Biologically inspired feature based categorization of objects.

T. Nathan Mundhenk[a,b1] , Vidhya Navalpakkam[a], Hendrik Makaliwe[a], Shrihari Vasudevan[a], Laurent Itti[a]

[a]Computer Science Department, Henry Salvatori Computer Science Center, University of Southern California, Los Angeles, CA 90089-0781;
[b]Aerospace Integration Science Center, Aerospace Corporation, 2350 E. El Segundo Blvd, El Segundo, CA 90245-4691

ABSTRACT

We have developed a method for clustering features into objects by taking those features which include intensity, orientations and colors from the most salient points in an image as determined by our biologically motivated saliency program. We can train a program to cluster these features by only supplying as training input the number of objects that should appear in an image. We do this by clustering from a technique that involves linking nodes in a minimum spanning tree by not only distance, but by a density metric as well. We can then form classes over objects or object segmentation in a novel validation set by training over a set of seven soft and hard parameters. We discus as well the uses of such a flexible method in landmark based navigation since a robot using such a method may have a better ability to generalize over the features and objects.

1. INTRODUCTION

1.1 Project overview

This paper discuses a feature based clustering mechanism which is designed to aid in visual scene understanding. As such we wish go gain a rudimentary ability to classify objects based upon simple feature vectors extracted from an image using visual saliency. Additionally, we wish to embed our program with the robustness afforded biological entities. Thus, we use as our starting point a biologically inspired visual saliency program which not only can attend to salient locations but can obtain image features at those locations. This can then can be used to classify objects by finding the way sets of features interact with each other.

1.2 Saliency and object features

One key problem in image processing is the size of the problem space. For instance, to analyze every single pixel in an image that is 640x480 pixels in size takes 307,200 operations. If you wanted to cross compare every pixel, that balloons to a whopping 94,371,840,000 operations. Even on a computer that can compute a trillion operations per second, you could only hope to achieve 10 image processing's per second (as in frames per second). Thus, for comparison operations, one would want to reduce the problem space such that only the most interesting pixels or image features are used. To do this, we use a method of bottom up saliency to extract the most interesting features in real time.

The bottom-up saliency model as proposed by Itti and Koch (2001), starts by taking in different types of features such as color, intensity and orientation and placing them into feature maps which are computed at different spatial scales(i.e., at different levels of resolution). Since the model is biologically based, features are computed in a center-surround fashion similar to the receptive fields of the simple cells in V1 of the human visual cortex. Thus, we simulate neurons which fires strongly if it is receiving features locally different from its neighbors. This is followed

[1] mundhenk@usc.edu; University of Southern California, 3641 Watt Way, HNB 06, Los Angeles Ca, 90089-2520

by a spatial competition which is comprised of non-linear interactions between all neurons within a feature map, and serves to promote those neurons which are globally different from their neighbors and suppress others that are similar to their neighbors. After applying the spatial competition, the feature maps at different scales are combined linearly to form a saliency map. A spatial competition in the saliency map yields the winner or the most salient location in the scene.

Some of the interesting aspects of the biologically inspired saliency model are as follows: if in an image, there are 10 red balls amidst 10 black balls, the red balls are not attractive or salient; but if there is only one red ball among 10 black balls, the red ball becomes salient and emerges as the winner of the spatial competition, otherwise known as the winner-take-all. In this sense, saliency helps to draw attention to scene locations that are unique and different from their surroundings. This motivates us to use saliency to find interesting locations within an object that may be unique to that object, and hence may help in distinguishing it from another object.

Note that the saliency model simply attends to a location based on its bottom-up salience (image-based properties) and does not have any knowledge about the identity of the object that it's currently attending to. One goal is to enhance the saliency model by allowing it to attend freely to interesting locations, extracting features from each of the attended locations, and apply a clustering technique to group the extracted features into object categories, so as to help the pure bottom-up saliency model acquire information about the objects in its environment which may further aid in understanding the scene. Thus, understanding can aid saliency which in turn can aid understanding alternately.

1.3 Motivation for feature based object recognition in landmark based robot navigation

A motivation for developing object classification and recognition algorithms is to support robot navigation. Here we have describe the Beobot project (Mundhenk, *et al*, 2003), which uses a high performance mobile robot for explorations in robot navigation, with a special focus on the development of biologically-inspired vision algorithms for navigation.

To achieve a goal-based navigation task, a robot should be able to perform localization and path planning. Localization is concerned with the identification of the current location of the robot with respect to its environment, while path planning is concerned with finding the appropriate actions (decisions on directions and distances of movements) for the robot to move among objects in its environment to reach a certain goal location.

The most common approach to achieve localization and path planning is by selecting and recognizing visual landmarks. Some recent examples are (Thrun, 1998; Nehmzow & Owen, 2000; Martinez & Torras, 2001; Busquets *et al*, 2003). In general the task of landmark-based navigation consists of a few main sub-tasks. The first is landmark selection. This involves finding objects that can be used as landmarks in the particular environment. Additionally map building involves incorporating the landmarks into a memory representation of the environment in a main sub-task. Finally was have landmark recognition in navigation which we define as the recognition of landmarks in the environment which facilitates using them for localization and path planning. Thus, localization is done by recognizing landmarks in the images acquired by the vision system. As such, the subtasks of landmark selection and recognition rely on the ability to freely classify objects in the environment. Hence, object classification and recognition is a major part of robot navigation.

Currently, the performance of most effective landmark-based navigation systems rely on heavy restrictions on the environment. They either use artificial landmarks (e.g. Scharstein & Briggs, 2001), or preselected landmarks in the environment such as doors or vertical lines. These approaches might work well within the particular domain, and might be very effective for certain tasks. However they tend to break down when the environment changes from its specified domain.

Besides being domain-dependent, approaches based on artificial or preselected landmarks face the problem of landmark distribution. Ideally, a landmark-based navigation system should use a collection of landmarks that are spatially distributed in an optimal way, in the sense that there are enough landmarks to navigate from every location, yet not too many as to be redundant and too computationally expensive. However, predetermined landmarks can lead to overcrowding of landmarks (resulting in inefficiency) or the lack of landmarks (resulting in inability to do localization).

In our robot navigation project, we are interested in navigation in unrestricted environments, such as natural outdoor environments or non-predefined indoor environments. To achieve generality across different environments, the robot has to be able to autonomously select and use useful parts of the scene that could be used as landmarks in each neighborhood. This requires:

(1) The ability to find parts of the scene that are useful as landmarks.
(2) The ability to select them in such a way as to make the spatial distribution optimal.
(3) The ability to recognize them again for the purposes of localization and path planning.

What we need is a general, domain-independent way for finding and using (i.e. recognizing) a good landmark in a particular neighborhood. However, the issue of finding useful landmarks boils down to the issue of the definition of landmarks. There is no robust definition of a landmark. A box-like object is probably a good landmark in the countryside, but the property of being box-like will not be enough for fitness in an urban environment, where objects with that general feature are common. Thus, different features might be useful as landmarks in different locations and situations. Additionally, a landmark does not necessarily correspond to an individual object. It could be part of an object (e.g. one bright side of a building), or a cluster of objects (e.g. a gas station). Furthermore, a useful landmark is not restricted by size.

1.2.1 Our approach: feature-based, landmark-based navigation

Although developing a robust way to find and recognize landmarks seems intractable, there are several properties that could be used effectively and efficiently in approaching this goal. One property of landmarks is that they are salient parts of a scene. To serve as a landmark, it has to be prominent in the image, such that it will be noticed quickly when the system is trying to find new landmarks, and will be quickly found again in future explorations in the same neighborhood. Hence, we take it as one of the necessary conditions for being a landmark is that it is salient in any part of the scene. We believe this condition is domain-independent & task-independent, and can be applied to unrestricted environments.

Our approach for landmark selection and use is based on saliency-based vision saliency algorithms developed by Itti and Koch; and subsequently extended in our lab. The details, which we explained in section 1.1, are used in each of the sub-tasks of landmark-based navigation. For instance, in landmark selection, we use the saliency algorithm to find salient locations of the scene, which will then be used to narrow the visual space for landmark identification. Additionally we may employ knowledge of features to create an expectation of object features in navigation such that navigation towards a goal is done in an expectation-based manner. After passing one landmark, it will expect the next landmark that leads towards the goal location, according to the spatial relations of landmarks given by the map. We then state that expectation-based recognition of landmarks is done in a top-down manner in a way similar to the saliency-based object-recognition algorithm developed by (Navalpakkam V, Itti L, 2003).

This saliency-based approach to landmark-based navigation has several advantages. First it mainly utilizes simple procedures such as the saliency and clustering algorithms, instead of heavy-duty object recognition algorithms. Additionally, the landmarks found are any salient parts of the image, which are more general than objects. They can be parts of an object, or clusters of objects, or simply any salient parts of the scene. The landmarks are not restricted by any specific notion of an object. Finally, by looking for the most salient parts in every scene, it will avoid both the problems of landmark overcrowding and lack of landmark. This reduces the problem of landmark distribution significantly.

The basic saliency and clustering algorithms may not solve all problems. However we believe the advantages mentioned make them worth it to explore them further. We are interested in exploring how far it will go in producing a feasible robot navigation system. A similar saliency-based approach for landmark finding was also followed by (Todt & Torras, 2000). However, our approach differ in several respects, mainly in the clustering algorithm and the approach to object recognition, as explained in 2.2.

1.2.3 Additional Motivations

In addition to landmark based navigation, we also wish to understand categories of objects to aid in a distributed surveillance system (Mundhenk *et al*, 2003b). In this system, multiple cameras will monitor a scene for interesting events. The camera agents must have a way of both distributing attention and flagging a scene as interesting to a human operator. As such having a rudimentary knowledge of objects aids in scene understanding. We use the feature based categorization system to both help us learn how certain objects behave in a scene, for instance are we observing the movement of a person or a photocopier. Additionally, since we wish to distribute attention, having an idea of the object being attended by one camera can be used to bias another camera to not look at it, or alternatively, to demand more attention from other cameras if it is deemed that a scene should require more attention.

1.3 Feature distance and clustering

It is well known that features can be used to classify objects with some success depending on the features used and method to bind them. For instance, the classic Kohonen map method takes in features and clusters objects based upon their perceived proximity in feature space (Kohonen, 1989). Thus, we would like to gain some understanding of objects using some method to cluster them together. Since we are interested in building a real time system, clusters must be built quickly. However, the method used must be not only be flexible enough to generalize over the problem space, but easy to train so that its usability is open to non-experts.

We have developed a feature clustering algorithm which attempts to take feature locality into mind as well as a feature locus in a similar way Kohonen maps do, but with less time complexity. Our approach thus involves clustering by feature similarity as a distance metric, but adds an attractor based upon the density of some feature. The goal is to be able to train the algorithm to cluster once, then use a faster mechanism to cluster after training has occurred. Additionally, the method should use soft parameters such that things such as class number, size or absolute spacing are not necessary for operation and thus require that we understand *how* to cluster, not *what* to cluster.

2. CATAGORIZATION METHOD

2.1 Feature Extraction

Features for the model are taken from the most salient points in natural images determined by the biological saliency model described earlier. The saliency model works by simulating V1cortical interactions and taking in an image and then finding image properties such as angles, intensities and color opponencies. Additionally, each feature is analyzed at six different scales using an image pyramid. Saliency is determined by competing features against each other such that a winner-take-all allows the most unique feature within a map to stand out. By combining the winning feature maps we are given the most salient location. Once we know the most salient location in the image, we can extract the features of that area (figure 1). This creates a feature vector with 42 items. These represent several “channels” defined as four angle features at 0,45,90 and 135 degrees, one feature for intensity and two for color oponency, which as in the human brain are blue/yellow and red/green. In all this makes seven features which we take at six different scales. For each image we analyze, we extract 300 feature vectors taken at different locations as determined by the saliency of the locations in the image.

Features are then reduced in dimension using ICA (Independent Component Analysis). It should be noted that while ICA is not designed explicitly for dimensionality reduction the same way as PCA (Principle Component Analysis), since we can construct a non-square unmixing matrix it can be used for this purpose. To perform ICA we used FastICA from the ICA group at the Helsinki University of Technology (Hyvärinen, 1999). This was chosen due to its ease of use and speed as well as its ability to easily do dimensionality reduction. ICA was itself chosen over PCA because we believe that the multi-scale analysis creates mixing dependencies in the data. That is, the result from an image analysis at a location should correlate and thus mix with the same analysis at that location at a different scale. Since ICA maximized the joint entropy, we are more likely to find interesting mutual information between the scales

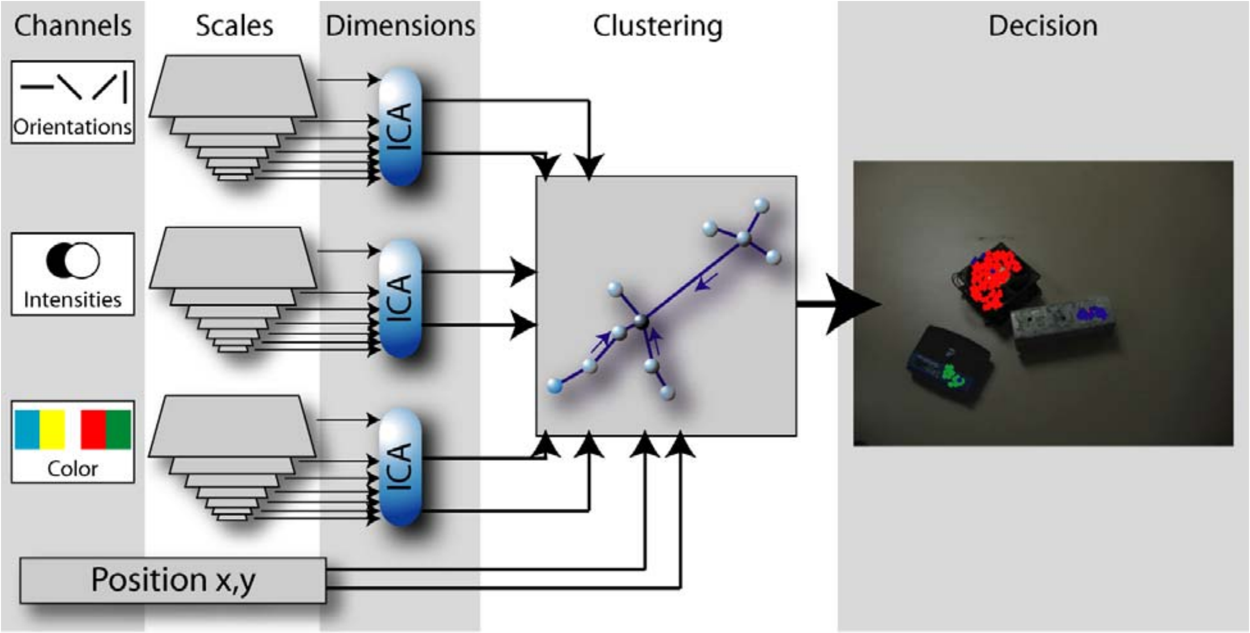

Figure 1: Features are extracted from the most salient locations in the image. The locations exists at multiple scales which undergo dimension reduction using ICA. Finally features are clustered into contiguous object parts.

than if we used PCA. The ICA unmixing matrix was determined by performing ICA over the entire data set of all images on a per channel basis.

2.2 Feature Clustering

To cluster features we developed our own clustering algorithm. We did this because most clustering algorithms make strong assumptions about hard parameters that we believe makes them less flexable. For instance the popular EM (Expectation Maximization) (Dempster, 1977) algorithm requires an assumption about the number of classes which are to be clustered. The method we use is a hybrid of minimum spanning tree clustering with a density function and global statistics which help us cluster based upon soft parameters (figure 2).

The clustering method works in several steps. The first step is to find the density and distance between each point in the feature space. For each point with feature vector **x** density *d* is defined as:

$$d_i = \frac{1}{\sum_j \left\| \mathbf{x}_i - \mathbf{x}_j \right\|} \tag{1.1}$$

This is the inverse sum of all distance between this point and each other point. Points which lie in very crowded regions or at the center of clusters will have a very high value for *d* while points in sparse regions or on the edge of clusters will have a very low value for *d*. Additionally, since this value is taken over all values, it is an absolute measure and not subject quite so much to local effects. The distance itself is calculated at the same time using Euclidian distance defined as

$$l_i = \left\| \mathbf{x}_i - \mathbf{x}_j \right\| \tag{1.2}$$

The next step is to connect up points into a hierarchy using the following rule: *Connect each point to another point as a parent iff the parent will have a higher density d and there is no other point closer which has a higher density*

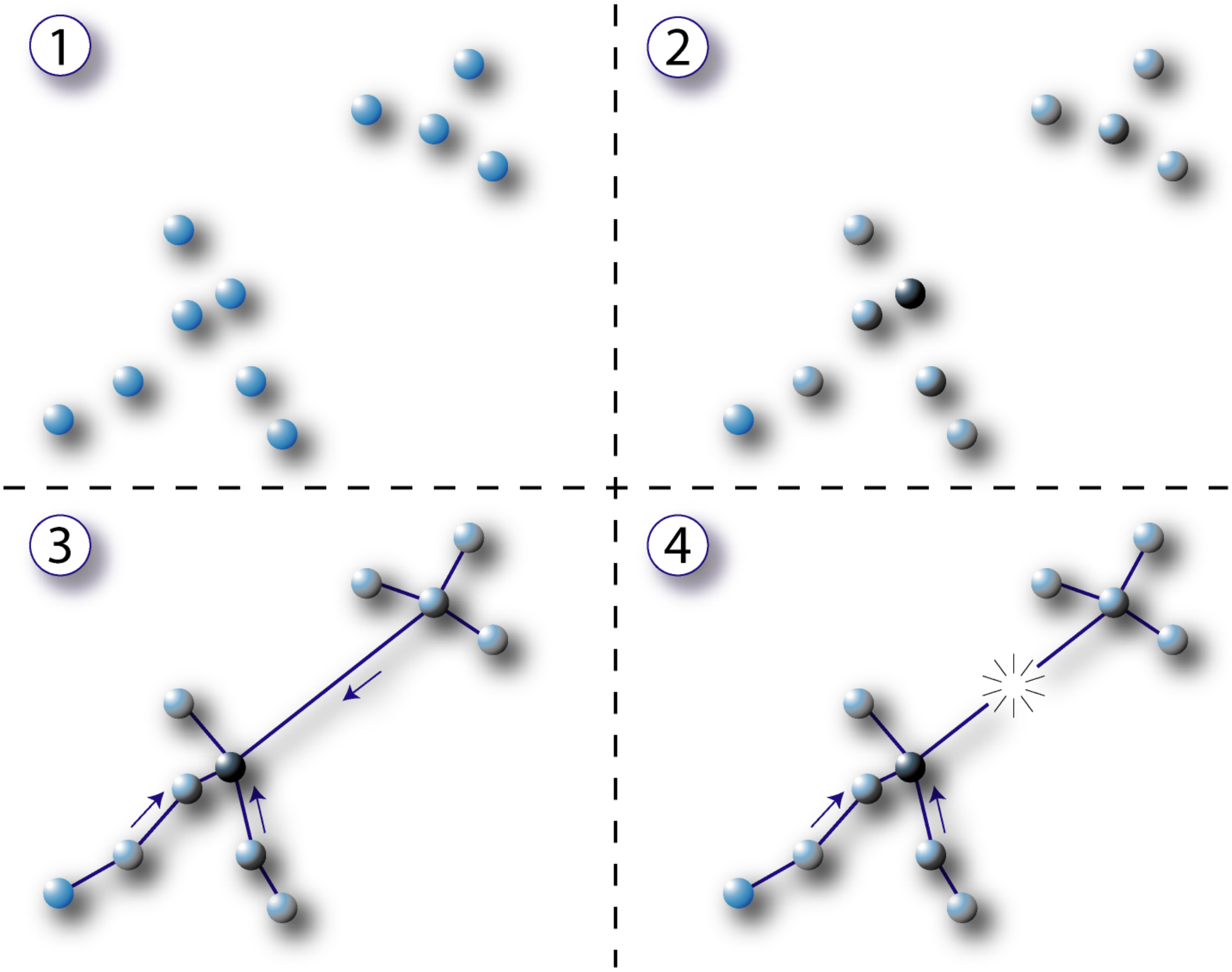

Figure 2: (1) An input set of features is plotted in feature space. (2) a density function is plotted based upon the sum inverse Euclidian distance between points. (3) Points are connected to points which are the closest point with a higher density. (4) Soft parameters for the relative size of edges determine which edges should separate classes.

than this point. What this means is that you connect a point to another point which is the closest point with a higher density than this point. The density metric is not used as a distance substitution but is intended to ensure that points are linked in the direction of higher density. This creates a kind of a gradient descent situation whereby links in clusters move towards higher density, but are constrained by distance into local clusters.

The next step is to decide how to break the tree into smaller but meaningful clusters. We achieve this with soft as well as some hard parameters. The parameters generally fall into three groups. The first group is the standard deviation of edge lengths. The second parameter is the size of a cluster that will be formed in the number of members by standard deviation of descendant number. This means you cannot be a cluster unless you have so many members compared with the normal number of descendants from any given node. The third set of parameters are hard parameters for length size and cluster members.

For the standard deviation metric, we sum the distances l from all children nodes to each parent node. We then compute the mean Euclidian distance E(l) and the standard deviation E[(l-E(l))^2]. The length to cut is defined as:

$$l - E(l) > C \bullet E\left[(l - E(l))^2\right] \tag{1.3}$$

That is, if an edge length between a child and a parent is longer than constant C times the standard deviation, cut that length and create two new separate classes.

Additionally, clusters are expected to have a somewhat normal number of members. This is done by making sure that any new parent has a number of descendants that is a good number. An example is that if most nodes have 20 or so descendants, we would not make a cluster out of 3 nodes since this is very small compared with how many points lie in this space as well as how many nodes tend to lie below one another. In general, this keeps nodes from being too relatively small.

The third constraint is that edge lengths longer than a certain size must always be cut, or clusters must always have at least a minimum number of members. These numbers tend to be set somewhat high, but are used as a heuristic of sorts in cases where it seems obvious that the rule should always be applied.

Additionally, we can iterate through and create locally weighted effects by doing a second pruning, but only considering statistics within the new classes from the first pass. We found that a second pass seemed very helpful but that a third pass was extraneous.

Thus to summarize, we form clusters based upon both density and distance between points using a tree. Clusters are formed by pruning the tree into sections based upon the relative length of a span as well as how many members will be found in each new class formed.

2.3 Training

To hone the clustering method we use basic gradient decent with sequential quadratic programming using the method described by Powell (1978). For this study, error was defined as the number of classes found versus how many it was expected to find. Thus, we presented the clustering algorithm with 80 training images. Each image had a certain number of objects in it. For instance an image with a ball and a wheel in it would be said to have two objects. The clustering algorithm would state how many classes it thought it found. If it found three classes in an image with two objects then the error was one. The error was computed as average error from the training images.

The training program was allowed to adjust any of several hard or soft parameters. The parameters were the ones discussed above, but during second pass classification, the clustering method was allowed to use a different set of parameters than during the first. In all, this added up to seven parameters to train over.

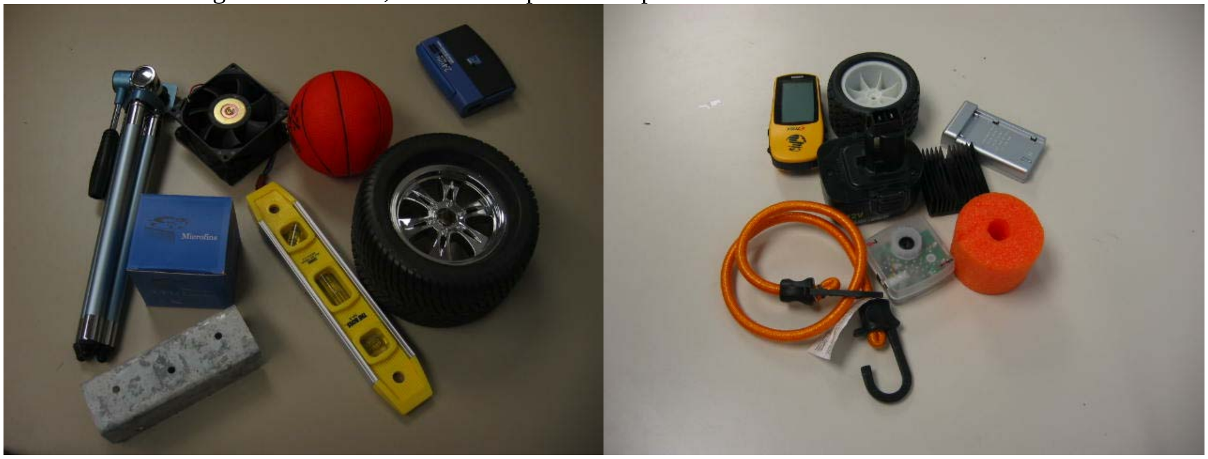

Figure 3: These are the sets of objects used. On the left is a training image which contains all eight objects used in the training images. The Image on the right is one of the validation images and shows the eight objects used in the validation images. Note that the image on the right is darker, the is intentional since we want some robustness to lighting conditions.

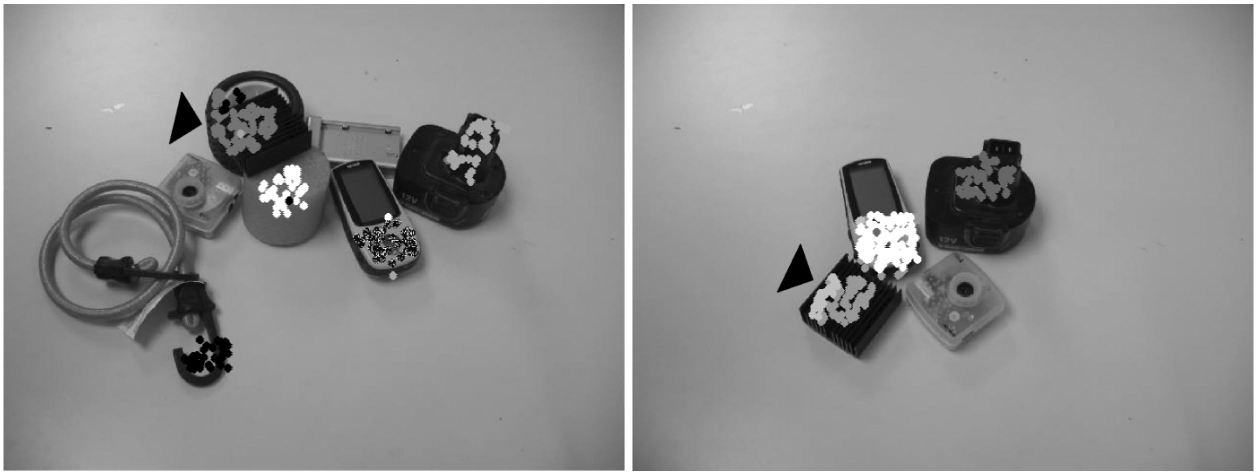

Figure 4: These are examples of errors from our program. The dots show where a feature was extracted by our saliency program. The shades of the dots indicate that a feature was categorized as the same type. The arrow on the left image shows an example of a merge. In this case a heat sink has been merged with a tire as a single object. In the right image the arrow shows a split where the heat sink has been split into two objects.

The training data was comprised of eight base objects of varying complexity such as balls and a wheel on the simple side or a mini tripod and web cam on the more complex side. Figure 3 shows an assortment of the training objects in one of the training images. Objects were placed on a plain white table in different configurations. Images contained different numbers of objects as well. For instance some images contained only one object at a time, while other contained all eight. A separate set of validation images was also created. These consisted of a set of eight different objects with a different lighting created by altering the f-stop on the camera. Thus, the training images were taken with an f-stop of 60 while the 83 validation images were taken with an f-stop of 30. Additionally, the angle and distance of view point is not the same between the training and validation sets. The validation images were not used until after optimal parameters were obtained by the training images. Then the exact same parameters were used for the validation phase.

3. RESULTS

3.1 Segmentation

Our first test was to examine if we could at the very least segment images such that the program could tell which objects were different from each other. For this test spatial interaction was taken into account. We did this by adding in spatial coordinates as two more feature vectors in with the original 14. The sum total of spatial features were weighted about the same as the sum total of non-spatial features. As such, the membership of an object in one segmented class or the other was based half by its location in space and half by its base feature vector composition. Reliability was measured by counting the number of times objects were classified as single objects, the number of times separate objects were merged as one object and the number of time a single object was split into two unique objects (figure 4). Additionally, there was a fourth category for when objects were split into more than three objects. This was small and contained only four instances.

The results were generally promising in that based upon simple feature vectors alone[2], the program was able to segment objects correctly with no splits or merges in 125 out of the 223 objects it attempted to segment. In 40 instances an object was split into two objects. Additionally 54 objects were merged as one object. While on the surface these number might seem discouraging there are several important factors to take into account. The first is that the program was segmenting based solely on simple features vectors with a spatial cue. As such it could frequently merge one shiny black object into another shiny black object. In 62 % of the cases of a merger, it was

[2] All results can be seen at http://www.nerd-cam.com/cluster-results

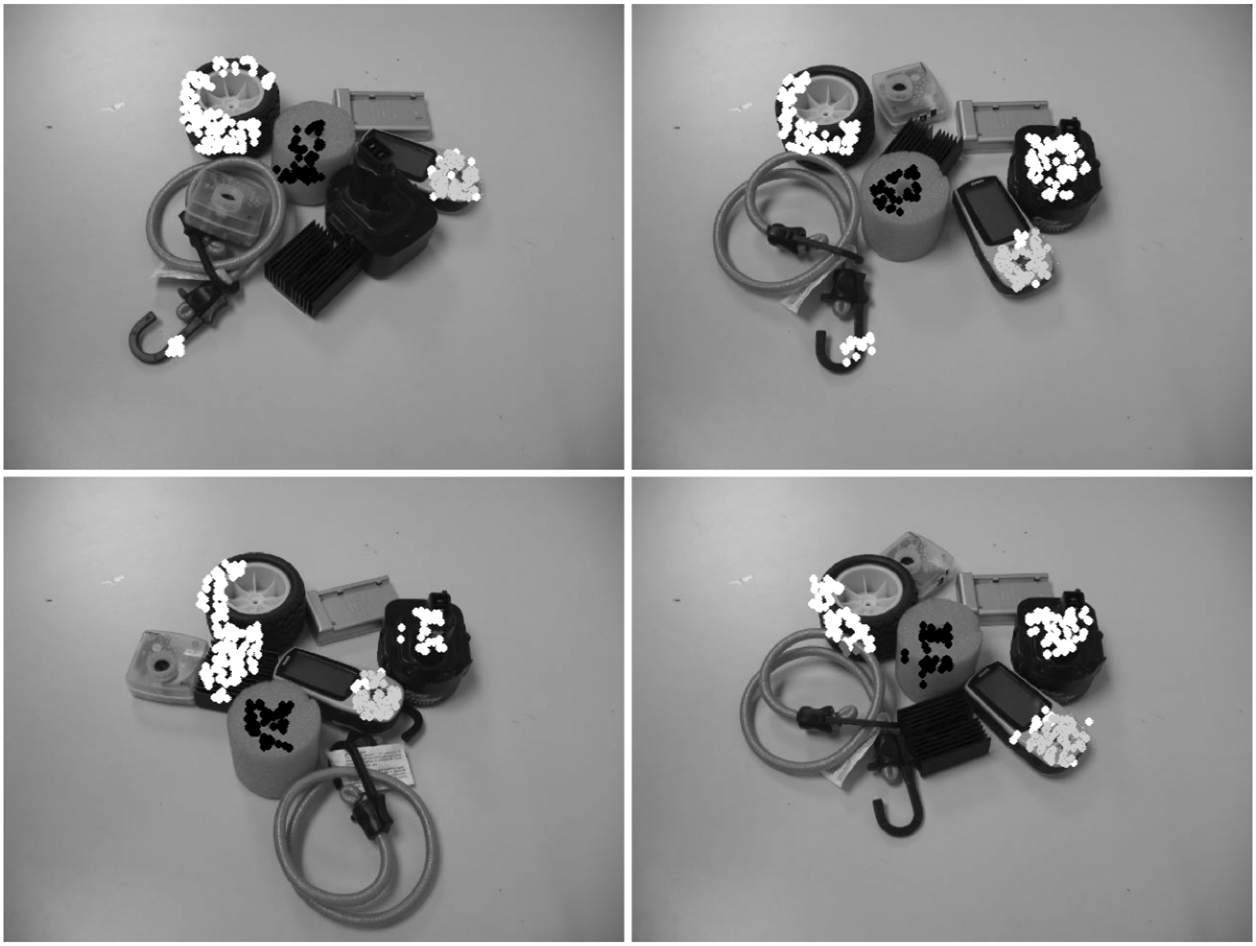

Figure 5: These are four examples of how objects were classified. The orange cylinder was always classified uniquely as was the GPS face plate. The tire, cordless drill battery and wheel were consistently lumped in as the same object.

obvious that the merged objects were very similar with respect to features. Generally splits occurred most often on the heat sink. This may be because the jutting edges create a strong variance between features.

Another result that should be noted is that the algorithm in its current unoptimized state took less than 200 ms to cluster each of the validation images on an Athlon 1400 processor based machine. This means that using the current method, we will be able to cluster at a rate of 10 frames per second on a newer PC running at 3 GHz. Since the bulk of computation involves independent analysis over several feature vectors, we could divide the task over several CPU's. This suggests that the current clustering scheme should be able to be implemented in real time (30 fps) on a multiprocessor-based machine such as the Beobot described in section 1.1.

3.2 Object Identification

The last 10 images with the most objects were taken and their features pooled into a large cluster. This cluster was then used to separate out objects according to features. The same parameters were used to create categories in this super cluster as were used in clustering individual objects. Thus, the super cluster ran with the exact same parameters as the earlier validation and training sets. Additionally, the super cluster was comprised of a set of features from the validation images. The difference between this experiment and the segmentation was that the same object had to be clustered as the same object between images. Thus, a successful result was one where feature vectors indicated that an object was classified for instance as object *A* in all images. Additionally it should be noted that spatial location of feature vectors was not used as a cue in this experiment.

The clustering algorithm reliably taught itself to classify objects into three categories[3]. The first category consisted solely of the orange foam cylinder. The second object class also consisted solely of the GPS faceplate. The last category consisted of glossy rounded black plastic objects. As can be seen in figure 5, these included a small RC car wheel, a cordless drill battery and a plastic hook.

Additionally, it learned to distinguish objects even when they were moved around both spatially and rotationally. This suggests an advantage to a feature based method for identification in that we only care about feature similarity, but do not care so much about the actual spatial arrangement of features. Thus, we might expect more robustness to rotational and location variance.

4. DISCUSSION

In our preliminary studies shown here, we were able to teach our program to segment and classify objects without it having any *a priori* knowledge of what comprises a sound classification and segmentation strategy. This is important for several reasons. The first is that what comprises an object or makes it unique is often subjective. Thus, if I hold up my hand and ask you how many objects there are, you might answer one, for *one hand* or your might count the fingers. The idea behind giving the program only the number of objects was so that it could begin to guess at the experimenters subjective judgment as to what comprises an object. Additionally, it is important because we have removed a layer of supervision in the training process. In order for artificial intelligence to be deployed in the real world, it must be comprehensible to non-experts. Thus, training was simple and only required that the experimenter know how many objects were in each image. More rigorous supervision may be too tedious for people not well versed in issues of computer vision and artificial intelligence. Further, by making the learning process more freeform we allow the system more ability to self organize. That is, a combination of soft parameters with free form training may help to avoid strong bias and enable more generalization.

It should also be mentioned what are some other strengths that may be found in freeform learning. The most obvious strength is that what you may think a program should learn may in fact not be what it should learn. That is, it is very difficult to quantify all the lessons that an AI should acquire. If we bias what it learns too greatly then it cannot explore and learn lessons that we had not intended but are still important. Thus, the acquisition of knowledge and judgment should be both intentional and unintentional.

4.1 Limitations

There are several limitations that should be noted to the current research as has been presented. The first thing to note is that we are not claiming that the algorithms presented perform superior on identification and segmentation tasks, instead what we are claiming is that its potential lies in its ability to do so with a minimal amount of supervision. So for instance, with the eight objects presented, a simple histogram based technique would be superior for classification. However, in order to use such a method a person must take the time to pick out the portions of the image which are to be categorized and label them. Additionally, such a method may not generalize as well since we have told to program exactly what to look for.

Additionally, the current method cannot make distinctions for complex objects with very similar feature vector sets. Thus, it will not tell the difference between a pentagon and a hexagon. However, for simple visual tasks, such a simple identification and classification scheme may be all that’s needed. So for instance, for navigation we may only need to be as good at visual tasks as a reptile or lower mammal. It may simply suffice for the time to be able to tell the difference between a telephone pole and a human and still not be able to tell the difference between people. As such, a sound visual approach may be to conquer visual problems on a lower evolutionary rung and move your way up the ladder.

[3] All results can be seen at http://www.nerd-cam.com/cluster-results

We should also note that the results are preliminary and we do not believe them to be statistically rigorous. As such the reader should have noticed that the results may seem somewhat subjective. For instance not all features are completely accounted for in such a way as to show the complete error in judgment made by the program.

We are also limited at the current time based on the complexity of the algorithm used. Since time complexity with *d* as the number of dimensions and *n* as the number of points is $O(dn^2)$ this limits the number of data points we can cluster at any given time. It should be noted however that the time complexity it not terrible considering that we can cluster odd shaped regions. If one were to use EM with covariance to describe an odd shaped region then the complexity of the problem, with *c* classes is between $O(cn^2 \log(n))$ and $O(cn^3)$. This is do to the fact that this involves matrix inversion to derive the covariance matrix which is another limiting factor since we can experience singularity problems in the process.

4.2 Future Work

For future work we will concentrate on three different areas for improvement. The first area is to improve the general performance issues. We would like to be able to do clustering in real time such that it will prove useful in the robotic and surveillance applications we discussed in the intro. Given that a high end PC can cluster at a rate of about 10 fps we believe that real time applications are forth coming with some optimization, distributed computation and maybe a little help from Moore's law.

Additionally, we will improve memory complexity by using the clustering algorithm for discovery but not storage. For this we will use the clustering algorithm in each image to find the unique objects, but we will then store the results of each cluster as a set of probabilities over the features. Thus, classification will be done using a more Bayesian like approach. This should allow us to store information about more objects since we will not be required to store every feature vector, but will instead just store general information about each cluster.

A third area of improvement will involve meta clustering. This will work by extending the Bayesian framework into a second stage cluster. This means that we will store the cluster information in a Bayesian framework, but that the Bayesian features will be merged into classes by clustering over them. As an example if we have several descriptions of a cup we would want to generalize over them such that we have a single general idea of what a cup is like. Ideally, the Bayesian description of cups should be similar enough to group together using the same clustering method to derive a general idea of what a cup really is. Thus, we would create identity by iterating over associations using clustering, and Bayesian statistics to reduce unneeded information to an ideal generalization similar to other information reduction clustering methods (Jain, Murty, Flynn, 1999).

5. CONCLUSION

We have developed and done a first round of testing on a program for taking features from the environment and clustering them into classes of objects. This will help us to develop a landmark based robot navigation scheme using a biologically motivated visual system. Additionally, this should help us in scene analysis and divided attention tasks since we have a rudimentary idea about the identity of different objects in a scene.

ACKNOWLEDGEMENTS

We would like to thank Mike Olson for his help and suggestions. This research is supported by the National Imagery and Mapping Agency, the National Science Foundation, the National Eye Institute, the Zumberge Faculty Innovation Research Fund, the Charles Lee Powell Foundation and Aerospace Corporation.

REFERENCES

1. Busquets B, Sierra C, López de Mántaras R, 2003, "A multi-agent approach to qualitative landmark-based navigation". *Autonomous Robots Journal***,** 15(2003):129-154.

2. Dempster, A P, Laird N M, Rubin D B, 1977, Maximum likelihood from incomplete data via the EM algorithm, *J royal Stat. Soc.* B **39**(1):1-38.
3. Hyvärinen A, 1999, Fast and Robust Fixed-Point Algorithms for Independent Component Analysis, *IEEE Transactions on Neural Networks* **10**(3):626-634.
4. Itti l, Koch C, 2001, Computational Modeling of Visual Attention, *Nature Reviews Neuroscience*, **2**(3):194-203.
5. Jain A K, Murty M N, Flynn P J, 1999, Data Clustering: A Review, *ACM Computing Surveys*, **31**(3):265-323.
6. Kohonen T, 1989, *Self-Organizing and Associative memory*, 3rd ed. Springer information Series. Springer-Verlag, New York, NY.
7. Martinez E, Torras C, 2001, Qualitative vision for the guidance of legged robots in unstructured environments, *Pattern Recognition*, **34**(8):1585 – 1599.
8. Mundhenk T N, Ackerman C, Chung D, Dhavale N, Hudson B, Hirata R, Pichon E, Shi Z, Tsui A, Itti L, 2003a, Low-cost high-performance mobile robot design utilizing off-the-shelf parts and the Beowulf concept: the Beobot project, *Proc. SPIE Conference on Intelligent Robots and Computer Vision XXI: Algorithms, Techniques, and Active Vision,* Providence RI, October.
9. Mundhenk T N, Dhavale N, Marmol S, Calleja E, Navalpakkam V, Bellman K, Landauer C, Arbib M A, Itti L, 2003b, Utilization and viability of biologically-inspired algorithms in a dynamic multi-agent camera surveillance system, *Proc. SPIE Conference on Intelligent Robots and Computer Vision XXI: Algorithms, Techniques, and Active Vision,* Providence RI. October.
10. Navalpakkam V, Itti L, 2003, Sharing Resources: Buy Attention, Get Recognition. *Proc. International Workshop on Attention and Performance in Computer Vision (WAPCV'03), Graz, Austria*, Jul.
11. Nehmzow U, Owen C., 2000, Robot navigation in the real world: Experiments with Manchester's FortyTwo in unmodified, large environments. *J. Robotics and Autonomous Systems*, **33**(4):223-242.
12. Powell, M J D, 1978, A Fast Algorithm for Nonlinearly Constrained Optimization Calculations, *Numerical Analysis*, ed. G.A. Watson, *Lecture Notes in Mathematics*, Springer-Verlag, Vol. 630.
13. Scharstein D, Briggs A, 2001, Real-time recognition of self-similar landmarks, *Image and Vision Computing*, **19**(11): 763-772.
14. Thrun S, 1998, Bayesian landmark learning for mobile robot navigation, *Machine Learning*, **33**(1):41-76.
15. Todt E, Torras C, 2000, Detection of natural landmarks through multiscale opponent features. *Proc. 15th Intl. Conf. on Pattern Recognition (ICPR-2000),* Barcelona, Sept.