

Object Recognition using Fuzzy Membership Rules

In 2004, we began working to create an intelligent vision system. To define the system as intelligent were striving for 5 main characteristics. In this update report, we will describe our research through 2005.

1. The system should learn broad categories of objects by learning from experience. By broad categories, we mean general classes of objects that may naturally be encountered in an environment. Such as trees, sidewalks, grass, brick walls, doors, windows, etc. We do not mean specific, unique, object identification.
2. The system should be able to explain how it is making its decision, as well as be able to justify (i.e. explain) any particular decision.
3. The system should be able to detect that new or novel elements are in a visual scene. This detection of novelty should trigger new learning. The system will ask to be taught about the new objects, thus achieving self-directed learning.
4. When appropriate, after making the general class identification, the system may employ other object recognition algorithms to identify a specific object. Such as but not limited to David Lowe's SIFT algorithm and neural networks. This general to specific identification is believed to relate to some biological approaches.
5. Eventually, the system should be able to work in an unsupervised learning mode:
 - Objects it identifies with very high confidence it will label and add to its training database.
 - Objects it identifies with lesser confidence it records for later confirmation or correction by the human "teacher".
 - Novel objects are recorded for subsequence teaching by the human "teacher".
 - As the data set increases, memory consolidation will be used.

Characteristic 1 is ongoing with constant changes in feature vectors and the addition of learned objects. The following is the work being done for Characteristics 1 and 2. We test the algorithm to determine how well these sections work when applied to different tasks.

Fuzzy information is an imprecise set of descriptions or instructions [10]. Membership functions (m_F) map the fuzzy set to an interval of 0 to 1. The membership function contains a value (grade of membership) for each component in the fuzzy set. The fuzzy operations for the membership functions that we use are as follows [11,12,13]:

$$(=) \text{Equality } A = B \Leftrightarrow m_A(x) = m_B(x) \quad (1)$$

$$(\subset) \text{Containment } A \subset B \Leftrightarrow m_A(x) \leq m_B(x) \quad (2)$$

$$(\sim) \text{Complement } m_{\bar{A}}(x) = 1 - m_A(x) \quad (3)$$

$$(\cap) \text{Intersection } m_{A \cap B}(x) = m_A(x)m_B(x) \quad (4)$$

$$(\cup) \text{Union } m_{A \cup B}(x) = m_A(x) + m_B(x) - m_A(x)m_B(x) \quad (5)$$

These are used to combine membership functions with each other.

During the training step, sections of images are selected and given a label (class) according to what the object is. The program creates a structure array called lmemory and allows the user to provide the objects to be trained. The name of the picture/ frame of video, upper left coordinate of the selection rectangle, the width of the rectangle, the height of the rectangle, and the class of the selected object are added to the lmemory structure. The features are extracted from these selected regions. The feature vector structure was originally 433 characteristics. As testing continued, some of the features were removed because of processing costs.

The first 250 of the 433 features are a histogram (or equivalently a probability density function, pdf) of color measurements for each training object. The image was converted from its original RGB format to HSV. The hue is broken into 10 bins, [0.00 0.05 0.14 0.22 0.28 0.45 0.54 0.75 0.81 0.92 1.00]. Bin 1 (orange with a bit of red) is 0.00 to 0.05, bin 2 (yellow) is 0.05 to 0.14, bin 3 (yellow-green) is 0.14 to 0.22, bin 4 (green) is 0.22 to 0.28, bin 5 (blue-green) is 0.28 to 0.45, bin 6 (blue) is 0.45 to 0.54, bin 7 (blue-violet) is 0.54 to 0.75, bin 8 (purple) is 0.75 to 0.81, bin 9 (red-violet) is 0.81 to 0.92, and bin 10 (red) is 0.92 to 1.00. The saturations and values are evenly distributed in to 5 bins each ranging from 0.00 to

1.00. The bin values are [0.00 0.20 0.40 0.60 0.80 1.00]. Each color is represented by one bin for each of the three bands. All possibilities of one bin selected from every band equals 250 different color features. The program looks at each color feature and finds the number of pixels in the region that are found. After doing this for all the color features, there are 250 numbers, each representing the number of pixels of a certain color that is in the selection region. The total number of pixels in the selection region divides these 250 numbers.

The next feature is a standard texture measurement. This feature measures the “roughness” of the area. The texture is measured from the grayscale of the image. The grayscale image is then filtered with a simple 3 by 3 Laplacian edge detector. Since there are possible negative values, the absolute value is taken of the normalized value (divided by 256 because of the 256 gray levels). The sum normalized across the entire selection region (divided by the number of pixels in the selection region) is the texture measure.

The next feature is motion detection. The motion feature used video feed to determine the amount of movement in an object. This feature also uses the 256 level grayscale image. The motion was calculated by taking the absolute value of the difference of consecutive frames in the selection region. The difference matrix now contains numbers from 0 to 255. The matrix is normalized from 0 to 1 and is summed to give the motion feature measure.

The final 180 were line features. This feature measures how much of the selection area contains lines at varying degrees. By using the radon function in MATLAB over the selection region, each degree for 180 degrees for each pixel is calculated. The maximum value for each row is taken and normalized over the selection region. These features were removed eventually, but some experiments were performed using this feature. All the features are shown in Figure 1.

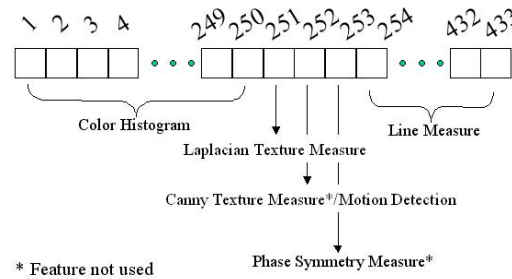


Figure 1: Feature Vector

After all of the samples from the training objects are taken, the program now has a matrix of features for each sample and an array of class labels.

Using these arrays, the Matlab treefit function returns a structure array containing all of the information about the created tree. For the classification process, you need four main items: sample set, splitting criteria, stopping criteria and pruning [14]. All the samples together are the parent node. The most discriminatory feature is used to split the parent node into two child nodes. The process is continued to create several more child nodes using the splitting function (Gini's index of diversity was used as the splitting criterion). This is continued until the stopping criteria (minimal group size) is reached for all groups that do not contain a single class. The tree is pruned to provide a completed classification tree. We used the Matlab treefit function with its default parameters.

Fifteen trees are created. The first tree contains all the features from the data and the rule is extracted. Every tree after the first removes the strongest feature (the feature at the top of the tree) from the data features, recreates the classification tree and extracts the rules. A rule consists of minterms. Minterms are the set of decisions made to reach a group of a class. The decisions are the discriminating features of the classification tree. Every tree may contain several groups of each class. The rule of a class is the union of all the minterms for that class.

$$\begin{aligned} \text{HandRule} &= ((\overline{X_{99}}) \cap (X_{13}) \cap (X_{14})) \cup \dots \\ &((\overline{X_{99}}) \cap (X_{13}) \cap (\overline{X_{14}}) \cap (X_{08})) \end{aligned} \quad (6)$$

↓

$$\text{Hand Rule} = (1 - X_{99})(X_{13})(X_{14}) + \dots \\ (1 - X_{99})(X_{13})(1 - X_{14})(X_{08}) \quad (7)$$

All the features of minterms that have value are represented. When the union rule is implemented, the final term ($-P(A \cap B)$) is not used. This part was ignored for ease of implementation. The reason this can be removed is it can be argued that the final term will become small upon calculation. Removing this term has not caused a noticeable decrease in the quality of results as compared to leaving it in. The rules are now extracted from the trees for each object class. The next step is to segment the images.

The process of extracting a feature set from a new image is the same method used to gather the training sample's features. The main difference is the moving window used to define areas in the image from which the features are extracted. Currently, the window is 15 by 15 pixels and its position increments by 10 pixels in the row and column directions. These values are arbitrary and can be changed easily, but a decrease in window position increment (increase in resolution) increases the processing time. The window increments are constrained not to exceed the image when the window reaches the borders of the image. The features of each window are extracted and put into a vector form. This vector is similar to the feature matrix in the training data except the number of rows is now the number of windows in the new image.

For each tree formed, we obtain one rule (in a sum of minterms form) for each object class. Thus, if the total number of trees formed is given by N_{trees} , then there will be N_{trees} rules for each object class. For a given object class, C_i , each rule is applied to the window feature matrix, resulting in N_{trees} result matrices for class C_i . A result matrix gives the degree of membership estimated for each region by the applied rule. What remains to be done is to combine the N_{trees} results for the given class into a single result. Viewing the individual results as fuzzy sets, we may combine them using the union of the N_{trees} results using Equation (5). However, this approach yielded poor results.

We use another approach to combining the results by switching to another paradigm of interpretation. If we take each result matrix of class C_i for each of the N_{trees} rules and divide it by the total sum of all values in the result matrix, the new normalized matrix now has a sum of values equal to 1. Thus, the matrix can be viewed as a probability density function (pdf) describing the distribution of the objects of class C_i as estimated by the associated rule, $R_j, j=1, \dots, N_{\text{trees}}$. This pdf is denoted by

$$f((x, y) \in C_i | R_j) \quad (8)$$

Then using a probabilistic assumption of equally likely rules we obtain

$$P(R_j) = \frac{1}{N_{\text{trees}}} \quad j = 1, 2, \dots, N_{\text{trees}} \quad (9)$$

and the pdfs may be combined as

$$f((x, y) \in C_i) = \sum_{k=1}^{N_{\text{trees}}} f((x, y) \in C_i | R_k) P(R_k) \quad (10)$$

This method of combining the rule outputs yielded good results. The fuzzy membership functions seem to give a more robust evaluation of the data when gathering features. The Bayesian paradigm gives a more robust expression of the features after they are combined. So by linearly combining the probability density functions of fuzzy membership rules across the entire image, the strengths of both fuzzy rules and Bayesian probability rules can be exploited.

These values can be represented visually as a saliency map by applying a grayscale colormap to display the results. The image has brighter spots for areas in the image having values that are higher compared to the other values of the set. The minterms for a class worked well if the values (brightness) of the object contrast highly with the values (brightness) of objects that are not of the class. The

resolution decrease observed in the output images is due to the reduction of each increment of the window down to one pixel representation.

The next step is to allow the detection of multiple objects in one picture. Now the program must be able to decide whether a pixel is of a certain class or of no class at all. Now the program is taking the window feature matrix and extracting the rules for all the classes. The resulting matrix (ImSc) has n (number of windows in image) rows by y (number of classes) columns.

$$\text{ImSc} = \begin{bmatrix} sc_{1,1} & sc_{1,2} & sc_{1,3} & sc_{1,4} & sc_{1,5} & sc_{1,6} & \dots \\ sc_{2,1} & sc_{2,2} & sc_{2,3} & sc_{2,4} & sc_{2,5} & sc_{2,6} & \\ sc_{3,1} & sc_{3,2} & sc_{3,3} & sc_{3,4} & sc_{3,5} & sc_{3,6} & \\ sc_{4,1} & sc_{4,2} & sc_{4,3} & sc_{4,4} & sc_{4,5} & sc_{4,6} & \\ \vdots & & & & & & \end{bmatrix}$$

score matrix (n x y)

n = number of windows in image

y = number of classes

Each index in score matrix is the value of that windows score with the class' minterms. These scores take values from 0 to 1 along classes (columns). The scores are then compared to each other in the same row. The window is decided to be a certain class if it has the largest score in the rows and it is greater than 0.15 (a threshold chosen empirically by observing graphs of the scores) and of no class if it is less than 0.15. This value can be increased to ensure only the higher scores to be recognized as different classes. The sample image in Figure 2 was taken from the psychology department experiments. The goal is to study the motion of humans when given different descriptions of the audience. It is necessary to be able to track the hand for this experiment.

The result of the sample image (Figure 3) shows the segmentation (yellow = BlueRing, green = RedRing, blue = YellowRing, purple = Hand, and red = RedStrip). The object recognition indicates detection of the hand in the image though there is no hand in view. The face of the person is represented as Hand, which is logical since the face has about the same color as the hand. As shown in the result, the red strip on the hat is correctly identified.



Figure 2: Sample Image

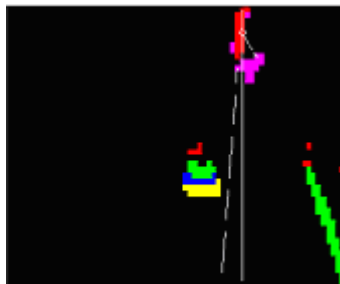


Figure 3: Segmentation Image for Sample Image

We use the fuzzy membership recognition system to discern objects in videos from the psychology department. One of the objects is the person's hand. As stated before, the ultimate goal is to study the differences the in motion of a person when demonstrating a task for different entities (humans or robots).

The main objective is to segment movies and be able to track the hand as well as the head angle. Each participant was to wear a hat with a red stripe at the center of the hat. This was to aid in discerning the direction the participant was looking. By tracking these two movements, they are able to store data on where the subject was looking and hand motion patterns. They speculated that the hand motion of the subject when told they were performing for a robot may be slower and more rigid. The information will also reveal the effectiveness of the fuzzy rules deduction and demonstrate the speed of training to new environments.

The immediate goal is to show that the recognition algorithm is effective in segmenting the objects in videos created for study by the psychology department. It can track the movements of the hand and the angle of the head, and the system is easily trained and implemented with a relatively small number of samples.

The participants are told to show either a human or a robot how to move some objects on a table to complete a task. The three tasks used were the card arrangement, towers of Hanoi, and tower building tasks. Each of the tasks has objects of different colors.

The card arrangement task is to take the cards arranged by color and shift them to be arranged by the number of white squares on them. The only limitations are that the person may only use one hand and the cards are not allowed to pass over each other.

The Tower of Hanoi task shown in Figure 5 is to move the stack of colored rings from the spindle on the right side of the image to the one on the left. A ring can only be moved to an empty spindle or on top of a larger ring. Again, the task is to be completed with one hand.

The tower building task is to take the colored blocks and construct them to a tower-like structure. The participant is to attempt to build the tower with one hand but may use both if necessary.

Two new script files were created to easily train and process the videos by consolidating the current functions. A simple regression algorithm is used to plot an estimated line for the center of the head, an estimated line for the direction of the gaze, and a marker to determine the hand position. The first two use the image segmentation of the red strip on the head. The strip is separated from the rest of the image by only considering the region where the head of the participant is. The center of the strip is considered to be the center of the head. A line is plotted going straight down from the center. The estimation of gaze uses what the algorithm determines is the red strip at the head region, and puts a best fit line to the points to determine the line of the gaze. The line of the gaze is plotted along with a vertical centerline, thus an angle of gaze from the center can be estimated. The hand marking finds the range of hue values that flesh tones fall in for the image and multiplies them by the amount of motion detected in the same image. So that the coordinates match, the flesh-tone/ motion product image is down sampled to match the segmented image coordinates. The hand position is simply the max value in the down sampled flesh-tone/motion image.

Each video was processed using training vectors from only 50 samples for each object. The collection of the 250 samples takes about 30-45 minutes to collect for each task. A huge feature space (252 dimensional) needs many samples to create solid clusters. Although the numbers of samples are only 250, the recognition of the videos is reasonably accurate. Table I shows the object classes for each task.

Table I: Class names for the three separate tasks

Towers of Hanoi task	Card arrangement task	Tower building task
BlueRing	BluePaper	GreenBlock
YellowRing	GreenPaper	PeachBlock
RedRing	PinkPaper	BlueBlock
RedStrip	RedStrip	RedStrip
Hand	Hand	Hand

The results had some noise but this is likely due to the limited number of samples. The videos themselves segmented very well considering the small number of samples. The two main areas of noise are from "inconsistent flashes" and the skin tone items in the image (i.e., PinkPaper). The inconsistent flashes are areas that are not the object it is identified to be but due to minor feature characteristics of the image scoring high enough to be considered an object. Remember the object is decided from its score

on the rules. Since the score is normalized from 0 to 1 across all the windows of the frame, some noise (i.e., lighting differences) causes a fluctuation in the scores of all the windows giving incorrect results. Also, the color features are not fine enough to separate the colors of the skin and pink cards. Integrating more samples into the training matrix can reduce the noise from the inconsistent flashes. Increasing the number of bins in the color feature space will separate very close colors in the hue space to different bins. The noise would probably be less if the number of samples were increased. The small number of samples allows for acceptable detection, and the pixels that were correctly detected can be saved and used as additional training vectors. This allows for the training vector samples to increase dramatically with each frame's results. This change can allow for analysis of the effectiveness of the fuzzy rules. If the object detection does not improve with samples, the most likely approach is to adjust the feature vector itself.

In all of the videos, the hand marking and gaze estimation worked rather well. The hand position is signified with a white x marker. Even though the face and the hand are moving, the program does not confuse the two when both are in the picture. By storing these numbers, the hand motions can be analyzed to reveal possible motions of the hand. By taking these vectors of positions in the hand, it is possible to extract changes in the direction of the hand and/or pauses. Using these to segment one motion of the hand from another, the frames where the changes occur can be displayed and analyzed. Figure 4 is an example of the motion feature displayed as an image.

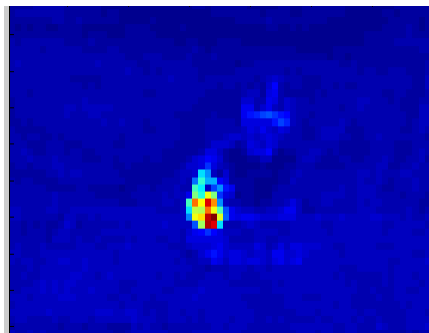


Figure 4: Motion image example

The collected hand positions give the following plots. Figure 5 is the median-filtered x-axis (column) position of the hand plotted vs frame number. As the numbers increase, the farther to the right side of the image is the hand location. Figure 6 is a derivative of the median-filtered x-axis hand motion used to aid in extracting the pauses in the x directions. Figure 7 is the median-filtered y-axis (row) position of the hand. As the numbers increase the closer to the bottom of the image is the hand location. Figure 8 is the derivative of the median-filtered y-axis hand motion used to aid in extracting the pauses in the y directions. The frames in the video where the derivative plots are both zero show where the pauses occur. The program also uses zero crossings of the hand position in the x direction and y direction to discern significant motions.

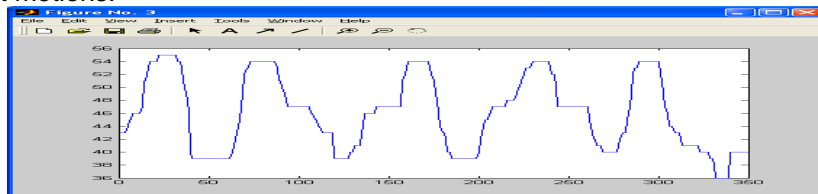


Figure 5: X-axis hand position

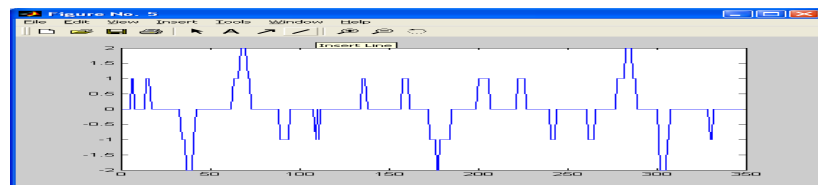


Figure 6: Derivative of x-axis hand position

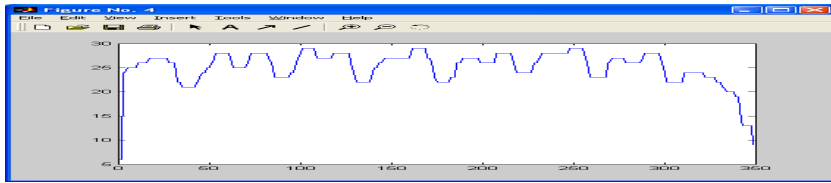


Figure 7: Y-axis hand position

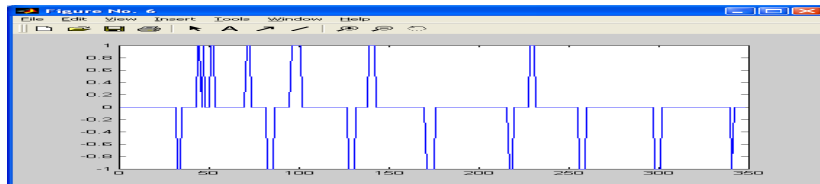


Figure 8: Derivative of y-axis hand position

For the tower of Hanoi task, the pause seemed to capture the significant motions of the subject, as show below. Figures 9 through 10 are the frames estimated to be the transition points from one motion to the next for the Tower of Hanoi video. It has been observed that some subjects that perform tasks for the robot audience will make rigid and linear movements compared to the smooth movements for a human audience.

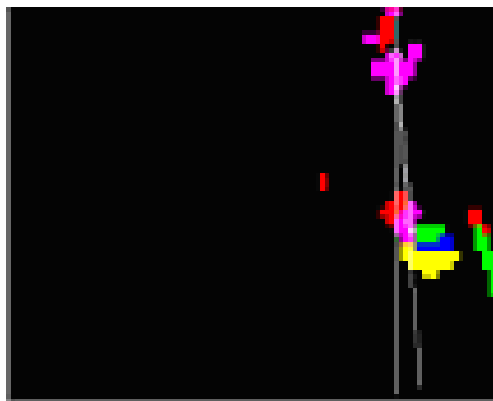


Figure 9a (left): Reaching/Grabbing the Red Ring

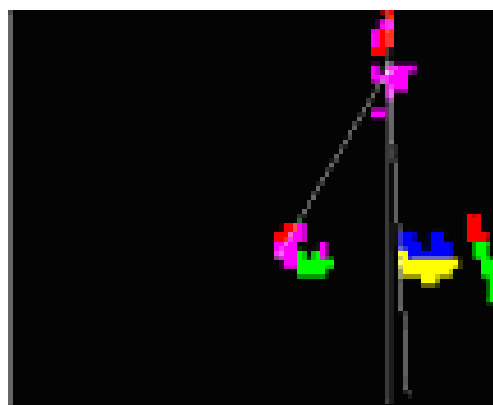


Figure 9b (right): Moving/Releasing the Red Ring

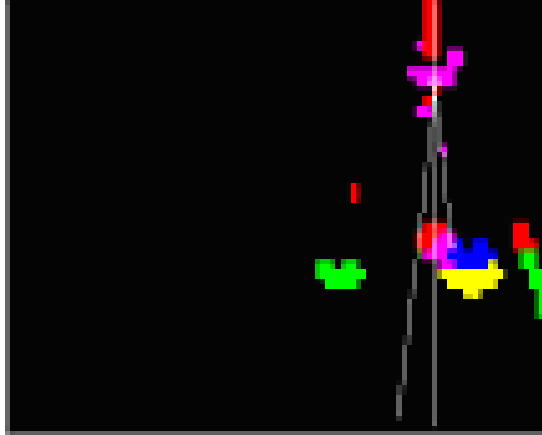


Figure 10a (left): Reaching/Grabbing the Yellow Ring

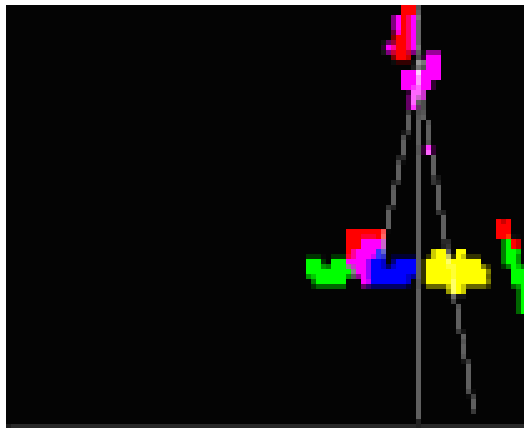


Figure 10b (right): Moving/Releasing the Yellow Ring

The use of fuzzy membership functions for object recognition is one of the better ways to develop a learning object recognition system. This system takes advantage of the classification trees, which allows it to relate to objects through the feature vectors in a manner that is easy to understand. The simple training process allows for many different features to be added. These features can be used in many interesting manners, due to the way they are stored. The motion image can be used to find when motion below the hand is at a maximum to determine if the hand is moving an object. Just as the hand's position can be stored, other objects (i.e., a red strip) can be used to track other significant motions in the image.

Some of the other tasks are performed more smoothly, which makes it necessary to develop the movement segmentation calculations for accurate segmentation. We are currently developing a method of storing the amount of motion below the hand to establish whether the hand is moving an object. This may be used to further define the significant motions.

The processing of the videos takes a long time. The eventual goal of the system is to run in real-time. This slowness may be due to MATLAB not being the optimal execution environment for image processing. The code will be ported to C++ to determine how much faster it can be processed.

The low resolution does not allow for the detection of small hand movements. Increasing the resolution increases the process time quite a bit. This problem may also be improved if the port to C++ will yield a significant decrease in processing time.

The results of object recognition in images can be used to add to the database of training vector by taking the window samples that are correctly identified. Using this method, the number of training samples will increase dramatically. As the number of training vectors grows larger, storage will become an issue. Since the vectors are mostly zero, they can be stored as a sparse vector to solve the problem immediately. In the long run, it will be necessary to create a general model for objects to conserve space.

It may be possible to use morphology to reduce noise in recognition. This process may help reduce the random noise spots that occur in the recognition process.

Combining the training sets could allow for subject identification for the same tasks. The subjects that are asked to do the same tasks may have different enough skin tones to be identified. It is likely that the color histogram would need to be able to represent a high number of colors in the skin tone.

The ability for a robot to create descriptions of a feature vector and motions may lead to instruction. These instructions can be used to teach other robots how to do certain tasks. Their feature vectors will identify objects. The human's hand can be tracked to observe the behavior. The robot can track its own hand and mimic the behavior observed from the human. In terms of navigation, a robot can observe the features of a pathway and use a simple cost/reward system to learn about objects in its environment.

- Submitted by Jonathan E. Hunter