CS280 Project, Spring 1996: Grouping of Color and Texture Features for Automated Image Annotation

Serge Belongie, Robert Blasi and Kevin Murphy

May 28, 1996

Abstract

We present a system for automated image annotation which is capable of detecting concepts such as sky, water and man-made structure in color images. The processing of each image consists of a feature extraction stage and a grouping stage. The ensuing concept-recognition is accomplished using a decision tree. The pixel-level features consist of basic color and texture information. The color information consists of hue, saturation and value (intensity) data and the texture information is obtained using the windowed-image second moment matrix.

The pixel-level features are quantized into one of a dozen or so bins based on an empirically determined perceptual partitioning of color/texture space. The set of binary images associated with this quantization step are each grouped in parallel according to three different strategies. The three grouping strategies seek to form regions according to (1) solid contiguity, (2) similarity in local orientation and (3) similarity in diffuseness. As an example, one of the above mentioned binary images contains a 1 at each point where the original image contains a pixel with a bluish hue. Should the input image contain clear blue sky above the horizon, the first grouping strategy would produce a large connected region in the binary image representing the pixels with a light-blue color. The second grouping strategy would abort due to lack of orientation strength and the third strategy would fail since the sky-blob is not diffuse.

Each blob is represented by a feature vector containing its area, coordinates in the image, eccentricity, principle orientation, mean saturation and mean intensity, as well as by the color/texture bin and grouping strategy which gave rise to it. These feature vectors are the input to the decision tree classifier. The decision tree attempts to assign a label to each blob according to these characteristics.

1 Introduction

Our goal in this project is to produce a system for automatic image annotation. Such a system has applications in many areas of computer vision, particularly in the domain of digital libraries. Automated image annotation has the potential to expand the capabilities of search engines beyond simple text queries into the realm of image-content based queries. In particular, automated image annotation could become a standard feature of the so called "web crawlers"– autonomous World Wide Web information gatherers– which currently only cull textual data from the hundreds of thousands of HTML pages in existence.

Not surprisingly, the task of automated image annotation is as difficult and unbounded as it is useful. Without *a priori* knowledge of an image categorization (e.g. faces, wrenches, textures), there is no single technique which will satisfactorily address the challenges presented by each new image which enters the system. This inherent difficulty suggests the use of image processing techniques which proceed from the very general to the very specific, adapting according to the outcome of each successive processing step.

A framework for implementing generic-to-specific processing in this manner is presented in [1], wherein the successive processing steps are integrated with a hierarchy of *grouping operators*. This grouping-based framework has been used, for example, to automate the detection of naked people in color photographs [2]. In this system, local color, texture and symmetry features are grouped to find candidate limbs, which are in turn grouped to detect human figures.

We have chosen in the project presented here to employ a grouping-based framework for the detection of a number of basic concepts such as sky, water, grass and manmade structure. Our approach begins with an early-visual processing step to obtain color and texture information and then proceeds with a number of parallel grouping strategies. The grouping strategies seek to combine the results of the first processing step in a manner which lends itself to the task of classification. For example, a region of an image which is (1) coherent with respect to its light blue color and lack of texture, (2) is located in the upper half of the image and (3) is elongated horizontally suggests the presence of sky. The classification of concepts based on grouped features is accomplished by means of a decision tree.

The structure of this paper is as follows. We first outline the early-visual processing stage which produces the color and texture features. Next we discuss the grouping strategies which operate on the local color and texture features. We then discuss the structure of the decision tree. We conclude with an overview of our results, relevant findings and plans for further investigation. An interactive demonstration of our system may be found at `http://www.cs.berkeley.edu/~murphyk/dwr/submitImage.html`.

2 The Early-Visual Processing Stage

The goal of the early-visual processing stage is to assign each pixel of the image to one of a dozen or so color/texture bins. We may think of this step as a quantization of the space of color and texture features. The color/texture space we use is 6 dimensional, with 3 dimensions for color and 3 dimensions for texture. The motivation for our choices of color and texture descriptors is provided next.

2.1 The HSV Color Space

Each image in the DWR database is stored as a 128 by 192 JPEG. Nominally the color at each pixel is represented by its three RGB values. We opted to convert the the RGB values to their HSV counterparts since we found HSV to be a more perceptually intuitive color space. The three components of an HSV pixel help us to discriminate between different colors as follows:

Hue: useful for distinguishing among pure colors, e.g. green grass vs. red rose

Saturation: allows one to specify shades of gray without regard for hue, e.g. for finding overcast sky or concrete

Value: this is simply the brightness value

A thorough discussion of color spaces may be found in [5].

2.2 The Windowed Image Second Moment Matrix

We have chosen to use the *windowed image second moment matrix* [6] to provide the pixel-level texture descriptors. The definition of the windowed image second moment matrix for an image $I(x, y)$ is

$$M(x, y) = \nabla I(x, y) \nabla I(x, y)^T * G_\sigma(x, y) \tag{1}$$

The chosen window function in our application is an isotropic Gaussian with variance 1, which we approximate using separable convolution with the 1-D binomial kernel $\frac{1}{16}[1 \quad 4 \quad 6 \quad 4 \quad 1]$. The image $I(x, y)$ is taken to be the V component of the HSV decomposition.

The windowed image second moment matrix reveals through its eigenvectors and eigenvalues several pieces of information which are pertinent to the description of local image structure. In particular, the two eigenvalues λ_{max} and λ_{min} of M, which are non-negative since M is postive semidefinite, offer the following interpretations:

$\lambda_{max}, \lambda_{min} \approx 0$	constant intensity; featureless
$\lambda_{max} \gg \lambda_{min}$	single orientation; parallel lines/edges
$\lambda_{max} \approx \lambda_{min}$	multiple orientations; junctions, line-ends

In the second case ($\lambda_{max} \gg \lambda_{min}$) the argument of the eigenvector corresponding to λ_{min}, which we denote by ϕ, may be inspected to obtain an estimate of the local orientation.

2.3 The Color/Texture Bins

Once the six color and texture descriptors for a pixel have been extracted, namely $H, S, V, \lambda_{max}, \lambda_{min}$ and ϕ, that pixel is assigned to one of 13 bins as follows. The first nine bins correspond to an empirically determined partitioning of the hue circle roughly corresponding to red, orange, yellow, green, blue-green, light blue, dark blue, purple and pink. The tenth, eleventh and twelfth bins are reserved for gray, black and white pixels, respectively. The last bin contains pixels which are deemed to be in "rough" neighborhoods, as determined by λ_{min} and λ_{max}. The exact binning criteria are given in Appendix A.

3 The Grouping Stage

After binning each pixel, a set of binary images are created to represent the locations of the pixels in the original image which belong to each of the 13 bins. The binary images created in this step generally contain a number of blobs of varying size and diffuseness. An example set of binary images is shown in Figure 1. In order to take these images from the raw "pixel domain" into a more salient "region domain," we appeal to three different grouping strategies which are described next.

3.1 Grouping Strategies

All three grouping strategies run on subsampled versions of the above binary images. The subsampling factor is taken to be 1/4 in both dimensions. This is done to decrease computation time and cut down on noise effects.

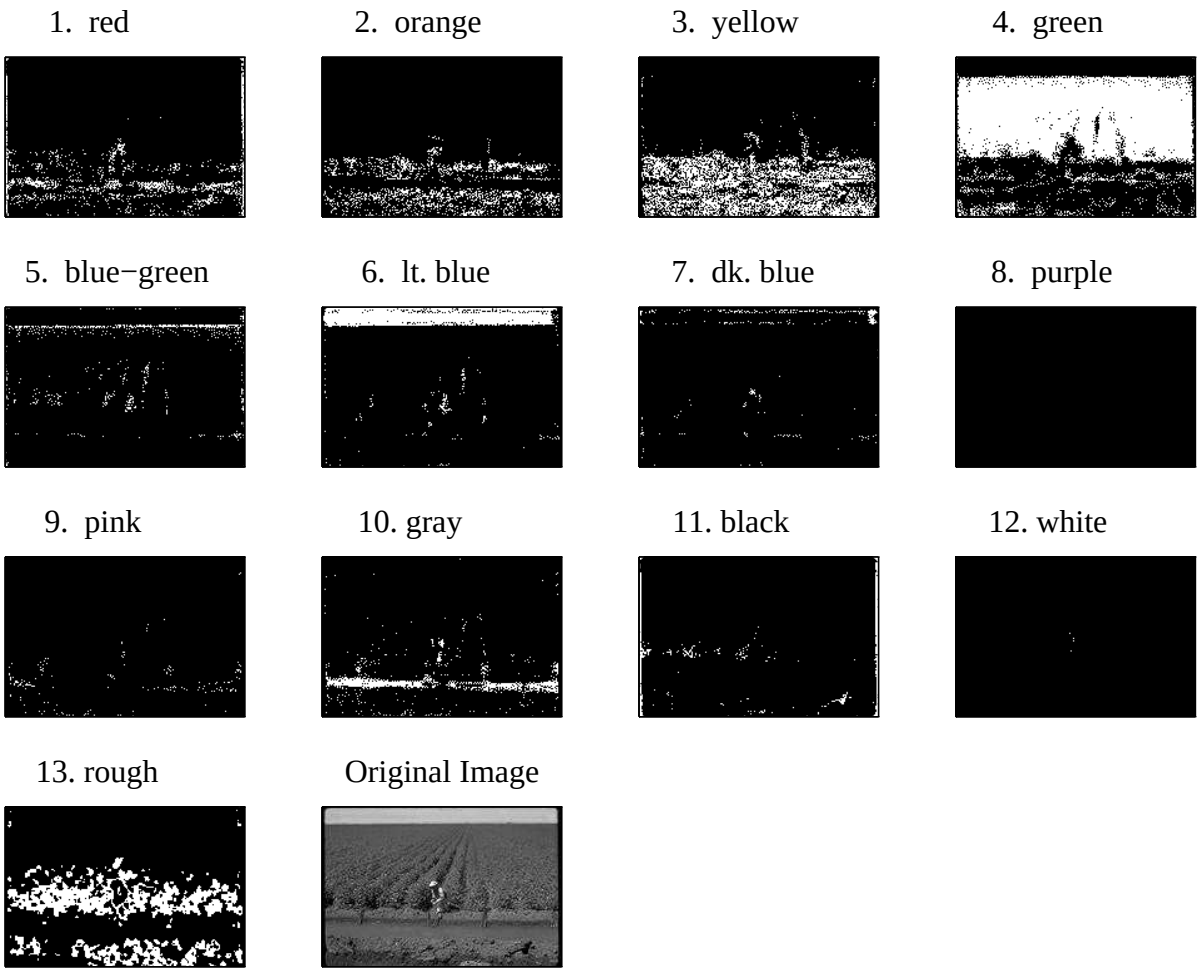

Figure 1: Illustrating the color/texture bin images for a test image.

The first (and simplest) grouping strategy (I) is preferential to solid regions and consists of connected components labeling (CCL) on the subsampled version of the original binary image. We assume 4-connectivity.

Grouping strategy II is preferential to regions of consistent local orientation. This requires inspection of the orientation angle ϕ underlying each pixel in a given binary image. The specific steps are as follows. First, a vector field of double-angle orientation vectors is formed by constructing $\exp(2i\phi)$, as prescribed by Granlund in [4]. This vector field is then smoothed using an 11×11 box filter. The magnitude of the smoothed vector field provides a measure of consistency in local orientation within a given 11×11 region. A maximum magnitude of 121 is attained in regions wherein all of the ϕ's are equal, plus or minus 180°. Pixels whose magnitudes are at least 110 are set to 1 and the rest are set to zero. If the underlying value of λ_{max} is less than .001, then that pixel is set to zero as well. The image containing the surviving pixels from these steps is then subsampled and logicaly ANDed with the binary image from the binning stage. Finally, CCL is performed on this image as in grouping strategy I. As a final note, on grouping strategy II alone the set of binary images from the binning stage is augmented with one binary image which is set to 1 at every pixel. This is to facilitate the detection of orientated regions irrespective of color.

Grouping strategy III is preferential to diffuse regions. The first step in this strategy is to convolve the binary image with an 11×11 box filter. Pixels in the resulting smoothed image whose values lay between 20 and 100 are then marked as "diffuse." As before, CCL is performed as the final step.

Figures 2 illustrates the three grouping strategies on four of the color/texture bin images from the test image shown in Figure 1. Figure 3 shows the results of grouping strategy II on an image containing stripes. In this image, the color/texture bin image was equal to one everywhere.

3.2 The Blob-Level Feature Vectors

The result of the grouping stage was to produce a large number of binary blob-images. The next step is to produce a feature vector for each grouped blob. The following quanitities are included in the feature vector:

- area
- x coordinate of center of mass
- y coordinate of center of mass
- eccentricity
- principle orientation
- mean saturation
- mean intensity
- color/texture bin
- grouping strategy

We next describe our learning/classification techniques which attempt to map the blob-level feature vectors into concepts.

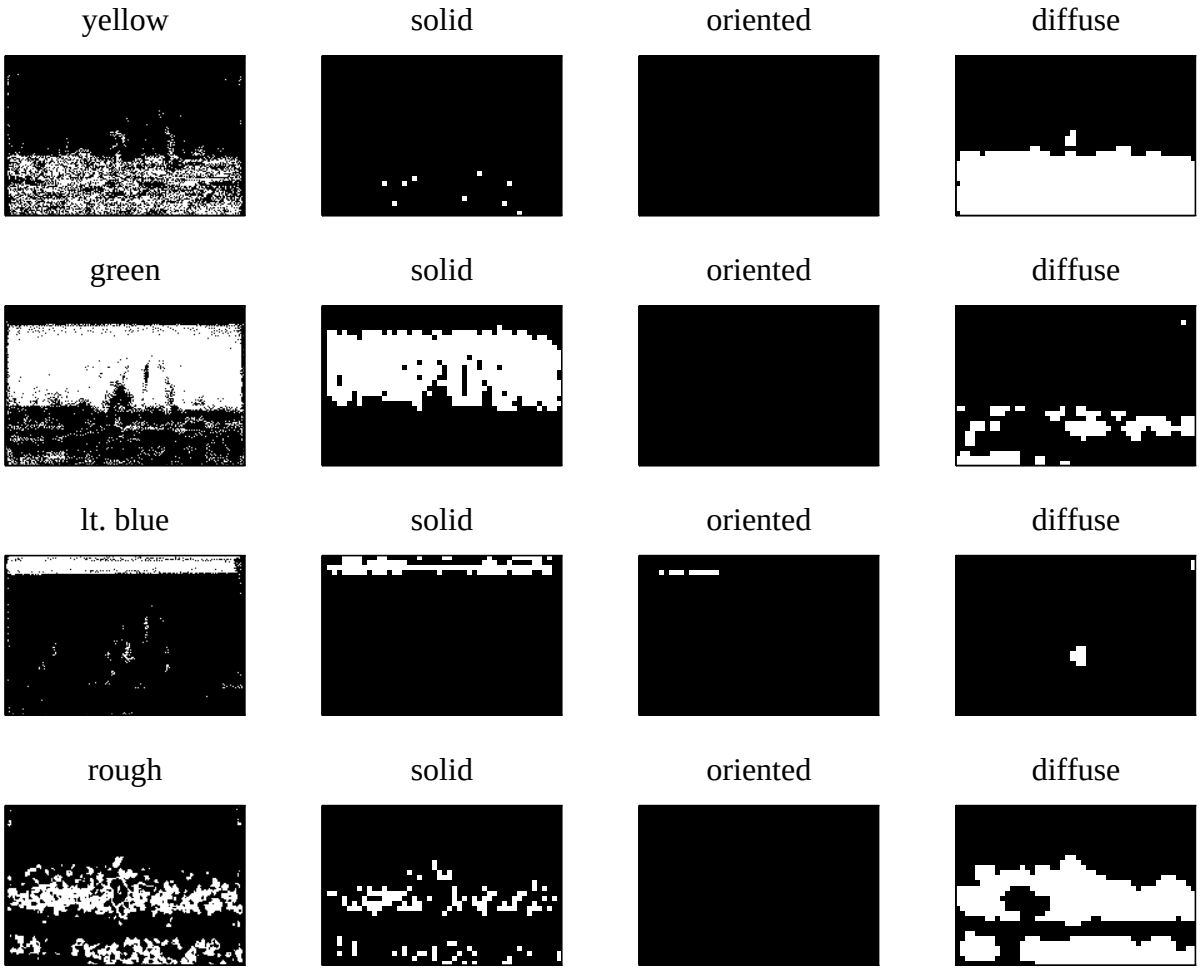

Figure 2: The results of the three grouping strategies for four of the color/texture bin images from the preceding figure.

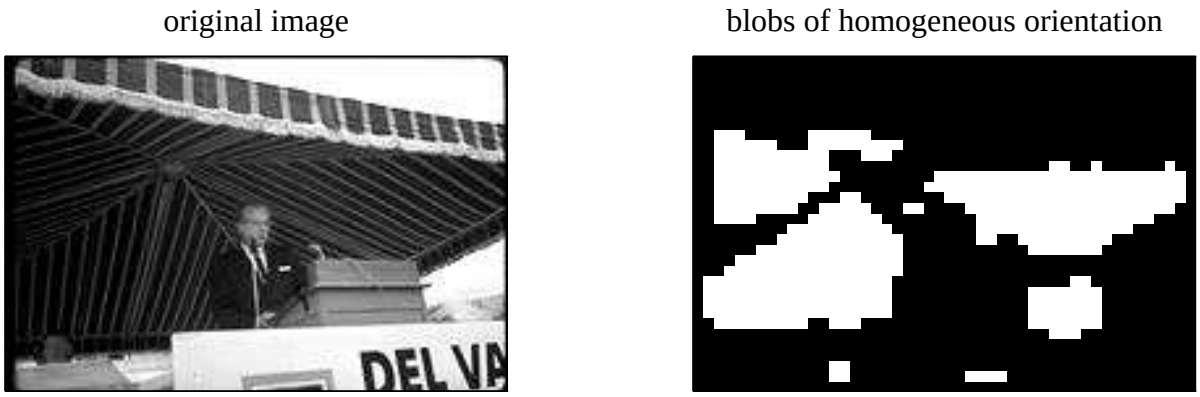

Figure 3: The homogenously oriented regions in the striped canopy in the image on the left show up as distinct blobs in the result of grouping strategy II, shown on the right. (The color/texture bin image was equal to one everywhere, thus defining the entire image as a potential area of interest.)

(animal-fur)	1	cloud	8	dirt	13	(fleshtone)	6
flower	13	(haze)	4	lawn	8	manmade	32
sky	31	snow	8	tan-ground	13	tarmac	12
tree	21	vegetation	37	water	12		

Table 1: The number of examples of each class in the training data. Classes for which we unable to collect more than 8 examples were not used, and are shown in parentheses.

4 The Classification Stage

We used supervised learning to construct a classifier to map the 9 dimensional blob-level feature vector to one of 12 classes ("concepts"). These classes are the labels which we attach to the blobs, and are listed in Table 1. We chose these classes because our intuition told us *a priori* that we should be able to reliably discriminate between them. Some classes could not be used because we did not have time to find enough examples of them. The classifier itself will be described later.

The training data consisted of 208 labelled feature vectors, which were generated as follows. We used the Cypress interface to the DWR database to pick images which contained examples of the classes. We computed (in Matlab) all of the blobs in the image, and for each blob which corresponded to a suitable region in the image, we computed its feature vector and attached a label.

4.1 Decision trees

There are many widely used classifiers. We chose to use a decision tree classifier for the following reasons.

- They are easy to understand once constructed. Each path down the tree can be interpreted as a rule of the form "if $P_1 \wedge P_2 \wedge \cdots \wedge P_n$ then C," where the P_i are predicates (tests on attribute values) and C is a class label. (An attribute is an element of the feature vector.) Geometrically they can be seen as dividing the feature space into box-like regions of varying size.

- We had access to a reliable and well-known commercial implementation of decision trees called C4.5 [11].
- They can be learned quickly and easily. C4.5 took about 10 seconds to learn a tree on our data set. In contrast, learning neural networks is much slower.
- They are a form of sequential decision process. Hence it is not always necessary to compute the whole feature vector. This can be useful if some of the attributes are expensive to compute, e.g., computing axes of symmetry. A more sophisticated method for deciding what features to compute, using utility theory, is discussed in [12].
- They inherently perform discretization of continuous data. This will be discussed below.
- They can return the value "unknown" if the example ends up being routed to a leaf with a very uniform (high entropy) class distribution.[1]

Very briefly, C4.5 learns trees as follows. It considers partitioning the data on the basis of a given attribute (e.g., dividing up the blobs as being produced by the red filter, the orange filter, etc.), and then measuring the quality of that split by computing the entropy of the class distributions in each of the resulting sets. The attribute which reduces the entropy (or some related quantity) the most is chosen. This step can be represented as a node with many branches leading to sub-nodes. Each sub-node is then recursively subdivided until the statistical significance of a split is too low.

Continuous data is handled using binary splits, as follows (see [9, 8] for more details). Suppose we are trying to split the data on the basis of mean saturation (S). We sort the vectors by their S values, and consider each value halfway between all the S values as a potential threshold t; all vectors for which $S < t$ go left, the rest go right. This split is evaluated as before, and the best threshold is chosen.

4.2 Experimental results

The horizontal component of the center of mass of the blob (x) is an irrelevant attribute (all of our classes are as likely to be on the left as on the right of the image). Unfortunately, because the training set was so small, the tree used x as a free parameter to try to fit the data better. Hence this attribute was removed by hand, reducing the dimensionality of the feature vector to 8, before C4.5 was re-run. The final tree is shown in Figure 4. There are a few interesting things to note about the tree.

- Visually similar classes often appear in the same subtree e.g., sky and water, or cloud and snow.
- The effects of too small a training set are painfully evident: e.g., every red blob is considered a flower, and every dark blue blob is considered sky. Returning the class probabilities instead of the majority label would be more informative, but is not possible in C4.5.
- The tree is quite large, despite pruning at the 99% confidence level. Forcing leaves to have a higher minimum count reduced performance.

We did not have enough data to create a separate test set, so instead we measured performance using 10-fold cross validation. The result was $58.71\% \pm 3.22\%$. (The error on the training set was 12.5% before pruning and 22.1% after pruning at the 99% confidence level.) The class confusion matrix on the training data is shown in Figure 5. Off-diagonal entries correspond to misclassifications. Notice that all 8 of the lawn

[1] Unfortunately, C4.5 does not do this, although it does compute some kind of certainty factor.

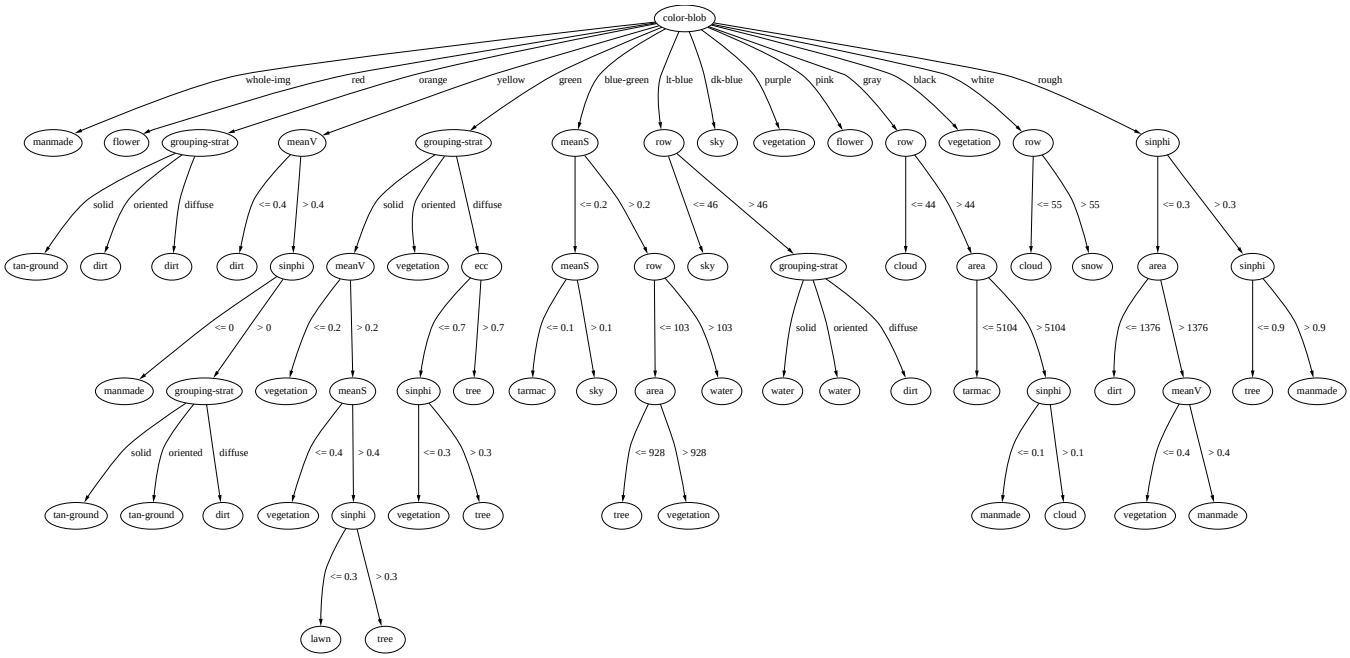

Figure 4: The decision tree. Row denotes the vertical position of the blob: smaller numbers are nearer the top of the screen. Sinphi denotes $|\sin(\phi)|$, where ϕ is the orientation of the ellipse which we fit to the blob; 0 corresponds to horizontal.

examples were misclassified as vegetation. Similarly, 8 out of 21 trees were misclassified as vegetation. This suggests that our features were not rich enough to discriminate these two classes. Notice also that 2 out of 8 cloud examples were misclassified as sky, probably because of noise in the training data.

4.3 Implementation details

We wrote a Perl program to convert the (text) tree produced by C4.5 into a series of if-then rules which could be used by Matlab. We used the MLC++ package [10] from Stanford/SGI and the dotty package [7] from Bell Labs for drawing the trees.

5 Conclusion

Our goal in this project was to demonstrate a system for automatic image annotation based on a small number of simple grouping strategies. While there is clearly more work to be done, the initial results are encouraging. By associating groups of color and texture features with blob-like image regions, we found that our decision tree classifier was able to produce plausible rules for detecting several basic concepts in real-world images. The same type of semantically meaningful result is not to be found in a system based solely on pixel-level concept classification.

A brief inspection of the decision tree we produced will indicate the importance of adding more training examples, as discussed in Section 4.2. We hope to add many more training examples in the future.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)	<-classified as
5				1	2							(a): class cloud
	9	1		1			2					(b): class dirt
		12						1				(c): class flower
										8		(d): class lawn
	1			27			2		1	1		(e): class manmade
1					28				1	1		(f): class sky
						6		1		1		(g): class snow
							13					(h): class tan-ground
				1				8		2	1	(i): class tarmac
1	1								8	11		(j): class tree
				1						36		(k): class vegetation
									1	1	10	(l): class water

Figure 5: The class confusion matrix.

One fundamental limitation of the system has to do with the interplay between the early-visual processing and the grouping stage which follows it. Namely, any concept which we can expect to identify via a successful grouping operation presupposes that at least one blob produced by the early-visual processing stage will contain this concept. As such, a close-up image of a chessboard, which contains several squares of color in distinct locations, would at best be grouped as several independent squares of color. To address this problem, another layer of grouping and/or reasoning must be added to the system to account for spatial relationships between simple concepts.

In future work, we intend to investigate learning strategies such as Bayes' nets and decision graphs to achieve this level of image understanding. We also intend to augment the set of early visual processing routines to provide richer features to the ensuing grouping stage. Finally, we hope to integrate several new grouping strategies (e.g. using texture as a repeating pattern [3] or axes of symmetry) which will extend the capabilities of the system to locate more complicated concepts.

A Definition of the Color/Texture Bins

red	$H < 0.07$ or $H \geq 0.95$
orange	$0.07 \leq H < 0.1$
yellow	$0.1 \leq H < 0.155$
green	$0.155 \leq H < 0.355$
blue-green	$0.355 \leq H < 0.455$
light blue	$0.455 \leq H < 0.625$
dark blue	$0.625 \leq H < 0.75$
purple	$0.75 \leq H < 0.825$
pink	$0.825 \leq H < 0.95$
gray	$S < 0.1$ and $V < 0.95$
black	$V < 0.05$
white	$S < 0.1$ and $V > 0.95$
rough	$0.002 < \lambda_{min} < .007$ and $\lambda_{max} < 0.03$

References

[1] David A. Forsyth, Jitendra Malik, Margaret M. Fleck, Thomas Leung, Chris Bregler, Chad Carson and H. Greenspan, "Finding Objects by Grouping," Submitted to the Workshop on Object Recognition, Cambridge, April 1996.

[2] M.M. Fleck, D.A. Forsyth, C. Bregler, "Finding Naked People," Proc. 4th European Conf on Computer Vision, 1996

[3] T.K. Leung and J. Malik, "Detecting, Localizing and Grouping Repeated Scene Elements from an Image," Fourth European Conf. Computer Vision, 1996, Cambridge, England.

[4] G. Granlund and H. Knutsson, *Signal Processing for Computer Vision*, Kluwer, 1995.

[5] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[6] J. Gårding and T. Lindeberg, "Direct computation of shape cues using scale-adapted spatial derivative operators," International Journal of Computer Vision, vol. 17, no. 2, pp. 163–191, 1996.

[7] E. Koutsofios and S. North, "Drawing graphs with dot", available from `www.research.att.com/orgs/ssr/book/reuse`.

[8] J. Dougherty, R. Kohavi and M. Sahami, *Supervised and Unsupervised Discretization of Continuous Features*, Proc. of the Conference on Machine Learning, 1995.

[9] U. M. Fayyad and K. B. Irani, *Multi-interval discretization of continuous-valued attributes for classification learning*, Intl. Joint Conf. on AI, pp. 1022–1027, 1993.

[10] R. Kohavi, G. John, R. Long, D. Manley and K. Pfleger, "MLC++: A machine learning library in C++", in *Tools with Artificial Intelligence*, pp. 740–743, 1994. available from `http://robotics.stanford.edu:/users/ronnyk/mlc.html`

[11] J. R. Quinlan, *C4.5 Programs for Machine Learning*, Morgan Kauffman, 1993.

[12] R. Rimey and C. Brown, "Task-oriented Vision and Multiple Bayes Nets", in *Active Vision*, eds. A. Blake and A. Yuille, MIT Press, 1992.