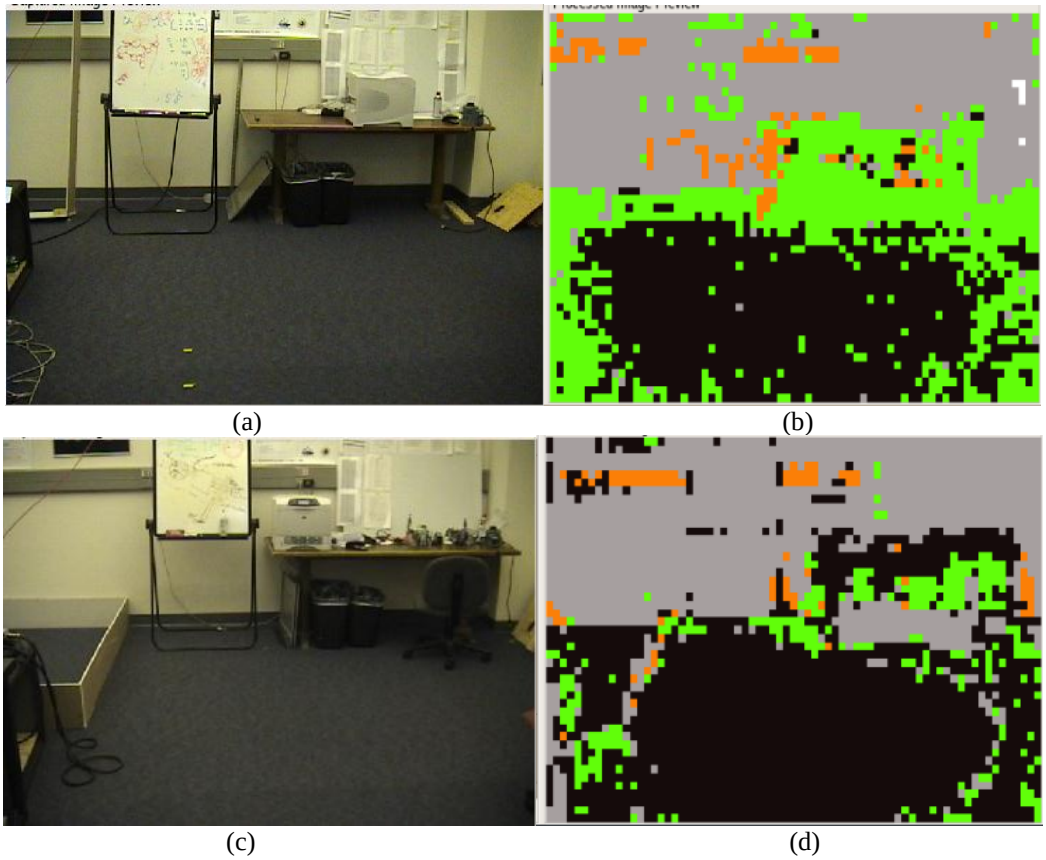



Figure 6-4 (a) Robotics lab in Featheringill Hall at Vanderbilt University. (b) Segmented (a) using a-NN search tree (c) Same as (a) (d) segmented (c) using MLE search tree [4]

as well as a MLE search tree.

*D.* Unsupervised Learning Using K-means clustering with NM segmentation Implementation

After the success of the MST and search tree methods of processing the data, the focus became speed in both training the system and segmenting an image. In both cases it was found that although the results provided by the methods used were good, the time it took to employ those methods was not acceptable for real time applications. Because of this the decision was made to use fast known clustering and segmentation algorithms and observe the results. The goal was to significantly reduce computational cost with little or not sacrifice of performance. This led to the use of a K=40 K-means clustering being used to train the data with a cluster merging technique to deal with over-segmentation, and the use of NM for segmenting the images [59].

To validate the use of NM an image was segmented using a supervised training database and comparing the NN segmentation to the NM segmentation. The results are shown in Figure 6-5. Note that the display labels of each of the percepts are randomly set

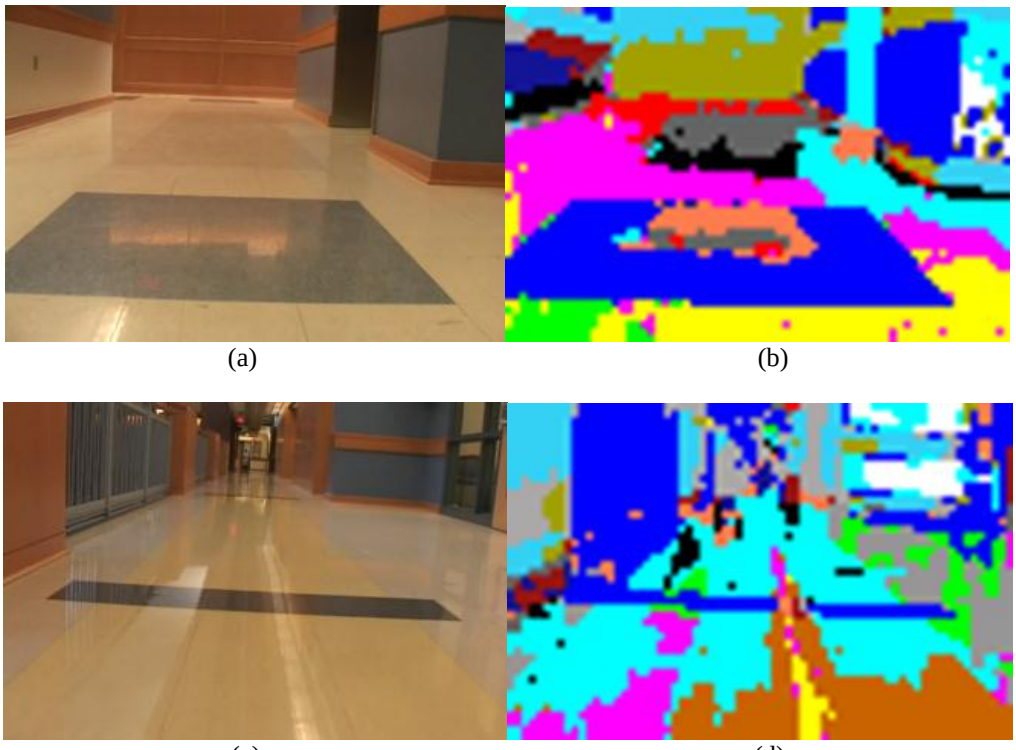
(a)

(b) (c)

Figure 6-5(a) Image of a hallway in Featheringill hall at Vanderbilt University (b) Segmentation of (a) using NN (c) Segmentation of (a) using NM [59]

and the segmented structures are what indicate the similarities in the processes. Figure 6-5 (b) is the image segmented using NN and Figure 6-5 (c) is the image segmented using the NM. These images show that the NM segmentation used on the very high dimensional data can perform very well.

The application of the vision system in this case was a location recognition system that was successfully able to determine what hallway it was in based on the percepts segmented. The results of the image segmentation are shown in Figure 6-6.



(a) (b)

(c) (d)

Figure 6-6. (a) image of hallway 1 in FGH. (b) segmented image of (a) using data trained with K-means clustering and NM segmentation. (c) image of hallway 2. (d)segmented image of (c) using data trained with K-means clustering and NM segmentation [59]

## Comparing Results

*A.* Results from Supervised Training, MST, and K-means Training

Throughout these works three types of training methods have been performed. Those methods are supervised learning, MST, and K-means. The supervised learning was first used to determine if the use of this type of high dimensionality for the purpose of object segmentation was even useful. As shown in [1] and Figure 6-2, we conclude that it is. The next step was to develop an autonomous means of training this system. Therefore one of the most rigorous methods of training, the MST, was applied to the system [2,3]. Again this method demonstrated the ability to segment the percepts in the environment. As will be discussed, the main drawback to both methods was the time required to train the system. With strictly speed in mind the next questions became how fast can the system be trained and what will the quality of the segmentation be? So the fastest, albeit not best, method was used, K-means [59]. The degraded quality can be seen in Fig.6 however based on the time improvements and the percepts still being determinable, the loss of quality is acceptable.

The timing results of these works as reported in [3,59] and through personal experience in supervising the training of the system in [4] are shown in Table 6-1. The

Table 6-1: Training phase times

| Training Phase | Time (hrs) |
|---|---|
| Supervised Learning | |
| Gathering Data | >12 |
| Creating Search Tree | >10 |
| Total | >34 |
| Unsupervised MST System | |
| Thinning | ~24 |
| MST | ~40 |
| Creating Search Tree | ~36 |
| Total | ~100 |
| Unsupervised K-means System | |
| Thinning | ~0.16 |
| K-means | ~0.16 |
| Total | ~0.32 |

results for the supervised learning show that in total this process took greater than 22 hours. This was a very rigorous process of selecting and labeling parts of multiple images. Creating the search tree could also require much longer to process as it was totally dependant on the number of feature vectors extracted by the user. The times reported were those found to adequately train the system for use on the images in Figure 6-4. The benefit of this method is the quality of the results. Figure 6-2 provides the best example of this. It was found in [4] that large amount of white in the room used for Figure 6-4 made this room very difficult to segment using just the HSV domain.

The MST method of training required three steps to have a fully training system. These steps are thinning the collected data, using the MST to label the data, and finally creating a search tree from the data. This method, as reported in [3], required ~100 hours to complete. The quality of this method was shown to be quite high as it was used for human motion segmentation in [3]. Figure 6-3 shows this. Again though, the length of

time required to train this system is very high. If the initial training is not adequate then it will require another 4 days to retrain the system.

Finally, the K-means training method required ~20 minutes to train the system. Part of this is due to using a different thinning method, but even if this thinning method were applied to the MST training method that would still require ~76 hours. So it is still much faster for generating a useable trained database. In this case, the weakness was segmentation quality. As shown in Figure 6-6, this method produces far more cluttered images. Although the images do provide enough information for location recognition and methods for dealing with the quality have been presented in [59,62].

Based on the results observed, the best method of training will depend on the goal of the system. If a user can train the system and time is not an issue then the supervised learning is the best means as this is also the most likely means of getting good results. Again if time is not an issue, but there is no user to supervise the training, then the MST would be the route to take. Finally, if time is of great concern and quality can be sacrificed then the K-means training method should be implemented.

It should be noted that other methods of training the system are being looked into, most notably with the use of O-Clusters [64]. This method will aim to balance the time requirements with the quality of the segmentation as an unsupervised training method.

*B.* Results from NN, a-NN, MLE, and NM Segmentations

The very high dimensional feature space has been implemented for image segmentation using four different classification methods. The NN method is the most rigorous and is largely regarded as one of the most precise methods of classifying data. The a-NN search tree is a well known method of speeding the NN process up while only

losing a small amount of performance. The MLE search tree was another attempt to speed the system performance up. While the NM is a very fast method of segmenting the data, it is typically viewed as an imprecise method of classification. The results of using these methods are shown in Table 6-2.

Table 6-2: Image segmentation times

| **Technique** | **Time (msec)** |
|---|---|
| Nearest Neighbor | >7,200,000 |
| Approximate Nearest Neighbor | ~12,000 |
| MLE | ~5,000 |
| Nearest Mean | ~50-100 |

The trained databases used to generate these times came from both supervised and unsupervised learning. However based on the observation that the supervised learning method will provide the most accurate database from which to segment the images, some visual comparisons can be made. By comparing Figures 6-2, 6-4, and 6-5 it can be seen that the NN, a-NN, and NM methods provide the best visual segmentation. The figures also show that the discrepancy in segmentation quality, using NM in Figure 6-5 (c) and Figure 6-6(b) (d), are based on the training method more then the segmentation method. Therefore, the true comparison of these methods will be based on the times it takes to segment an image.

The results of these methods indicate that the NM is the best method for use. This method can operate at both real time speeds and provide a comparable segmentation to NN, as shown in Figure 6-5. The real problem with the use of NM is that it is dependant on the number of percepts learned. For the images used in generating the time provided, there were 12-18 segmented percepts.

This does not mean that the other methods do not have their uses as well. If a system is not time critical then the NN or a-NN methods will still provide very high quality results.

## Conclusions and Future Work

This paper describes multiple implementations of a very high dimensional feature space as applied to image segmentation. Multiple methods of training the system were described, including both supervised learning and unsupervised learning methods involving a MST approach and K- means approach. Also multiple means of segmenting the trained data were provided. These methods include NN, a-NN search tree, MLE search tree, and NM classifications. As is well known a supervised learning approach provides the most accurate training database, but this data is still able to be classified using autonomous means. If time is of no concern then the MST is a good implementation that can provide accurate results, and if time is of more concern then a K-means approach can still provide enough information for the system to provide useful segmentations.

In the case of segmenting the images, it was shown that time was the most important factor since all of the methods provided good visual results depending on the training method used. It was found that the NM method provided the fastest results allowing a system to run at near frame rate speeds. The bottleneck to this method being the number of percepts that need to be considered in the segmentation. This can be dealt with though through clever implementation and proper use of the CUDA architecture.

The next step in the work will to be to continue to improve the unsupervised training method. There are numerous methods for classifying large databases that use very high dimensional data and those methods should be investigated. As far as the image segmentation goes, the NM method appears to work quite well leaving the next step to be optimizing the implementation for parallel implementation on modern graphic processing units.

# CHAPTER VII

# CONCLUSIONS

As shown in Chapter II, this visual system has been built to deal with unmodified environments while maintaining as much robustness as possible. This began with Tugcu [1] creating the very high dimensional feature vectors used for segmentation. His iteration of the system used a supervised learning approach to create an approximate nearest neighbor search tree for segmenting the images. This was combined with a working memory toolkit to give the robotic system the ability to learn important percepts.

The next work on this system was done by Wang [2]. Wang built upon what Tugcu had done and developed a means for the system to create its own percepts using a MST. On top of that, Wang added a technique to discover novel objects in the environment. These additions gave the system a semi-autonomous means of segmenting the environment. The lack of autonomy comes from the parameters that Wang had to manipulate for the MST to work well.

From the work performed by Wang, Hunter [3] was able to improve the autonomy of the MST, show the importance of using normalized vectors, and demonstrate that the system can perform well in multiple environments. Hunter used a threshold that was found based on the distances of the training vectors used as shown in Figure 28. He was also able to show that by using normalized feature vectors the full power of the very high dimensional feature space as a means of segmentation could be realized. His final contribution was showing that this system could be used to segment natural outdoor areas.

The last modification of this system, prior to this work, was performed by Costello [4]. This work focused on expanding the power of the system in terms of properly incorporating new percepts, adding change detection to the system, and using a MLE approach to speed up the segmentation process. When this work was completed the approximate nearest neighbor search tree was able to expand its leaf nodes on the fly when new percepts were added. It was also able to detect when a new object had entered the viewing space and determine if any object in the viewing space had been moved. Finally the system was able to segment images faster using a MLE based tree.

After all of these works had been completed the largest problem with the system was still the time it required to process a single image. The MLE based tree was able to segment an image every five seconds. This was far too long for a real time system to process each image. Therefore the first step necessary was to speed the processing time up significantly. This was done by porting the segmentation process to a GPGPU. Porting the system onto the GPGPU also resulting in overhauling the technique used for image segmentation. As shown in Chapter III, the image segmentation now only requires a single mean vector to represent each potential percept leading to a processing time of ~10-20 images/sec or ~50-100 msec.

The next step taken in the current work was to add location recognition. As explained in Chapter III new techniques were implemented to speed up the training phase compared to the work done in [3]. Therefore instead of using an MST to segment the objects, an over-segmentation of the training vectors was done using K-means clustering. These clusters were then recombined using a threshold based on the distances between

them. This process resulted in repeatedly providing ~16-19 percepts which is roughly the same number of percepts the human user found when doing a supervised training.

While the percepts were found, the system then created its own sub areas within the training region. This division of the training area is shown in Figure 38. This was done by comparing each image to prior images and determining how much the overall view of the area had changed. If significant change had taken place then a new area was created, resulting in six distinct areas being found.

Finally after the system had generated its own percepts and understanding of the training regions, each region was modeled based on the presence of the percepts. The mean feature vector was calculated for each percept in each region and combined to form the local percept mean vectors. Once this had been found the system was able to take new images and determine what region it was in based solely on what was seen. As explained, the system was also capable of updating its knowledge base to improve performance.

Although the idea of using a model of an area is not novel, the method and approach used here is. This method uses a different philosophy from many others. Where the other types of systems mentioned (SLAM, landmark detection, and template matching) use specific techniques for locating what region the robots believe they are in, answering the question "where am I", this system attempts to understand where it is through segmentation and then location recognition in a manner we feel is closer to how humans do it. Meaning the system sees all the percepts in the room and then decides where it is, instead of seeking out specific landmarks or using an overall representation of the space. This type of rudimentary scene understanding is at the root of this work. As

stated in Chapter I, this work is intended to be the base of a system that is capable of continuing to develop and learn autonomously. To that end, the ability to generate percepts and divide areas into smaller more meaningful areas autonomously and then use that information for segmentation and location recognition is very important. As stated, instead of seeking out a specific object in an area or manipulating the area to create a model that carries no other inherent information; this work aims to take advantage of all the information present. Thus opening the door for the system to learn further about the region that it is in.

Furthermore, the system has the ability to detect reflections created by distant light sources, further classify the tracked percept blobs, detect novel objects, and detect novel areas. The reflection detection is performed by tracking the behavior of a percept blob over an extended period of time. It has been observed that reflections that are created by distant light sources will appear to remain in the same location in the segmented images regardless of the motion of the robot. Therefore, by tracking the nonmoving percept blobs reflections were found to be very identifiable.

Secondly, due to the large number of reflections in the areas segmented, a method of further classifying the percepts needed to be developed. The classifications created for each blob tracked are as follows: actual percept, probably a percept, probably an aberration of light, aberration of light. Each classification is determined based on the behavior of the percept blobs. Then, left for future work, based on the classifications the blobs should be able to be combined. An example would be if a percept blob suddenly disappeared and was classified as probably an aberration of light. Then where that percept blob disappeared a new blob formed with a different percept label and was found

to behave like a percept from there out. When this situation occurs it would be possible to infer that a lighting change was covering the percept that exists and when the robot got close enough the percept became visible. Then a connection between the two labels could be created.

The third post location recognition addition made to the system was novel object detection. This is performed similar to the threshold method developed in [2],[4]. The threshold was found by finding the average distance and standard deviation between the trained percepts and feature vectors extracted from images containing those percepts. Two times the standard deviation was then added to the mean to initially set the threshold. This threshold was then experimentally increased to be sure that only truly novel objects would be detected. In addition to the distance threshold needing to be crossed the size of the novel object must exceed 50 connected pixels. This ensured that only dominant percepts from the environment were found. This method has demonstrated the ability to accurately and reliably detect novel objects in the region. The next step for this aspect of the work, which has been left to future works, is to include the novel objects into the database as both global percepts and within the models of each local region to help define the local regions that the novel objects have been determined to exist in.

The final addition to the system is the ability to detect novel regions. The method for accomplishing this includes three criteria needing to be met. First a preset distance threshold for the regions must be exceeded. Secondly, 25 images must exceed that threshold with the 25 images representing at lease a 25 foot area. Finally, the dominant region that classifies all the images that cross the threshold can not represent more the

70% of the images. This ensures that there is sufficient "confusion" within the system as to where it believes it is. If all three of these criteria are met, then a novel region exists within the set of images exceeding the threshold. This method was shown to detect novel regions in three different scenarios. The first scenario involved the system detected that two areas that had already been trained and classified needed to be further segmented. The second scenario involved finding a novel region that consisted of known percepts. Although, it should be noted that this method failed when the untrained area tested appeared to be exactly the same as the area the system was trained on. It did, however, work when the area was made up of known objects, but the region was a distinct region. Finally the system was able to detect novel regions that consisted of completely novel objects.

So to sum up this work I have created a system capable of the following:

1. Unsupervised learning of percepts (done)
    a. Decide number of clusters
    b. Find clusters
    c. Train system
    d. Segment images on GPGPU
2. Location detection (done)
    a. Detect distinct regions
    b. Model regions
    c. Identify regions
3. Motion tracking (done)
    a. Reflection detection

b. Model change in appearance of objects with change in distance

c. Detect novel objects

d. Detect new areas

With these additions the system is able to reliably segment and locate itself within an area in a manner closer to how humans locate themselves. It is also able to further classify the percepts that it sees and can determine if what it sees is an actual percept or an aberration of light. Finally the system is able to detect novel object and novel areas.

# REFERENCES


[1] Tugcu, M., “A computational neuroscience model with application to robot perceptual learning”, Ph.D. Dissertation, Vanderbilt University, August 2007.

[2] Wang, X., “A Vision-Based Perceptual Learning System for Autonomous Mobile Robot”, Ph.D. Dissertation, Vanderbilt University, August 2007.

[3] Hunter,J., “Human Action Segmentation and Recognition with a High Dimensional Single Camera System”, Ph.D. Dissertation, Vanderbilt University, May 2009.

[4] Costello, C., “Change-Detection with Limited Situational Awareness”, Masters Thesis, Vanderbilt University, August 2008.

[5] Hunter, J.E., Tugcu, M., Wang, X., Costello, C., and Wilkes, D.M., “Exploiting sparse representations in very high-dimensional feature spaces obtained from patch-based processing”, *Machine Vision and Applications*, 2010

[6] Thrun, S.; Thayer, S.; Whittaker, W.; Baker, C.; Burgard, W.; Ferguson, D.; Hahnel, D.; Montemerlo, D.; Morris, A.; Omohundro, Z.; Reverte, C.; Whittaker W; , "Autonomous exploration and mapping of abandoned mines," *Robotics & Automation Magazine, IEEE* , vol.11, no.4, pp. 79- 91, Dec. 2004
doi: 10.1109/MRA.2004.1371614
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1371614&isnumber=29987

[7] D. Ferguson, A. Morris, D. Hähnel, C. Baker, Z. Omohundro, C. Reverte, S. Thayer, W. Whittaker, W. Whittaker, W. Burgard, and S. Thrun. An autonomous robotic system for mapping abandoned mines. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Proceedings of Conference on Neural Information Processing Systems* (NIPS). MIT Press, 2003.

[8] J.C. Cox. A revier of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53-66, 1993

[9] M. Bosse, P. Newman, M. Soika, W. Feiten, J. Leonard, and S. Teller. An atlas framework for scalable mapping. In *Proceedings fo the IEEE International Conference on Robotics and Automation* (ICRA), 2003.

[10] P. Besl and N. McKay. A method for registration of 3d shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239-256, 1992.

[11] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333-349, 1997

[12] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation* (CIRA), 2000

[13] Yong-Ju Lee; Jae-Bok Song; , "Visual SLAM in indoor environments using autonomous detection and registration of objects," *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on* , vol., no., pp.671-676, 20-22 Aug. 2008
doi: 10.1109/MFI.2008.4648022
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4648022&isnumber=4648003

[14] D.G. Lowe, "Distinctive image features from scale invariant keypoints," *Int'l Journal of Computer Vision*, vol. 60, no.2, pp. 91-110, November, 2004.

[15] P. Bao, L. Zhang, and Xaiolin Wu, "Canny Edge Detection Enhancement by Scale Multiplication," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol.27, No.9, pp.1485-1490, September, 2005.

[16] J. Canny, "A Computational Approach to Edge Detection". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No.6, November, 1986

[17] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csobra, "A solution to the Simultaneous Loclization and Map Building Problem," *IEEE Trans. On Robotics and Automation*, vol. 17, no. 3, pp. 229-241, June, 2001.

[18] Frintrop, S.; Jensfelt, P.; Christensen, H.I.; , "Attentional Landmark Selection for Visual SLAM," *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* , vol., no., pp.2582-2587, 9-15 Oct. 2006
doi: 10.1109/IROS.2006.281711
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4058779&isnumber=4058335

[19] Angeli, A.; Doncieux, S.; Meyer, J.-A.; Filliat, D.; , "Visual topological SLAM and global localization," *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on* , vol., no., pp.4300-4305, 12-17 May 2009
doi: 10.1109/ROBOT.2009.5152501
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5152501&isnumber=5152175

[20] P. Jansen-Osmann, G. Wiedenbauer, "The representation of landmarks and routes in children and adults: A study in a virtual environment", *Journal of Environmental Psychology*, Volume 24, Issue 3, September 2004, Pages 347-357

[21] Kai Ni; Kannan, A.; Criminisi, A.; Winn, J.; , "Epitomic location recognition," *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* , vol.,no.,pp.1-8,23-28,June,2008
doi:10.1109/CVPR.2008.4587585
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4587585&isnumber=4587335

[22] N. Jojic, B. Frey, and A. Kannan, "Epitomic Analysis of Appearance and Shape," *Proc. Int'l Conf. Computer Vision*, 2003.

[23] B. Johansson and R. Cipolla, "A System for automatic Pose-Estimation from a Single Image in a City Scene," *Proc. Int'l Assoc. of Science and Technology for Developmental int'l Conf. Signal Processing*, Pattern Recognition and Applications, 2002.

[24] B.J. Frey, N. Jojic, "Learning the "Epitome" of an Image", Technical Report PSI-2002-14, University of Toronto, November 10, 2002.

[25] A. Criminisi, J. Shotton, A. Blake, C. Rother, and P.H.S. Torr, "Efficient Dense Stereo with Occlusions by Four-State Dynamic Programming," *Int'l J. Computer Vision*, 2006.

[26] A. Torralba, K.P. Murphy, W.T. Freeman, and M.A. Rubin, "Context-Based Vision System for Place and Object Recognition," *Proc. Int'l Conf. Computer Vision,* vol. 1, pp. 273-280, 2003.

[27] E.P. Simoncelli and W. T. Freeman. "The steerable pyramid: A flexible architecture for mulit-scale derivative computation. In *2nd IEEE Int'l. Conf. on Image Processing*, 1995.

[28] Tugcu, M., Wang, X., Hunter, J.E., Phillips, J., Noelle, D., and Wilkes, D. M. "A computational Neuroscience model of working memory with application to robot perceptual learning", *Third IASTED International Conference on Computational Intelligence* (CI), Banff, Alberta, Canada, July 2-4 2007.

[29] Gabor, D., 1946. Theory of communications. *Journal of Institute of Electrical Engineering*, vol. 93, pp. 429-457.

[30] Phillips J.L. & Noelle, D.C. (2005). A biologically inspired working memory framework for robots. Proc. *of the 27th Annual Meeting of the Cognitive Science Society*, Stresa, Italy, July,

[31] Wilkes, D.M., M. Tugcu, J.E. Hunter, and D. Noelle, "Working Memory and Perception", Proceedings of 14th Annual IEEE International Workshop on Robot

and Human Interactive Communication (RO-MAN 2005), Nashville, TN, August 13-15, 2005, pp 686-691, 2005.

[32] J. Wolf, W. Bugard, and H. Burkhardt. “Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In *Proc. Of the IEEE Int. Conf. on Robotics & Automation* (ICRA), 2002.

[33] Kosecka, J.; Liang Zhou; Barber, P.; Duric, Z.; , "Qualitative image based localization in indoors environments," *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* , vol.2, no., pp. II-3-II-8 vol.2, 18-20 June 2003

[34] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data. Prentice Hall, 1988

[35] Koku, A.B., “Egocentric Navigation and its Applications”, Ph.D Dissertation, Vanderbilt University, May 2003

[36] Bishop, C. M., (2006). Pattern Recognition and Machine Learning. Singapore: Springer.

[37] “NVIDIA CUDA C Programming Guide” http://developer.nvidia.com/object/cuda_3_1_downloads.html, website accessed June 2010.

[38] Li, F. and Košecká, J., “Probabilistic location recognition using reduced feature set”, Technical Report GMU-CS-TR-2005-2, George Mason University, 2005.

[39] Heth, C.D., Cornell, E.H., and Alberts, D.M., “Differential use of landmarks by 8- and 12-year-old children during route reversal navigation”, *Journal of Environmental Psychology*, 1997, 17, 199-213.

[40] Cornell, E.H., Heth, C.D. & Alberts, D.M. (1994). Place recognition and way finding by children and adults. Memory & Cognition 22, 633-643.

[41] Cornell, E.H., Heth, C.D., & Broda, L.S. (1989). Children’s way finding: Response to instructions to use environmental landmarks. Developmental Psychology 25, 755-764.

[42] Escolano, F.; Bonev, B.; Suau, P.; Aguilar, W.; Frauel, Y.; Saez, J.M.; Cazorla, M.; , "Contextual visual localization: cascaded submap classification, optimized saliency detection, and fast view matching," *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on* , vol., no., pp.1715-1722, Oct. 29 2007-Nov. 2 2007

[43] Magliano, J.P., Cohen, R., Allen, G.L., and Rodrigue, J.R., "The impact of a wayfinder's goal on learning a new environment: Different types of spatial knowledge as goals", *Journal of Envvironmental Psychology*, 1995, 15, 65-75.

[44] D. Hähnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Proceedings of the Conference on Intelligent Robots and Systems* (IROS), Lausanne, Switzerland, 2002.

[45] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335-363, 2001.

[46] Y. Bar-Shalom and X.-R. Li. Estimation and Tracking description, results and lessons learned. *International Journal of Robotics Research*, 18(7):621-649, 1999.

[47] H. M. Kalayeh and D. A. Landgrebe, Predicting the Required Number of Training Samples, PAMI(5), No. 6, pp. 664-666, November 1983.

[48] Shepard, R.N. (1964). Attention and the metric structure of the stimulus space. Journal of Mathematical Psychology, 1:54-87

[49] Radovanović, M., Nanopoulos, A.,and Ivanović, M., "Nearest neighbors in high-dimensionaldata: The emergence and influence of hubs." In Proceedings of the 26th International Conference on Machine Learning (ICML), pages 865-872,2009.

[50] Radovanović, M., Nanopoulos, A.,and Ivanović, M., "Hubs in space: Popular nearest neighbors in high-dimensional data." Journal of Machine Learning Research 11 (2010) 2487-2531

[51] Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U., "When is "nearest neighbor" meaningful?" In Proceedings of the 7th International Conference on Database Theory (ICDT), volume 1540 of Lecture Notes in Computer Science, pages 217-235. Springer, 1999.

[52] Nadimi, S.; Bhanu, B.; , "Physics-Based Cooperative Sensor Fusion for Moving Object Detection," *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on* , vol., no., pp. 108, 27-02 June 2004
doi: 10.1109/CVPR.2004.144
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1384902&isnumber=30163

[53] Papageorgiou, C.; Poggio, T.; , "A pattern classification approach to dynamical object detection," *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* , vol.2, no., pp.1223-1228 vol.2, 1999
doi: 10.1109/ICCV.1999.790420
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=790420&isnumber=17141

[54] Junyeong Yang; Hyeran Byun; , "A Combination of Generative and Discriminative Approaches to Object Detection," *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on* , vol.3, no., pp.249-253, 0-0 0
doi: 10.1109/ICPR.2006.46
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1699513&isnumber=35819

[55] Pandey, S.; Kumar, S.; , "A novel approach to automatic object detection using intensity pattern recognition in YCbCr Color space," *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on* , vol., no., pp.1274-1277, 7-9 July 2008
doi: 10.1109/ICALIP.2008.4590147
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4590147&isnumber=4589950

[56] Bellman, R.E., Adaptive Control Processes. Princton University Press (1961)

[57] Tan, P., Steinbach, M., and Kumar, V., Introduction to Data Mining. Addison Wesley, 2005.

[58] Chapelle, O., Schölkopf, B., and Zien, A., (Eds). (2006). Semi-supervised learning. The MIT Press.

[59] Costello, C., Wilkes, D.M., "Unsupervised modeling of an indoor environment for a developmental location recognition system", Submitted for review to Transactions on Autonomous Mental Development

[60] Evangelista, M.J., Embrechts, M.J., and Szymanski, B.K., "Taming the curse of dimensionality in kernels and novelty detection. In Abraham, A., Baets, B., Koppen, M., and Nickolay, B., editors, Applied Soft Computing Technologies: The Challenge of Complexity, volume 34 of Advances in Soft Computing, pages 425-438. Springer 2006.

[61] Francios, D., High-dimensional Data Analysis: Optimal Metrics and Feature Selection. PhD thesis, Université catholique de Louvain, Louvain, Belguim, 2007.

[62] Costello, C., D.M., "Percept classification, novel object detection, and novel area detection for a vision based developmental location recognition system" unpublished.

[63] Costello, C., Tugcu, M., Xiaochun, W., Hunter, J.,Wilkes, D.M., "A review for efficient use of very high dimensional feature space for percept classification" unpublished

[64] Milenova, B.L.; Campos, M.M.; , "O-Cluster: scalable clustering of large high dimensional data sets," *Data Mining, 2002. ICDM 2002. Proceedings. 2002 IEEE*

*International Conference on* , vol., no., pp. 290- 297, 2002 doi: 10.1109/ICDM.2002.1183915 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1183915&isnumber=26564

[65] Wang, X. et al., Exploration of configural representation in landmark learning using working memory, Pattern Recognition Lett. (2008), doi:10.1016/j.patrec.2008.09.002

[66] Chen, W., Shi, Y.Q., Xuan, G., 2007. Identifying computer graphics using HSV color model and statistical moments of characteristic functions. IEEE Int. Conf. Multimedia Exp., 1123–1126.

[67] Serrano, N., Savakis, A., Luo, J., 2002. A computationally efficient approach to indoor/outdoor scene classification. In: International Conference on Pattern Recognition 2002, Quebec City, Canada.

[68] Serrano, N., Savakis, A.E., Luo, J., 2004. Improved scene classification using efficient low-level features and semantic cues. Pattern Recognition 37, 1773–1784.

[69] Vlassis, N., Motomura, Y., Krose, B. 2000. Supervised linear feature extraction for mobile robot localization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00), pp. 2979–2984.

[70] Piater, J.H. and Grupen, R.A., "Feature learning for recognition with Bayesian networks," *Proc. ICPR 2004,* pp. 17-20, August, 2004.

[71] Bredeche, N., Z. Z. Shi, et al. (2006). "Perceptual learning and abstraction in machine learning: An application to autonomous robotics." Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews **36**(2): 172-181.

[72] Duda, R. O., P. E. Hart, et al. (2001). Pattern Classification, Wiley-Interscience.

[73] Puzicha, J., Hofmann, T. & Buhmann, J.M. (1999). Histogram clustering for unsupervised segmentation and image retrieval. *Pattern Recognition Letters*, 20, 899-909.

[74] Kaufman, L., and Roussseeuw, P.J., 1990. *Finding Groups in Data: an Introduction to Clustering Analysis*. John Wiley & Sons.

[75] Ng, R., and Han, J., 1994. Efficient and effective clustering method for spatial data mining. In *Proc. of the 20th VLDB Conference*, Santiago, Chile.

[76] El-Sonbaty, Y., Ismail, M.A., and Farouk, M., 2004. An efficient density based clustering algorithm for large database. In *Proc. of the 16th IEEE Int'l Conference on Tools with Artificial Intelligence*.

[77] Markou, M. & Singh, S. (2003). Novelty detection: a review-part1: statistical approaches. *Signal Processing*, 83, 2481-2497.

[78] Markou, M. & Singh, S. (2003). Novelty detection: a review-part2: neural network based approaches. *Signal Processing*, 83, 2499-2521.

[79] Fleischer, J. & Marsland, S. (2002). Learning to Autonomously Select Landmarks for Navigation and Communication. In *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*. Pp, 151-160. MIT Press.