Report Number: WUCS-87-25

1987-10-01

Image Classification Using Laws' Texture Energy Measures

Will D. Gillett

IMAGE CLASSIFICATION USING LAWS' TEXTURE ENERGY MEASURES

Will D. Gillett

WUCS-87-25

October 1987

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

This research was supported in part by the Defense Mapping Agency under contract number DMA800-85-C-0010.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Overview

This working paper reports several aspects of a series of experiments using Laws' texture energy measures [1] (TEM) to classify texture in aerial photographs. Within this work, the standard training set paradigm was used. Results will be presented which (a) indicate the accuracy of Laws' TEM when the training set is in the *same* image as the points which are to be classified, (b) indicate the accuracy when the training set is in a *different* image than the points which are to be classified (when the images are in the same photographic space and different photographic spaces), and (c) show the results of classifying an entire 512×512 image, both when the training set is in that image and when it is not.

Previous work has been done to determine the effectiveness of Laws' TEM for classifying texture. This is an extension of that work. For a description of the training set paradigm and a comparison between co-occurrence matrix methods and Laws' TEM, see [2].

1.1. Images

Three images were used in these experiments. These three images are portions of a stereo pair (left and right photograph) of the Washington, DC area. The left photograph and right photograph will be termed to reside in *different photographic spaces* because they were taken at different times of the day and have different photographic properties. Specifically, the left photograph has a morning haze which tends to obscure features in the photograph, while the right image has no morning haze and is much sharper. There are other artifacts which differentiate them also, such as the length of shadows and the presence of cars.

The image shown in Figure 1, **image.r**, is a 512×512 "chip" from the right photograph showing a portion of the Washington DC area around the Lincoln Memorial. Figure 2, **image.l**, is the corresponding "chip" from the left photograph. Figure 3, **image.o**, is a "chip" from the right photograph just above **image.r** and overlaps it slightly; this third image was chosen

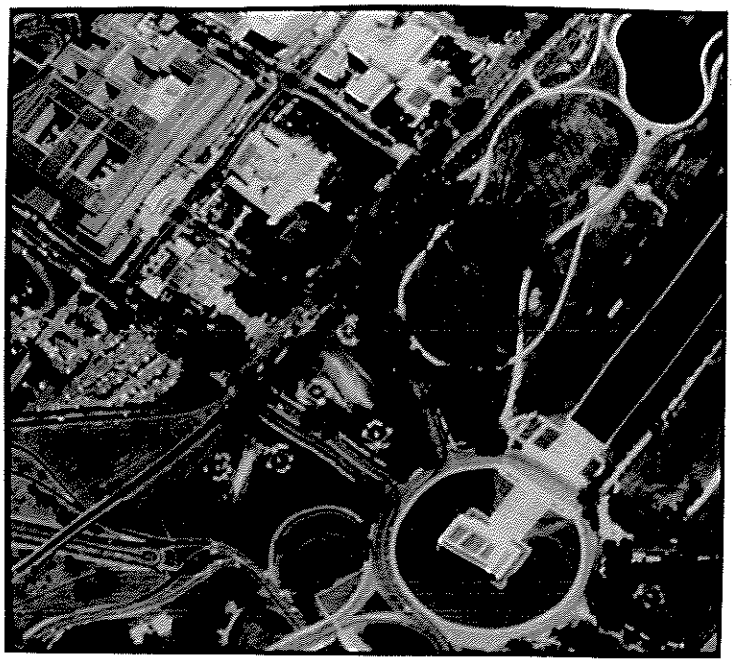

Figure 1: **image.r**

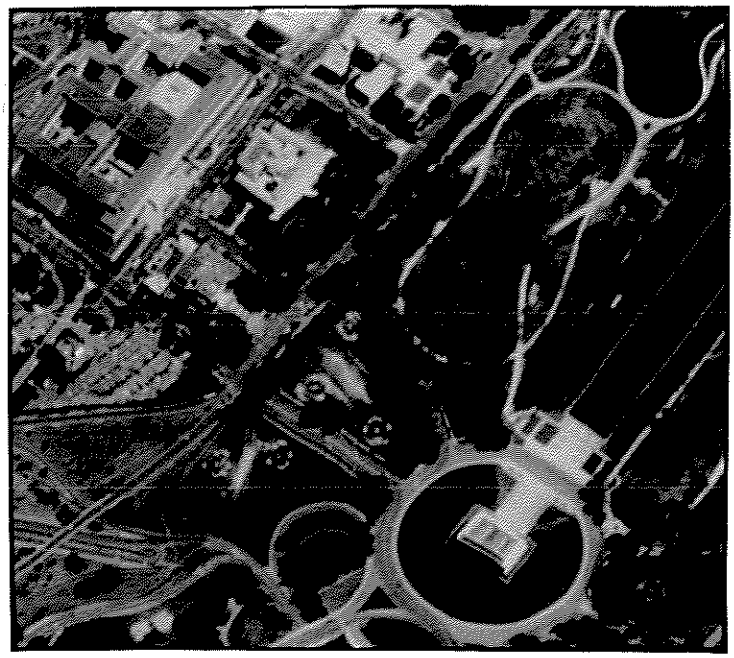

Figure 2: **image.l**

because it contains all the texture fields present in the other two "chips".

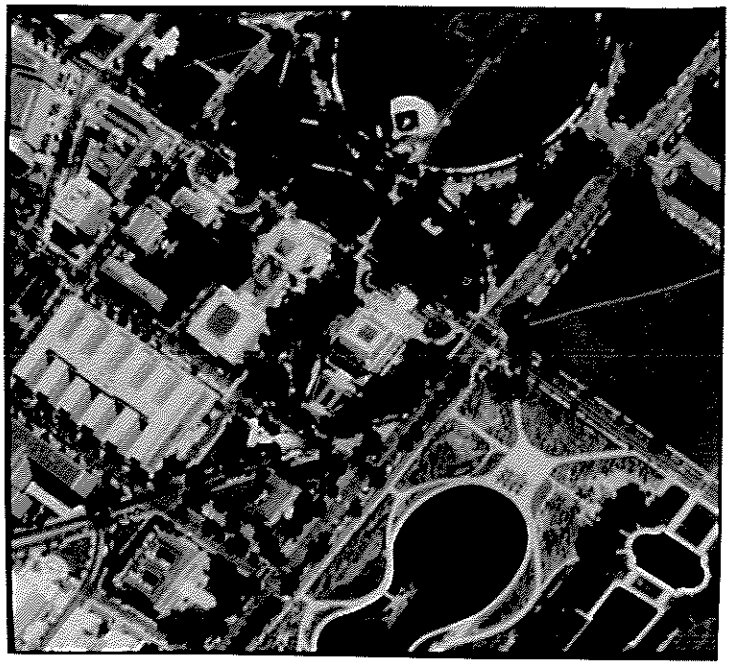

Figure 3: **image.o**

All three images will be used to determine the accuracy of Laws' TEM as training/test sets are used within an image and across images. All pixels in **image.r** and **image.o** will be classified using a training set of points selected from **image.r**.

1.2. Training/Test Sets

Ten training/test sets were developed for use in these experiments: `ctts1.r`, `ctts2.r`, and `etts.r` reside in **image.r**; `ctts1.l`, `ctts2.l`, and `etts.l` reside in **image.l**; `ctts1.o`, `ctts2.o`, and `etts.o` reside in **image.o**; `btts.r` resides in **image.r**. The training/test sets `ctts1.r`, `ctts2.r`, `ctts1.l`, `ctts2.l`, `ctts1.o`, `ctts2.o`, `etts.o` and `btts.r` will be designated as *central* training/test sets because their points lie in the center of texture fields. The training/test sets `etts.r`, `etts.l`, and `etts.o` will be designated as *edge* training/test sets because their points lie near the edge of texture fields (2 or 3 pixels from the edge). (`ctts` stands for *central training/test set,* `etts` stands for *edge training/test set,* and `btts` stands for *big training/test set.*) The first nine of these training/test

sets are based on a set of seven pre-selected categories: tree, grass, water, roof, earth, road, and shadow. Each contains 84 point, 12 points from each of the seven categories. These training/test sets will be used to determine the accuracy of Laws' TEM and will not be used to classify an entire image.

The last training/test set, `btts.r`, is based on a more refined category structure. It contains 208 points from 47 pre-selected categories; the 47 categories can be abstracted into 11 supercategories. This concept of category vs. supercategory is introduced because of the human semantics or perceptions placed on texture. More specifically, two different regions of an image that humans might claim are both grassy areas may have significantly different texture (even to the human) even though we would conceptually wish them to be placed into the same semantic category, grass. For this reason, the conceptual categories (*supercategories*) are broken down into conceptual subcategories (*categories*) having similar actual texture. The conceptual subcategories are easily recombined back into their abstracted conceptual categories within the software. The 11 supercategories in this training/test set are: tree, water, earth, grass, bush, roof, steps, wall, parking-lot, road, and shadow. The 47 categories (along with the number of points in each) are: tree1 (4), tree2 (7), tree3 (3), tree4 (3), water1 (5), water2 (5), earth1 (5), earth2 (2), grass1 (4), grass2 (5), grass3 (7), grass4 (4), grass5 (3), grass6 (5), grass7 (3), bush1 (4), bush2 (4), bush3 (5), roof1 (9), roof2 (6), roof3 (3), roof4 (4), roof5 (3), roof6 (6), steps1 (8), steps2 (3), wall1 (4), wall2 (3), wall3 (4), wall4 (3), wall5 (3), parking-lot (6), road1 (4), road2 (5), road3 (4), road4 (7), road5 (3), road6 (4), road7 (4), road8 (4), road9 (3), road10 (4), road11 (4), road12 (4), road13 (3), shadow1 (6), shadow2 (6).

The training/test sets `ctts1.r` and `etts.r` reside in **image.r** and are shown in Figures 4 and 5, respectively. For brevity, `ctts2.r` is not shown, but it is similar to `ctts1.r`. These are the same training sets used in the comparative study of [2]. The training/test sets `ctts1.l`, `ctts2.l`, and `etts.l` (not shown) reside in **image.l**; the points in these training/test sets correspond to exactly the same point present in training/test sets `ctts1.r`,

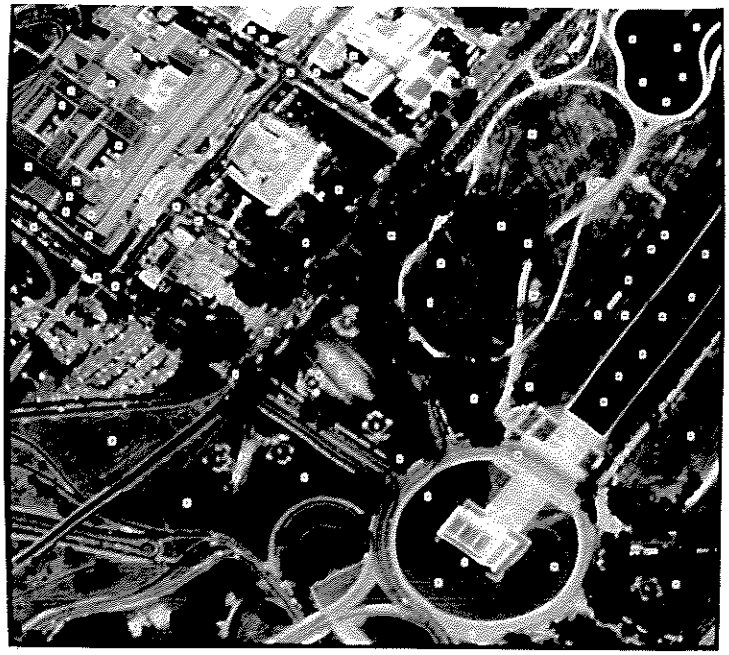

Figure 4: Training/Test Set `ctts1.r`

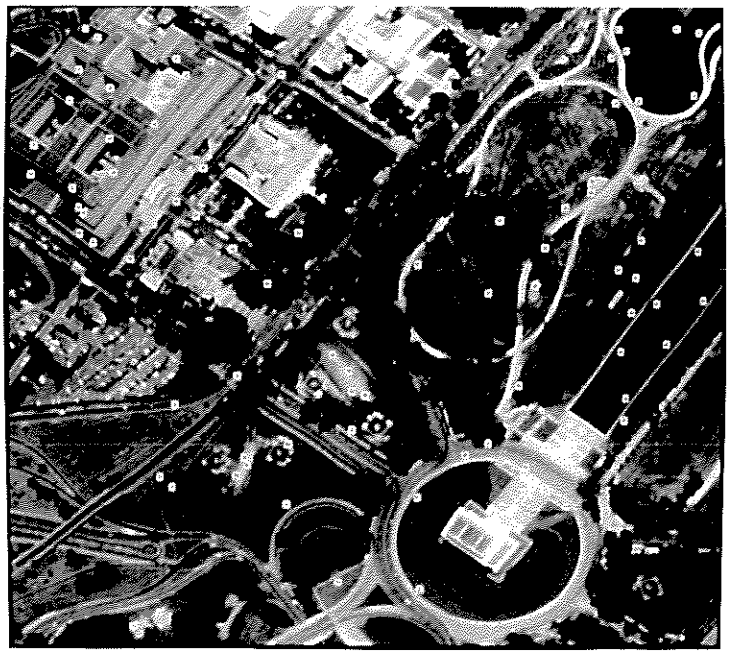

Figure 5: Training/Test Set `etts.r`

`ctts2.r`, and `etts.r`, respectively; the correspondence was done by human visual inspection.

The training/test sets `ctts1.o`, `ctts2.o`, and `etts.o` (not shown) reside in **image.o**.

There is no relationship between the positions of points in these training/test sets and the positions of points in any other training/test set.

The last training/test set, `btts.r`, contains 208 points in 47 categories; these points lie in the center of texture fields. It will be used as the training set for classifying the entire images of **image.r** and **image.o**. It is not shown here for brevity and because it is difficult to specify in photographic form the semantics of the 47 categories.

2. Laws' Texture Energy Measures

In this section, we discuss Laws' general approach, and our specific implementation of it.

2.1. Laws' General Approach

Laws found, after analyzing many texture analysis methods, that certain 1-dimensional convolution masks identified or extracted certain important properties of an image that are useful in texture analysis. Specifically, the four most important that he found were:

$$L5 = [1 \;\; 4 \;\; 6 \;\; 4 \;\; 1]$$

$$E5 = [-1 \;\; -2 \;\; 0 \;\; 2 \;\; 1]$$

$$S5 = [-1 \;\; 0 \;\; 2 \;\; 0 \;\; 1]$$

$$R5 = [1 \;\; -4 \;\; 6 \;\; -4 \;\; 1]$$

These convolution masks enhance intensity level, edges, spots, and ripples, respectively, in any specific direction that the mask is applied. The cross product of these four masks in the horizontal and vertical directions produce 16 2-dimensional convolution masks, which he found extracted information useful in texture analysis. As an example, the E5E5 2-dimensional convolution mask is given by:

$$\begin{bmatrix} 1 & 2 & 0 & -2 & -1 \\ 2 & 4 & 0 & -4 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -4 & 0 & 4 & 2 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

The L5L5 mask does not produce useful information for direct texture analysis, but was found useful for normalizing the other 15 convolutions as a function of intensity level. Laws found the other 15 masks to be useful in texture analysis, and these are the ones we have used in our implementation.

The images produced by applying the 15 convolutions masks to the original image are not directly useful in texture analysis because the information at each pixel is very local to that pixel (no more than two pixels away). However, it does extract local artifacts of the original image (e.g., edges, spots, and ripples). In essence, a blurring affect is needed to collect information from more distant pixels. Taking a standard deviation over a macrowindow of size **n×n** produces an appropriate blurring affect by pulling in information from adjacent pixels. However, taking the standard deviation is a computationally intensive process; a very close approximation can be obtained by instead taking an average (over the same **n×n** macrowindow) of the absolute value of the convoluted image. This is much less computationally intensive, and is the method that we have chosen.

Using these 15 images (i.e., the averages over the absolute value of the convolution for the 15 different convolution masks), each pixel can be considered to be a vector in 15-dimensional space. Using these vectors as his definition of texture space, he performed a texture analysis by specifying predefined categories, selecting a training set, performing a discriminant analysis[3], and classifying an entire image based on closeness to category centers in discriminant space.

2.2. Implementation

In this subsection, we describe our implementation of Laws' TEM (used in the experiments presented in Section 3), and present a pictorial representation of convolution images and averaged images. The important components of the implementation will be presented by giving a scenario of the computational activities, in the sequence in which they occur.

Step 1: Convolution Masks

Fifteen different 2-dimensional convolution masks are applied to the original image. Each convolution is performed by a standalone preprogrammed convolution process. We have convolution software (`conv`) that runs directly on the VAX/750 and separate software (`dvpconv`) that runs on the Digital Video Processor (DVP) of the DeAnza[4]. We have created the 15 convolution images (**convolutions**) using the VAX/750 software. Each of these convolutions extracts or enhances certain artifacts of the original image. As an example of what these convolution masks extract, the results of applying convolution mask E5E5 to **image.r** (shown in Figure 1) is shown in Figure 6; the E5E5 mask extracts edges in both the horizontal and vertical directions.

Step 2: Averaging over an **n**×**n** Window

The absolute value of each of the 15 convoluted images is then averaged over an **n**×**n** macrowindow (where **n** ranges from 5 to 15 in increments of 2). Again, this is performed by a standalone preprogrammed averaging software (`dvpavg`); in this case we have created the 15 averaged images (**averages**) using a version that runs on the DVP of the DeAnza. This averaging has the affect of blurring the image and pulling in information from distant pixels (a maximum of 7 pixels away). The averaging over a 15×15 pixel macrowindow for the convoluted image shown is Figure 6 is presented in Figure 7. Note the blurring affect.

The results of Step 2 supply the preprocessed information needed by the remaining processing, including category and training set selection, discriminant analysis, and classification. As

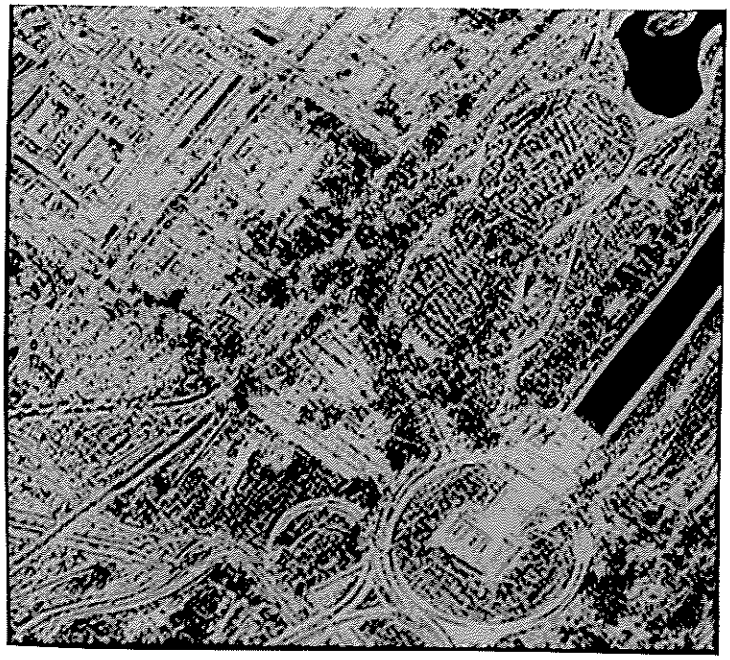

Figure 6: Convolution Of **image.r** With E5E5 Mask

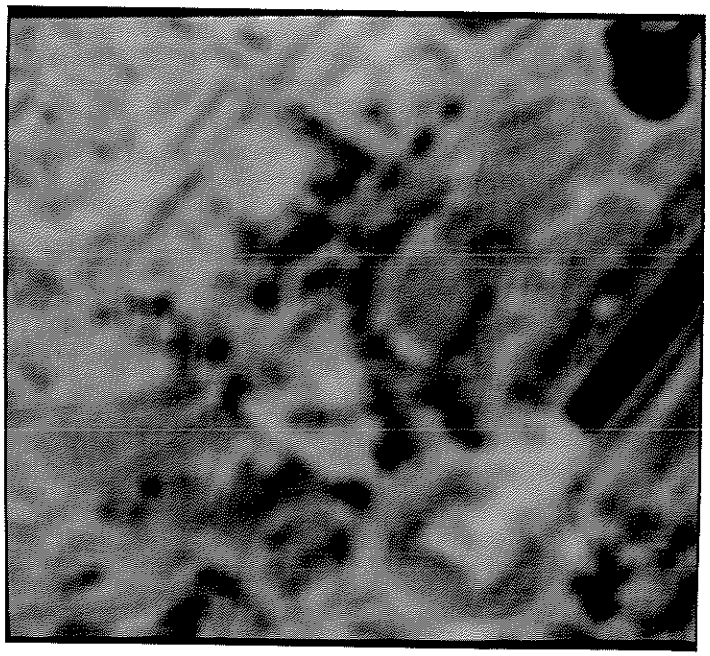

Figure 7: Figure 6 Averaged Over A 15x15 Macrowindow

subsequent processing changes (e.g., using different training/test sets), these two steps need not be repeated.

Step 3: Category and Training Set Selection

The seven categories and various combinations of the nine training/test sets are used for different analyses.

Step 4: Discriminant Analysis

A discriminant analysis[3] based on the texture space of a specific training set is performed. We are currently using an interactive statistical package called S[5] to perform this analysis; however, we suspect that the discriminant analysis of any given statistical package would be sufficient for this computational component. S has been very useful to us in understanding the nature of our training sets because it is easy to interactively compute and display (in both tabular and graphical form) the results of our analyses. The insights that can be obtained from such an interactive statistical tool have been invaluable. The discriminant analysis in S produces a transformation (in the form of a 15×15 matrix) of texture space into discriminant space.

Step 5: Classification

A classification analysis is performed both in the original texture space and in discriminant space. Centers of categories are calculated in the original texture space. Then each point of the given training set is transformed into discriminant space and centers of the categories are calculated there.

Each point in a given test set is compared against the centers of the categories in the original texture space, and classified into the corresponding category there. Then each point is transformed into discriminant space and compared against the centers of the categories in discriminant space.

Calculations of the centers, transformation into discriminant space, and comparisons to categories are all done in S.

3. Accuracy of Laws' TEM

There are a number of important questions that can be asked about a texture classification scheme based on a training set paradigm. How sensitive is the accuracy of the classification to the selection of a specific training set? What happens as training point or test points approach the edge of a texture field? Do the photographic properties (e.g., sharp or hazy) of an image affect the accuracy? How is accuracy affected if the training set is selected from the image in which points are to be classified vs. being selected from a different image? These are the kinds of questions that will be addressed in this section.

Figure 8 presents the scheme for the tables that will be presented showing the percentage of correct classification that occurs for combinations of training/test sets. The **CE-CE** region corresponds to central training/test sets being used as both training sets and test sets. The **ED-CE** region corresponds to an edge training/test set being used as a training set and central training/test sets being used as test sets. The **CE-ED** region corresponds to central training/test sets being used as training sets and an edge training/test set being used as a test set. The **ED-ED** region corresponds to an edge training/test set being used as both a training set and a test set.

<table>
<tr><td></td><td></td><td colspan="3">Training Set</td></tr>
<tr><td>Test Set</td><td>Window Size</td><td>ctts1</td><td>ctts2</td><td>etts</td></tr>
<tr><td>ctts1</td><td>5-15</td><td colspan="2" rowspan="2">CE-CE</td><td rowspan="2">ED-CE</td></tr>
<tr><td>ctts2</td><td>5-15</td></tr>
<tr><td>etts</td><td>5-15</td><td colspan="2">CE-ED</td><td>ED-ED</td></tr>
</table>

Figure 8: Scheme For Percentage Of Correct Classification Tables

3.1. Accuracy Within an Image

In this section, data are presented to indicate the accuracy of Laws' TEM when the training set and test set are selected from the same image. Table 1 presents percentage of correct classification data (extracted from **image.r**) for training/test sets `cttsl.r`, `ctts2.r`, and `etts.r` combined in all combinations. Note that, as a general rule, accuracy tends to increase as window size increases. Comparing main diagonal entries to off diagonal entries, observe that the percentage of correct classification (both in texture space and discriminant space) tends to be highest when a training/test set is used both as a training set and as a test set, as might be expected. This is even true for the `etts.r` training/test set, although the

Table 1
Percentage Correct Classification Within **image.r**

		Training Set (texture, discriminant)		
Test Set	Window Size	`cttsl.r`	`ctts2.r`	`etts.r`
`cttsl.r`	5	71,88	64,70	46,63
	7	75,89	68,79	46,67
	9	80,90	69,82	48,74
	11	81,94	70,88	46,73
	13	82,94	75,90	54,76
	15	79,94	76,88	56,76
	Summary	71-82,88-94	64-76,70-90	46-56,63-76
`ctts2.r`	5	58,68	74,81	42,50
	7	68,79	75,82	46,62
	9	80,86	80,88	48,76
	11	82,86	77,98	48,77
	13	87,88	86,96	55,82
	15	87,87	85,96	58,80
	Summary	58-87,68-88	74-86,81-98	42-58,50-82
`etts.r`	5	42,38	35,38	64,82
	7	44,50	42,44	65,81
	9	45,54	44,50	63,86
	11	49,52	43,49	62,85
	13	48,55	45,48	63,86
	15	46,51	49,52	63,86
	Summary	42-49,38-55	35-49,38-52	62-65,81-86

accuracy is somewhat less than that for the central training/test sets.

Restricting our attention to the **CE-CE** region, main diagonal entries range from 71-86% correct classification in texture space and 81-98% correct classification in discriminant space. Off diagonal entries (where the training set is not the test set) range from 58-87% correct classification in texture space and 68-90% correct classification in discriminant space. This indicates a slight drop in accuracy in texture space when a test set other than the training set is used, and a more significant drop in accuracy for discriminant space. However, the off diagonal entries are the only ones that are really relevant to analyze, since in a production environment one attempts to classify points for which the category is not known.

Moving our attention to the `etts.r` training/test set, we see that accuracy always drops off when edges of texture fields are involved, as might be expected. Note however that accuracy is significantly higher (in both texture space and discriminant space) in the **ED-CE** region than in the **CE-ED** region. This indicates that it is not extremely detrimental to have training points near the edge of texture fields, but whether or not edge training points are present, it is difficult to classify edge test points with high accuracy.

Table 2 presents similar information for **image.l** using the corresponding training/test sets `cttsl.l`, `ctts2.l`, and `etts.l`. Although the actual data presented here are slightly different, the pattern suggested above for **image.r** seems to be identical here.

Table 3 presents percentage of correct classification data for the training/test sets `cttsl.o`, `ctts2.o`, and `etts.o` in **image.o**. The data presented here again exhibit patterns similar to Tables 1 and 2.

Table 2
Percentage Correct Classification Within **image.l**

Test Set	Window Size	Training Set (texture, discriminant) ctts1.l	ctts2.l	etts.l
ctts1.l	5	71,82	67,64	49,45
	7	74,89	70,70	52,56
	9	79,82	71,85	56,62
	11	77,86	76,81	57,65
	13	80,94	79,76	60,70
	15	82,96	77,83	64,74
	Summary	71-82,82-96	67-79,64-85	49-64,45-74
ctts2.l	5	70,62	81,89	49,58
	7	71,75	81,90	51,52
	9	79,80	79,93	50,63
	11	77,81	79,90	54,71
	13	80,87	80,92	60,71
	15	85,87	81,95	64,76
	Summary	70-85,62-87	79-81,89-95	49-64,52-76
etts.l	5	39,38	39,29	56,70
	7	42,39	37,31	52,70
	9	38,40	37,35	56,74
	11	39,46	33,40	60,80
	13	38,49	33,42	61,83
	15	40,56	37,46	58,85
	Summary	38-42,38-56	33-39,29-46	52-61,70-85

Table 3
Percentage Correct Classification Within **image.o**

Test Set	Window Size	Training Set (texture, discriminant) cttsl.o	ctts2.o	etts.o
cttsl.o	5	69,86	63,65	42,49
	7	63,88	71,82	42,62
	9	64,92	69,88	40,64
	11	70,92	70,85	37,65
	13	74,94	70,89	37,75
	15	76,90	75,88	39,76
	Summary	63-76,86-94	63-75,65-89	37-42,49-76
ctts2.o	5	64,70	69,88	46,58
	7	62,75	73,90	48,63
	9	67,85	76,94	43,68
	11	79,83	79,95	44,71
	13	83,92	86,95	44,77
	15	85,90	82,95	42,80
	Summary	62-85,70-92	69-86,88-95	42-48,58-80
etts.o	5	33,40	37,39	63,71
	7	33,38	36,54	57,80
	9	38,45	39,50	58,83
	11	42,49	37,54	61,81
	13	44,56	35,46	68,83
	15	45,55	37,52	65,83
	Summary	33-45,38-56	35-39,39-54	57-68,71-83

3.2. Accuracy Across Images

In this section, we examine the accuracy of Laws' TEM across images, i.e., when a training set from one image is used to classify a test set in another image. In this analysis, all three images, **image.r**, **image.l**, and **image.o**, will be used. Training sets and test sets will be selected from each image to see if results are symmetric as the training set image changes.

3.2.1. Similar Photographic Spaces

In this section, we examine accuracy across images in the same photographic space. The only image pair that resides in the same photographic space is (**image.r**, **image.o**). Table 4 presents percentage of correct classification data for training sets selected from **image.r** and

test sets selected from **image.o**. Table 5 presents percentage of correct classification data for training sets selected for **image.o** and test sets selected from **image.r**. Note that the overall accuracy indicated by the data in Tables 4 and 5 is lower than that for Tables 1, 2 and 3 (even though there are a few random entries that indicate higher accuracy). This indicates that some accuracy is lost as we move across images. Of course, we had hoped that no loss of accuracy would occur across images in similar photographic spaces, but this seems not to be the case. This loss of accuracy might be due to the training/test sets, although we believe that this is not the case. We suspect that this loss of accuracy is actually due to real changes in perceived textures across the images. Such texture changes might occur because of: (a) a different sun angle, (b) a different photographic angle, (c) a different watering pattern for foliage, or (d) a

Table 4
Percentage Correct Classification From **image.r** To **image.o**

		Training Set (texture, discriminant)		
Test Set	Window Size	`ctts1.r`	`ctts2.r`	`etts.r`
`ctts1.o`	5	48,58	45,56	26,42
	7	53,70	56,62	26,44
	9	61,75	55,70	29,49
	11	56,73	55,71	31,55
	13	52,79	52,76	31,51
	15	54,82	54,79	38,49
	Summary	48-61,58-82	45-56,56-79	26-38,42-55
`ctts2.o`	5	52,55	48,52	30,37
	7	60,61	56,61	29,42
	9	63,71	61,67	30,51
	11	67,69	64,71	31,57
	13	64,76	67,80	36,57
	15	68,80	65,80	43,50
	Summary	52-68,55-80	48-67,52-80	29-43,37-57
`etts.o`	5	40,38	40,37	45,51
	7	38,44	45,42	48,57
	9	36,39	42,38	54,60
	11	35,44	31,37	50,60
	13	32,51	35,48	49,62
	15	29,52	31,51	45,58
	Summary	29-40,38-52	31-45,37-51	45-54,51-62

Table 5
Percentage Correct Classification From **image.o** To **image.r**

Test Set	Window Size	Training Set (texture, discriminant) ctts1.o	ctts2.o	etts.o
ctts1.r	5	42,50	44,50	51,51
	7	43,56	48,50	54,50
	9	49,57	49,63	49,60
	11	55,67	55,63	49,61
	13	65,70	62,64	51,68
	15	64,75	63,70	56,73
	Summary	42-64,50-75	44-63,50-70	49-56,50-73
ctts2.r	5	45,46	43,52	51,52
	7	50,57	46,54	51,55
	9	52,60	57,57	51,57
	11	58,67	62,69	52,61
	13	68,77	64,67	52,67
	15	70,73	65,76	60,74
	Summary	45-70,46-77	43-65,52-76	51-60,52-74
etts.r	5	23,30	23,35	49,49
	7	17,31	25,37	49,51
	9	23,40	27,44	46,64
	11	30,38	27,40	46,56
	13	33,37	33,36	51,56
	15	33,32	36,38	46,55
	Summary	17-33,30-40	23-36,35-44	46-51,49-64

different usage pattern for foliage. Specifically, in these two images, there is a difference between the "richness" of grass and trees that is visually evident. Grass and trees in **image.o** seem to be more uniform and darker than grass and trees in **image.r**.

Contrasting Tables 4 and 5 with themselves, we see a surprising lack of symmetry. In the **CE-*** region of Table 4, the accuracies are significantly higher than those of Table 5. However, in the **ED-CE** region, the reverse is true. We have no explanation for this asymmetry. However, this may suggest a methodology for determining an image (or images) from which the "best" training sets should be selected. In this case, **image.r** is clearly the better image, since we are primarily interested in the **CE-*** region.

3.2.2. Dissimilar Photographic Spaces

In this section, we examine accuracy across images in different photographic spaces. Here, we have two such image pairs, (**image.r**, **image.l**) and (**image.l**, **image.o**). Tables 6 and 7 present the percentage of correct classification data for the (**image.r**, **image.l**) image pair. Comparing these tables to Tables 4 and 5, we see a significant drop in accuracy (we assume due to the dissimilar photographic spaces), as might be expected.

Contrasting Tables 6 and 7 with themselves, we see the similar pattern of asymmetry as was present in Tables 4 and 5 of the previous subsection. Here, the discriminant space accuracies for Table 6 are uniformly higher than those of Table 7. However, note that the

Table 6
Percentage Correct Classification From **image.r** To **image.l**

		Training Set (texture, discriminant)		
Test Set	Window Size	ctts1.r	ctts2.r	etts.r
ctts1.l	5	39,55	43,55	17,42
	7	40,58	44,60	19,39
	9	39,54	43,61	21,40
	11	34,56	42,61	24,39
	13	33,52	39,57	24,40
	15	30,56	37,61	25,36
	Summary	30-40,52-58	37-44,55-61	17-25,36-42
ctts2.l	5	39,49	38,50	19,33
	7	39,54	40,61	19,40
	9	37,52	38,62	21,36
	11	32,51	38,58	25,36
	13	30,50	38,58	26,32
	15	30,50	36,57	26,33
	Summary	30-39,49-54	36-40,50-62	19-26,32-40
etts.l	5	32,33	29,35	21,38
	7	32,31	25,31	23,40
	9	26,25	23,31	26,37
	11	20,26	19,31	26,35
	13	19,26	23,26	26,31
	15	18,27	19,30	25,30
	Summary	18-32,25-33	19-29,26-35	21-26,30-40

Table 7
Percentage Correct Classification From **image.l** To **image.r**

Test Set	Window Size	Training Set (texture, discriminant) cttsl.l	ctts2.l	etts.l
cttsl.r	5	50,43	38,35	51,35
	7	45,43	32,31	57,30
	9	38,36	33,32	57,31
	11	40,40	35,30	50,29
	13	37,42	30,32	43,29
	15	39,43	30,32	44,29
	Summary	37-50,36-43	30-38,30-35	43-57,29-35
ctts2.r	5	48,40	39,39	44,36
	7	40,38	37,36	52,33
	9	39,39	37,32	50,29
	11	40,45	35,32	49,30
	13	40,43	35,33	43,29
	15	43,43	35,30	42,29
	Summary	39-48,38-45	35-39,30-39	42-52,29-36
etts.r	5	29,21	20,17	36,33
	7	21,18	18,18	33,29
	9	19,20	17,20	35,35
	11	20,26	17,19	32,32
	13	19,24	15,25	32,27
	15	17,24	15,23	31,29
	Summary	17-29,18-26	15-20,17-25	31-36,27-35

texture space accuracies for the `cttsl.r` and `etts.r` training sets are generally higher in Table 7 than in Table 6. Since we are primarily interested in discriminant space, **image.r** seems to be the preferred image from which to select training sets.

Looking at Tables 6 and 7 more closely, we note an interesting pattern in the texture space entries of the **CE-*** region. There seems to be a tendency for accuracy to decrease as window size increases. This pattern is even present in the **ED-*** region of Table 7 (and the **CE-ED** region of Table 4). This is in contrast to the discriminant space accuracies in which accuracies are relatively stable or tend to increase as window size increases.

Tables 8 and 9 present percentage of correct classification data for the (**image.l**, **image.o**) image pair. Note that accuracies here are very similar to those present in Tables 6 and 7. Contrasting Tables 8 and 9 with themselves, we find that accuracies are higher in Table 9, indicating that **image.o** is the preferred image from which to select training sets, as might be expected. Looking at Tables 8 and 9 more closely, we note that the texture space pattern of decreasing accuracy as window size increases is not as prevalent as it was in Tables 6 and 7.

Table 8
Percentage Correct Classification From **image.l** To **image.o**

		Training Set (texture, discriminant)		
Test Set	Window Size	`cttsl.l`	`ctts2.l`	`etts.l`
`cttsl.o`	5	46,39	39,44	46,35
	7	49,40	42,38	43,37
	9	49,49	39,39	48,38
	11	49,42	40,38	45,36
	13	52,52	42,37	38,27
	15	49,48	44,43	36,30
	Summary	46-52,39-52	39-44,37-44	36-48,27-38
`ctts2.o`	5	44,37	32,40	39,35
	7	49,36	31,36	46,37
	9	40,39	37,35	46,35
	11	43,46	37,40	40,31
	13	45,50	37,35	36,27
	15	50,42	42,42	36,30
	Summary	40-50,36-50	31-42,35-42	36-46,27-37
`etts.o`	5	27,18	18,17	39,32
	7	18,14	15,14	36,33
	9	14,13	15,14	31,27
	11	17,12	15,17	29,29
	13	20,19	14,15	29,27
	15	23,18	18,17	26,27
	Summary	17-27,12-19	14-18,14-17	26-39,27-33

Table 9
Percentage Correct Classification From **image.o** To **image.l**

Test Set	Window Size	Training Set (texture, discriminant) ctts1.o	ctts2.o	etts.o
ctts1.l	5	40,39	37,40	21,37
	7	49,39	37,42	23,38
	9	51,42	31,52	23,36
	11	49,40	33,62	24,38
	13	45,55	32,61	25,38
	15	38,55	33,62	24,37
	Summary	38-51,39-55	31-37,40-62	21-25,36-38
ctts2.l	5	40,45	30,42	27,39
	7	42,44	31,45	26,37
	9	43,45	29,51	26,37
	11	40,48	27,60	27,36
	13	40,58	29,57	27,35
	15	39,51	33,61	27,36
	Summary	39-43,44-58	27-33,42-61	26-27,35-39
etts.l	5	33,29	24,29	36,36
	7	37,24	32,26	33,38
	9	37,29	29,26	30,42
	11	26,21	21,29	30,39
	13	23,23	20,27	30,43
	15	23,18	23,29	25,43
	Summary	23-37,18-29	20-32,26-29	25-36,36-43

3.3. Categories vs. Supercategories

In the next major section, entire images will be classified. In performing this classification the concept of abstracting categories into supercategories is introduced. In order to support the effectiveness of this abstraction, this section presents percentage of correct classification data for the large training/test set `btts.r` when the points are perceived to reside in different abstractions of categories.

Table 10 presents percentage of correct classification data for training/test set `btts.r`. Here, the 47 categories are considered to be stand alone categories for which no abstraction (into the 11 supercategories) is performed. Note that the accuracies here are less that those of

Table 10
Percentage Correct Classification -- 47 Categories

Test Set	Window Size	Training Set (texture, discriminant) btts.r
btts.r	5	64,77
	7	63,75
	9	69,80
	11	72,83
	13	75,85
	15	75,88
	Summary	63-75,75-88

the diagonal entries of the **CE-CE** regions of Tables 1, 2, and 3.

Table 11 presents percentage of correct classification data for this same training/test set except that after preliminary classification has been performed, similar categories are abstracted into the appropriate supercategories. For example, any points classified into the categories of tree1, tree2, tree3, or tree4 are now classified into the supercategory of tree. Note that the accuracies present in this table are comparable to those of the diagonal entries of the **CE-CE** regions of Tables 1, 2, and 3. Thus, we are able to retain high accuracies by breaking conceptual semantic categories down into different physical categories (with similar texture) and

Table 11
Percentage Correct Classification -- 47 Categories (11 Supercategories)

Test Set	Window Size	Training Set (texture, discriminant) btts.r
btts.r	5	74,83
	7	76,85
	9	78,85
	11	81,89
	13	82,90
	15	82,94
	Summary	74-82,83-94

recombining after the fact.

In contrast, if we combine the 47 categories into their 11 semantic categories before the preliminary classification analysis, we obtain the results shown in Table 12. Note that the accuracies indicated in this table are significantly less than those of Tables 10 and 11.

3.4. Conclusions

The training/test sets used in these experiments have not been "tuned" to produce maximal accuracies. They were selected in a random ad hoc manner, the way we assume that training sets might be selected in a production environment. It may be possible to develop a methodology in which preliminary training points are selected and then "moved" slightly by some algorithmic means until maximal accuracies are obtained (similar to many root finding methods). If this is done, higher accuracies can probably be achieved. In general, the data presented here does indicate that the selection of specific training sets does effect the accuracy of classification, but if training sets are selected in a "reasonable" manner, this is not the overwhelming factor.

Table 12
Percentage Correct Classification -- 11 Categories

		Training Set (texture, discriminant)
Test Set	Window Size	`btts.r`
`btts.r`	5	48,62
	7	48,61
	9	47,65
	11	50,66
	13	48,66
	15	50,72
	Summary	47-50,61-72

If training sets are selected from the actual image that is to be classified, accuracies in the range of 83-92% correct classification can be achieved; this range was arrived at by looking at the off diagonal entries of the **CE-CE** regions of Tables 1, 2, and 3 for window sizes of 13 and 15. This range may be a little high, because points near edges of texture fields will also be present (42-56% correct classification). The actual range will depend on the percentage of central points vs. edge points present in the specific image being classified.

If training sets are to be used across images having similar photographic properties, accuracies in the range of 64-82% correct classification (32-52% correct classification for edges of texture fields) can be achieved. This is somewhat lower, but still significantly high.

If training sets are to be used across images having dissimilar photographic properties, accuracies in the range of 30-61% correct classification (15-30% correct classification for edges of texture fields) can be achieved. These accuracies are low enough that it is probably not productive to attempt such classifications. Note that in the **CE-ED** regions of Tables 6 through 9, many of the entries indicate that the accuracy is no better than random chance, i.e., there is really no information retained in the training set paradigm.

Looking at all the information in all the tables, we also conclude that higher accuracies are obtained from images that have "sharp" photographic properties instead of "hazy" ones, as might be expected. We also note that the highest accuracy does not always occur for the largest window size of 15. Often, this is achieved for window sizes of 11 or 13. We suspect that this is an artifact of the specific images that we are analyzing, in which many of the actual texture fields are "narrow", e.g., narrow roads and shadows and small trees. We suspect that maximum accuracy has been achieved somewhere in the 11-15 window range; however, this may not be true for images with "wider" texture fields, and even higher accuracies may be achievable. Laws found that even higher accuracies were obtainable for window sizes in the 31 range. In general, our results are consistent with Laws' results.

The data presented here suggest the possibility of developing methodologies for determining "good" images from which to select training sets and for determining "good" training sets. Also, the technique of dividing semantic categories into different physical categories (with similar internal texture) seems to be effective as the number of categories and training points increases.

4. Classifying an Entire Image

In this section, we refine the computational scenario presented in Section 2.2, show pictorial results for classifying the entire images of **image.r** and **image.o**, and present a space and time analysis.

4.1. Implementation

The computational scenario for classifying an entire image is very similar to that of Section 2.2. Here, we present a refinement of that scenario.

Step 1: Convolution Masks

Same as that of Section 2.2.

Step 2: Averaging over a 15x15 Window

In this case, we average over only the maximum window size, i.e., a 15×15 window.

Step 3: Category and Training Set Selection

In this case, we use the `btts.r` training set containing 208 points over 47 categories abstracted into 11 supercategories.

Collection of training points is done by a C program called `laws`. It has three major inputs: (a) the 15 images (**averages**) produced by Step 2, (b) a file containing the 47

category names (`class.def`), and (c) a file containing the training set points (`btts.r`). Training set points are specified by: (i) an (x, y) pixel coordinate, (ii) a unique name, and (iii) the name of the category to which the point belongs.

It has two major outputs: (a) a 512×512 image (**overlay.train**) indicating the positions of the training points (this is how Figures 4 and 5 were produces), and (b) the collected data (in S format) to be analyzed by S. These data are written in four files: (i) the number of training points, (ii) the unique names of the points, (iii) the category names corresponding to the points, and (iv) the vectors in 15-dimensional space selected from the 15 images.

Step 4: Discriminant Analysis

Same as that of Section 2.2.

The discriminant analysis in S produces a transformation (in the form of a 15x15 matrix) of texture space into discriminant space. The texture space of the training set is transformed into discriminant space, and the transformation matrix and centers of the 47 categories are written to files (`transform` and `class.cent`) for use in subsequent processing (Step 5).

Step 5: Classification

Each pixel of the original image is transformed into discriminant space (using the transformation produced by the S analysis) and classified into one of the 47 categories based on the category center to which it is closest. Although S is very good for interactive work, it is relatively inefficient for processing large amounts of data because it is interpreted interactively. Based on our preliminary tests, we estimate that it would require 52 hours of CPU time to process the 512×512 pixels of an original image. Thus, we have chosen to offload the classification activity to a more efficient C program.

The C program (`classify`) reads in (a) the 15 images (**averages**) produced by Step 2, (b) the category names (`class.def`), and (c) the transformation (`transform`) and category

centers (`class.cent`) produced by Step 4. It classifies each pixel into one of the 47 categories by performing the required matrix multiplication (transforming the points into discriminant space) and determining the category based on the closest center algorithm. Three "images" are produced as the output of this classification analysis: (a) a 512×512 **class** "image" containing the values from 1 to 47, indicating the category into which the corresponding pixel was classified, (b) a 512×512 **superclass** "image" containing the values from 1 to 11, indicating which of the 11 supercategories the corresponding pixel falls, and (c) a 512×512 **certainty** "image" containing the values from 0 to 255, indicating the certainty of correct classification of the corresponding pixel.

Besides the three "images" that are its primary output, `classify` creates a number of output files to be used as utilities during display of images (see Step 6 below). The file `anot.class` contains a shell file for annotating the color legend of the **class** "image" by name of category. The `class.cmap` file is a shell file that is used (by `colorize`) to map pixel classifications into specific pseudocolor for the **class** "image". The files `anot.sclass` and `sclass.cmap` contain analogous information for the **superclass** "image".

For each supercategory, two files are produced to help select and display all the pixels classified into that specific supercategory: `zdi.`**supcat** and `zcm.`**supcat** (where **supcat** is the name of the specific supercategory). `zdi.`**supcat** is a shell file that selects only those pixels classified into the supercategory **supcat** and overlays them over the original image. `zcm.`**supcat** is a color mapping file that is used by `zdi.`**supcat** in the selection process.

Besides these files, which are used to display different aspects of the classification process, a file `center.dist` is produced which gives the distance from every category center to every other category center. This can be used to determine which categories are "close to" one another. This information can be used in a methodology for creating or deleting specific categories and selecting training set points to be placed in or removed from categories.

Step 6: Display

We have chosen to display the texture classification by mapping the pixel classification into color space. The DeAnza has three color channels: Red, Green and Blue. In order to visually superimpose the original image along with the texture classification, we have chosen to display the original image in Red and map the texture classification into different shades of combinations of Green and Blue. Then by temporarily suspending specific color channels, it is possible to quickly determine the accuracy of the texture classification by visually identifying those region which have and have not been classified correctly.

We have developed a C program (called `colorize`) which produces a pseudocolorization by mapping a specific intensity (or range of intensities) of an image into a 3-color display image. (In this application, the Red channel is always mapped to zero intensity.) Other peripheral software has been developed to annotate the pseudocolors with respect to their corresponding categories.

A overview of the dataflow involved in Steps 1 through 6 is shown in Figure 9.

4.2. Pictorial Results

In this subsection, we present pictorial results for classifying the two images **image.r** and **image.o**, using `btts.r` as the training set. Figure 10 displays the colorized **class** "image" after the computational scenario of the previous subsection is applied to **image.r**. Note the annotations along the sides of the image, giving a legend of the correspondence between the 47 category and pseudocolor. Figure 11 displays the colorized **superclass** "image" showing the classification into the 11 supercategories. It is not instructive to attempt to show the superposition of the original image (in the Red channel) in the photographic representation of this working paper. Only the physical manipulation of the DeAnza monitor makes this superposition technique useful. Figure 12 displays the **certainty** "image"; dark regions indicate

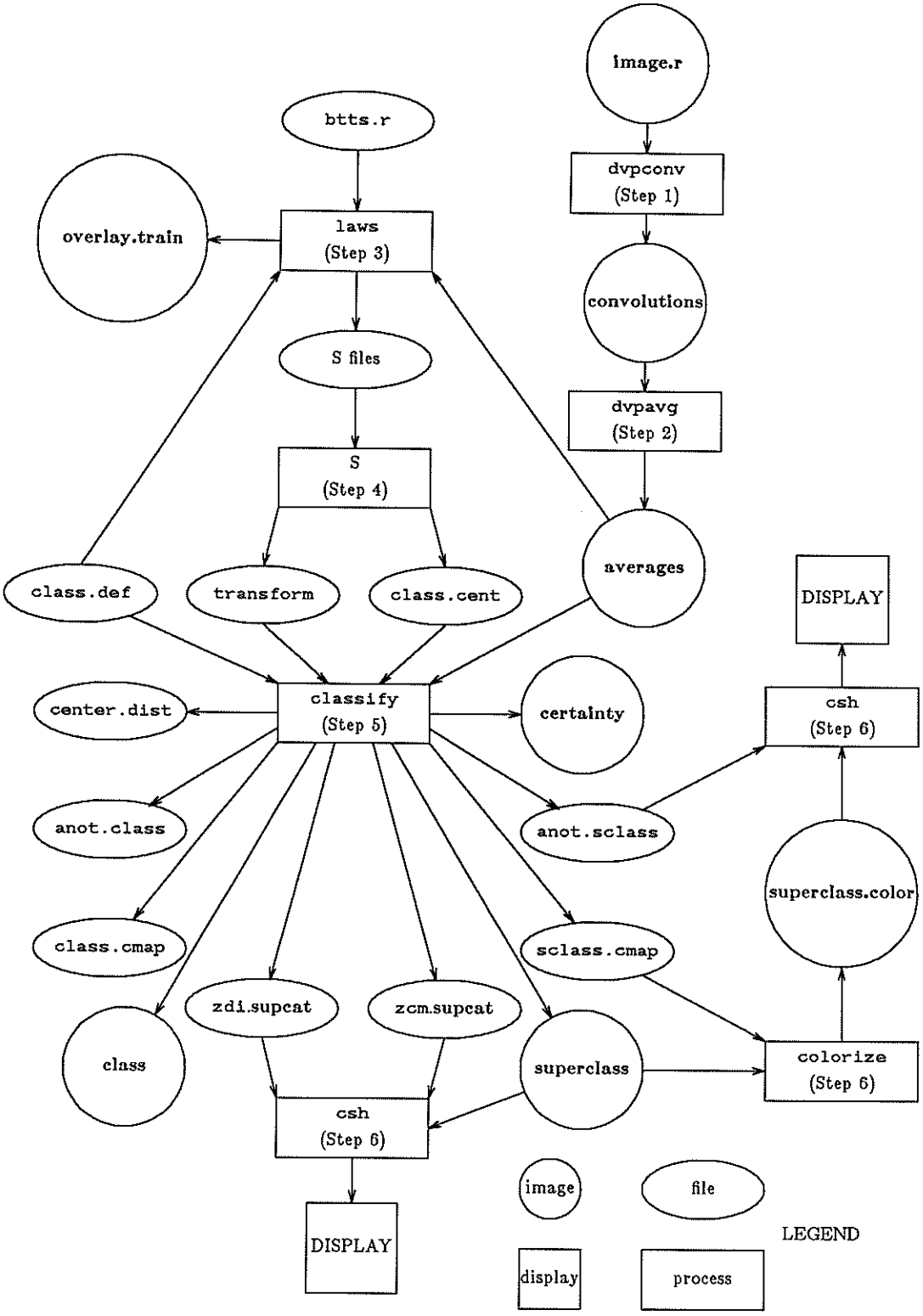

Figure 9: Dataflow For Steps 1 Through 6

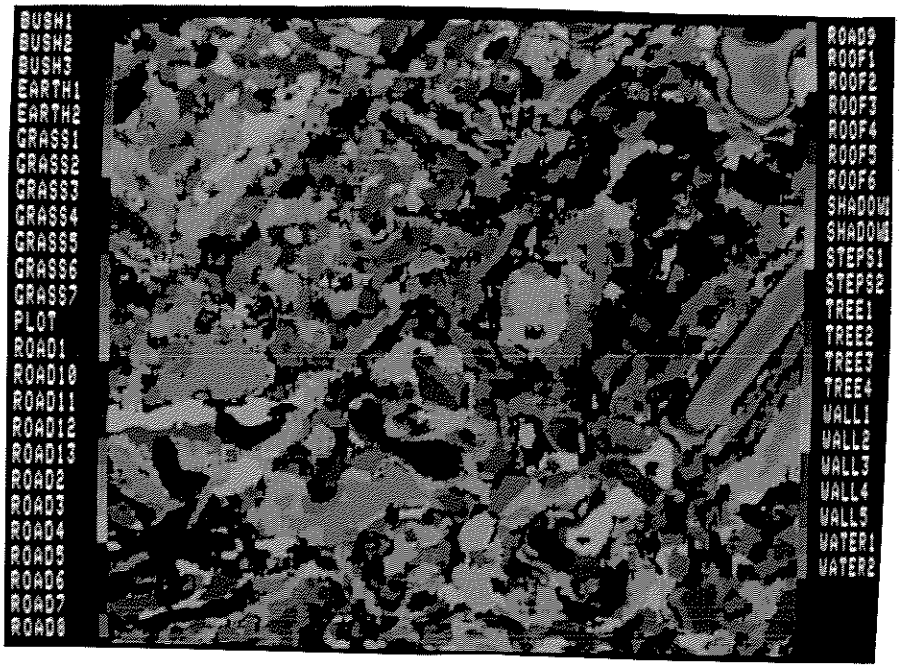

Figure 10: **class** Colorization Of **image.r** With 47 Categories

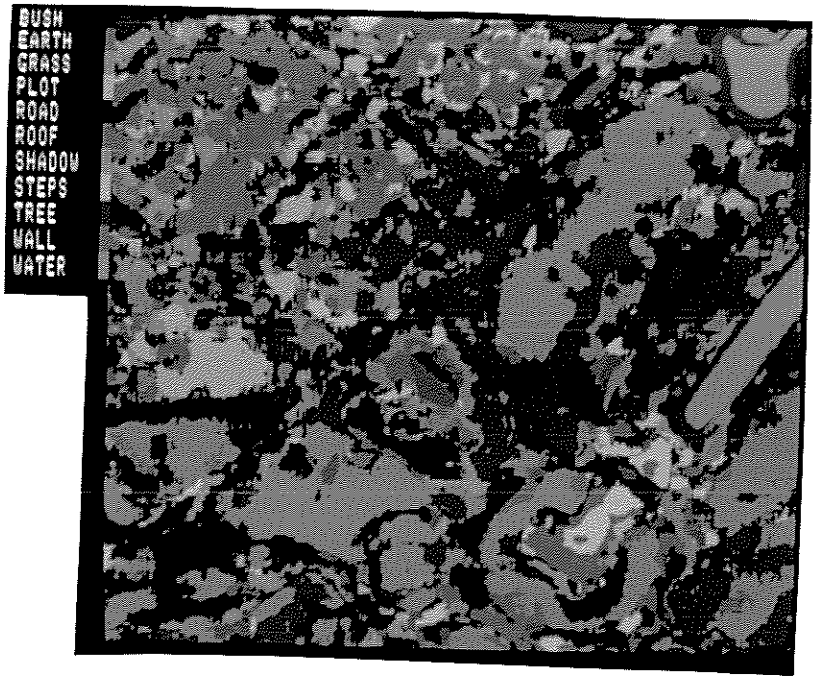

Figure 11: **superclass** Colorization Of **image.r** With 11 Supercategories

high certainty, and bright regions indicate low certainty (as a function of distance to the closest category center). Note certain regions of low certainty: boundaries around the water, and the center of the Lincoln Memorial. Given a display methodology such as this, it is relatively easy

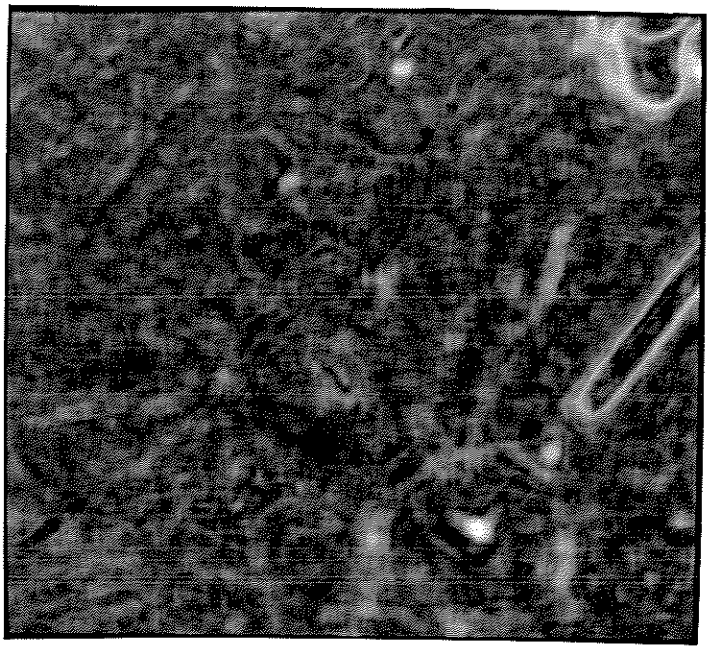

Figure 12: **certainty** Image For **image.r**

to quickly identify regions which have been misclassified. Once these regions are identified, new categories and training set points can be added and the classification process (only Steps 4-6) can be reapplied and further refinements made. Here, we have suggested a methodology involving visual inspection for identifying potentially misclassified regions. However, this is easily transformed into a more quantitative methodology, since the intensities in the certainty image are simply a scaling of the distance to the nearest category center. Clearly, a threshold can be supplied for which a specific classification is assumed to be invalid.

It is somewhat difficult to discriminate the different classifications based on the pseudocolor in Figure 11. In order to see the supercategories more clearly, four categories (water, tree, grass, and road) have been selected for a more binary display (using the `zdi` and `zcm` files). Figure 13 displays those pixels from **superclass** that were classified as water. Note that there are *no* pixels classified into this supercategory that do not correspond to water. However, there are pixels around the edge of the water bodies that were not classified into this supercategory (see figure 12). Figures 14 through 16 display the corresponding classification for

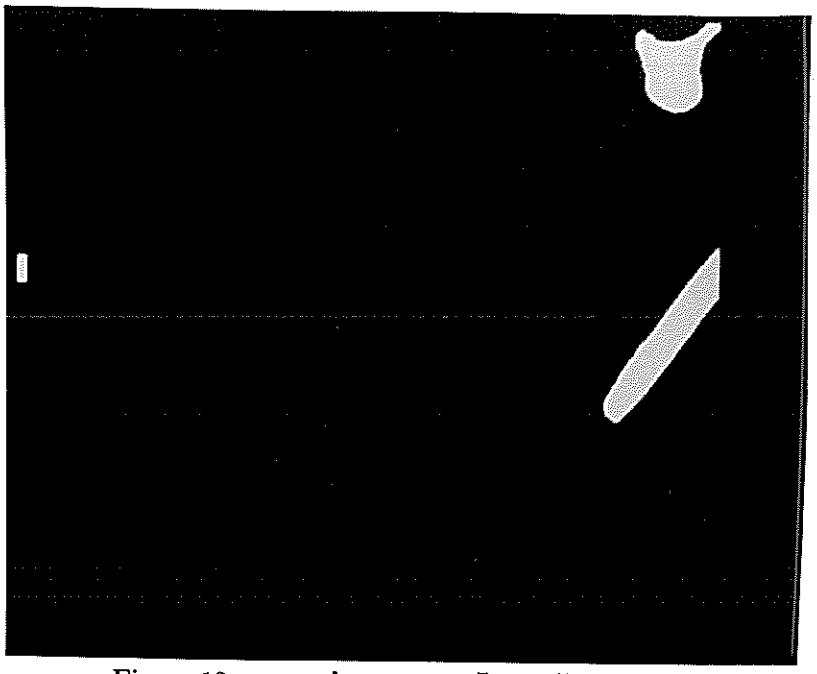

Figure 13: **superclass.water** Image For **image.r**

tree, grass and roof, respectively. Note that there *is* misclassification (in both directions) of

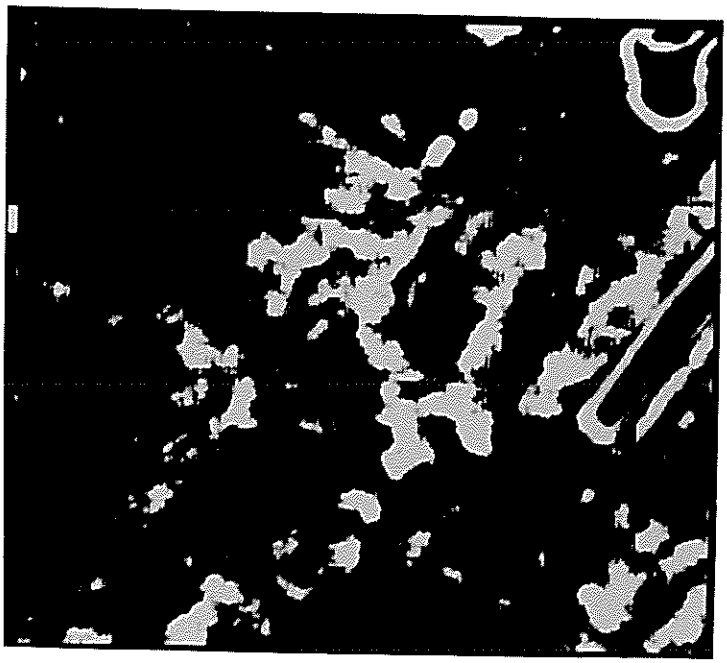

Figure 14: **superclass.tree** Image For **image.r**

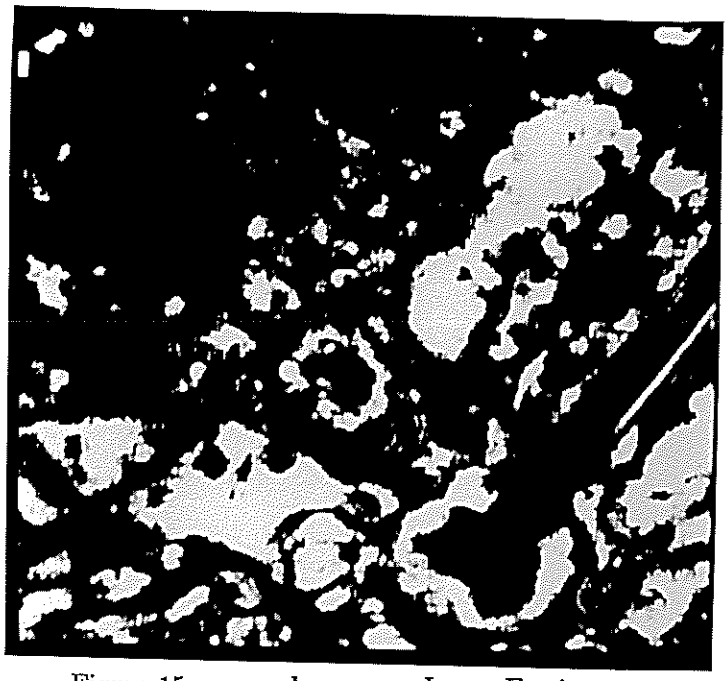

Figure 15: **superclass.grass** Image For **image.r**

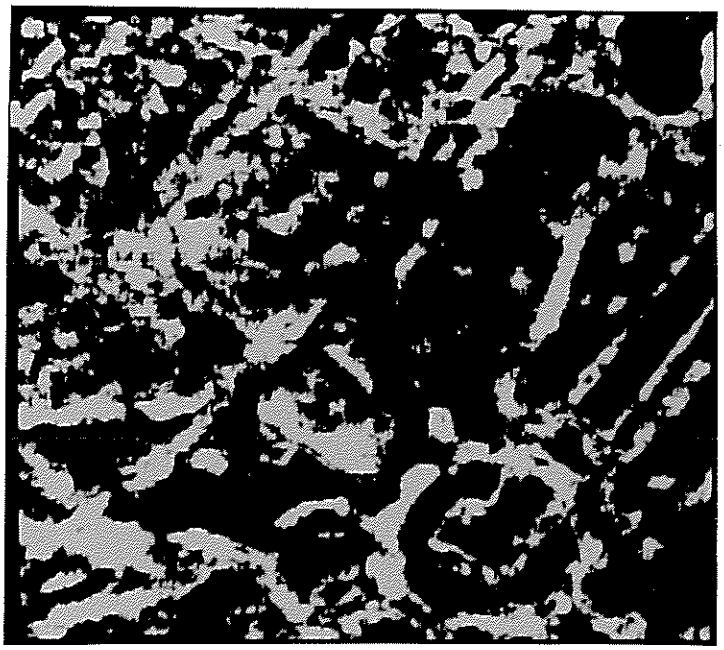

Figure 16: **superclass.road** Image For **image.r**

pixels for each of these supercategories. Trees and grass have a relatively high accuracy of classification. Road has a lower accuracy, but the basic abstract road structure can be seen in

Figure 16. Most of the misclassification in this category is caused by roofs being classified as roads.

The same classification analysis that was performed for **image.r** has also been performed for **image.o**. Note that only Steps 5 and 6 need be performed. Figure 17 is a display of the colorized **superclass** "image" for **image.o**. Figure 18 displays the corresponding **certainty** "image". Note again that the edges around the water have a high probability of being misclassified. Figures 19 through 22 display the binary classification for water, tree, grass, and road, respectively. These results are similar to those for **image.r**.

4.3. Space and Time Analysis

Each of the 15 images produced by Step 1 require 0.25 megabytes of disk storage, for a total of 3.75 megabytes. Each of the 15 images produced by Step 2 requires exactly the same

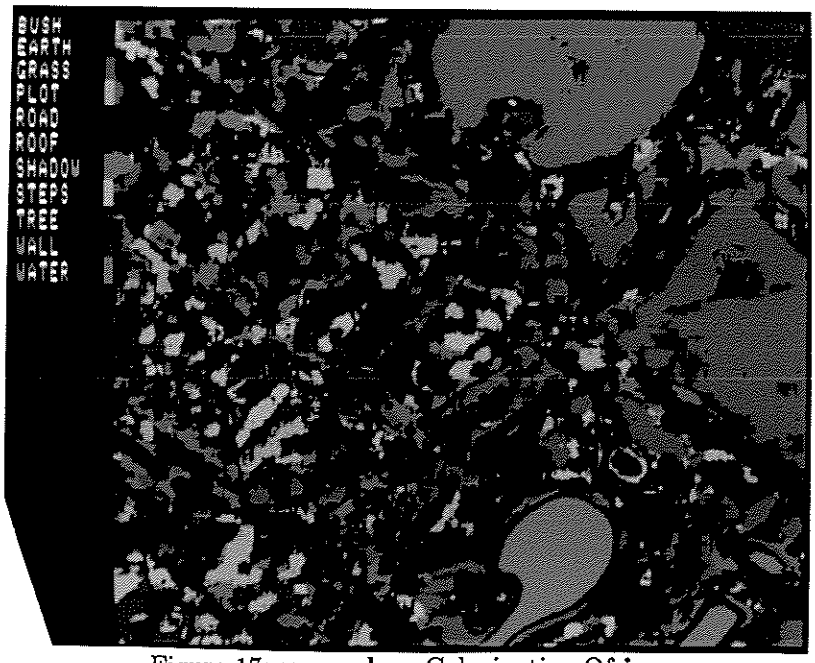

Figure 17: **superclass** Colorization Of **image.o**

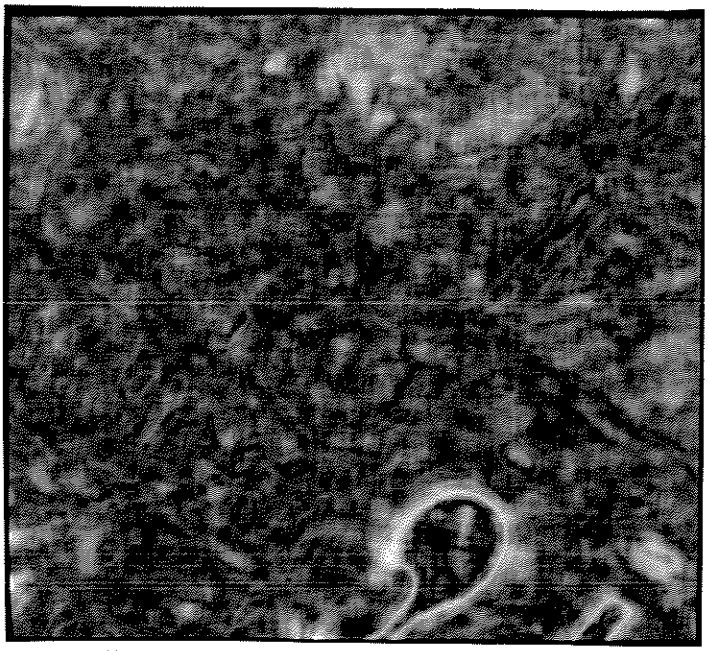

Figure 18: **certainty** Image For **image.o**

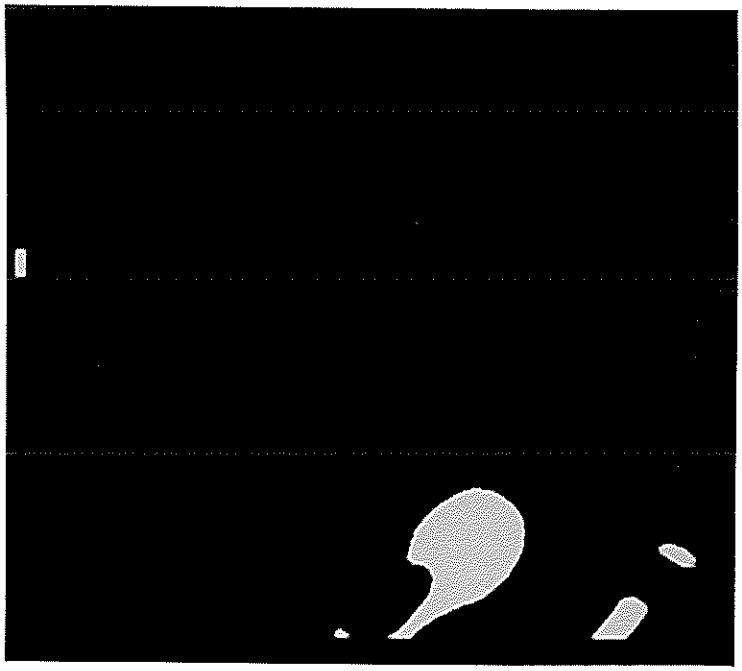

Figure 19: **superclass.water** Image For **image.o**

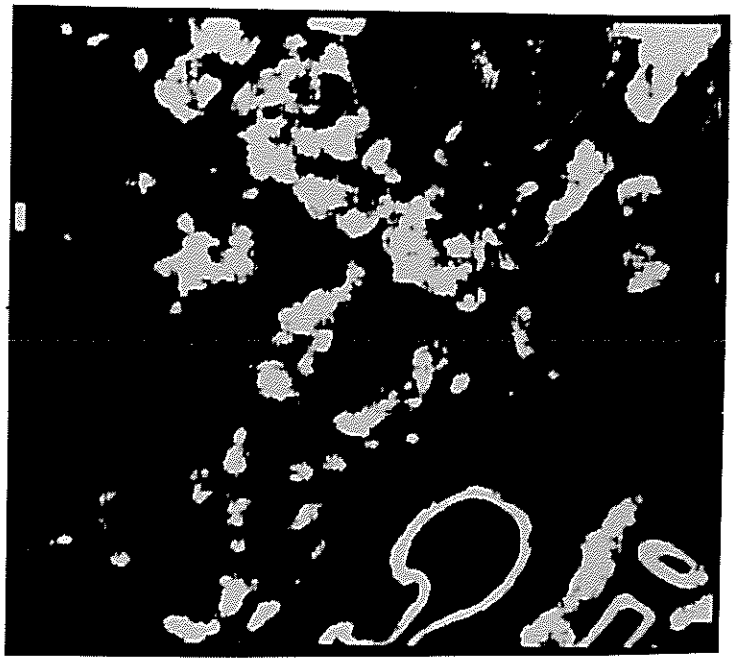

Figure 20: **superclass.tree** Image For **image.o**

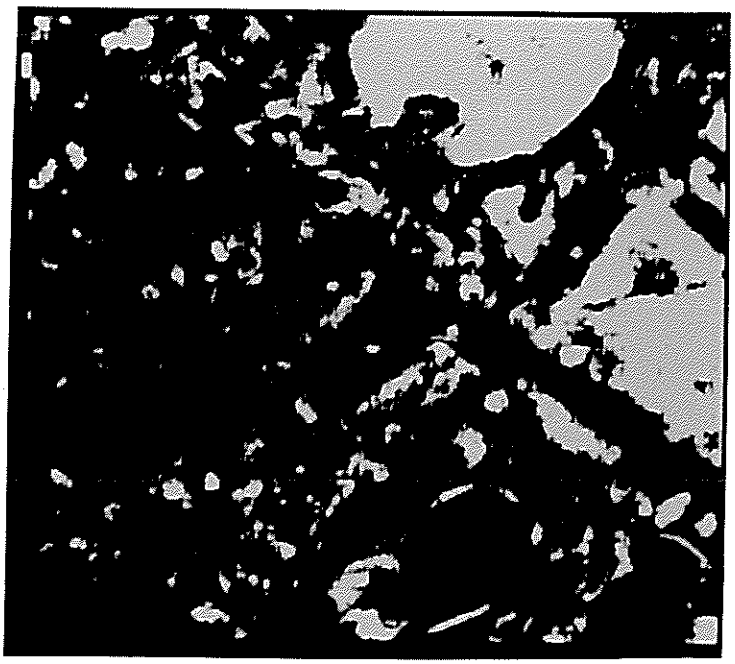

Figure 21: **superclass.grass** Image For **image.o**

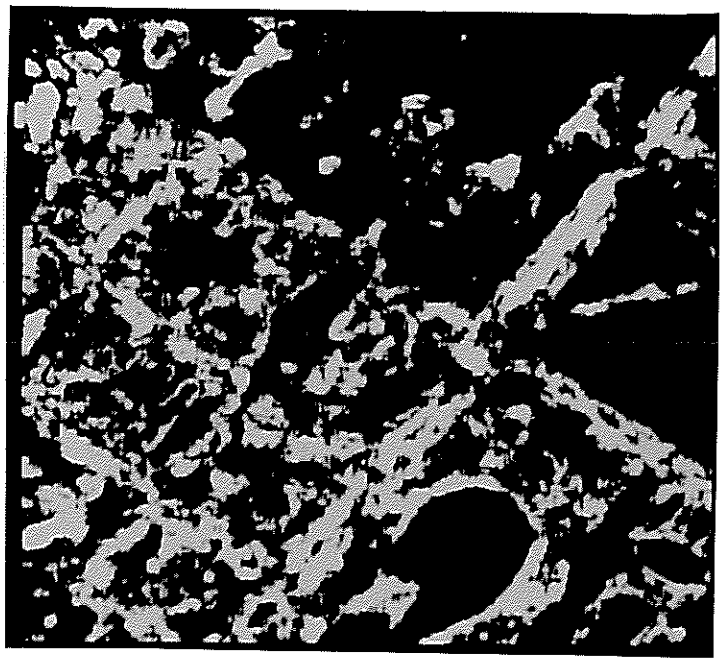

Figure 22: **superclass.road** Image For **image.o**

space; once these images are available, the 15 images produced by Step 1 can be removed (unless further averaging over a different macrowindow size is desired). The image **overlay.train** requires 0.25 megabytes of disk storage. Each of the **class**, **superclass** and **certainty** "images" require 0.25 megabytes of disk, and each of the two colorized display images requires 0.75 megabytes of disk (these are pixel interleaved images for all three color channels). Thus, during development and refinement of the classification processing, a maximum of 6.25 megabytes of disk storage is required for images. Once the category and training set selection is finalized, this can be reduced to 4.0 megabytes (eliminating the **class**, **certainty**, and **overlay.train** "images" and the colorized display images). Besides the disk space required for images, approximately 0.5 megabytes are required for source and object programs and other small files that communicate information between computational components.

In the following time analysis, both CPU and wall clock times will be given. These times will be given in the form XX:YY, where XX is the number of minutes and YY is the number of seconds (eg., 3:19 means 3 minutes and 19 seconds). These times represent execution on a

quiescent VAX/750 with 8 megabytes of primary memory with only one user on.

In order to understand the speedup possible by using special purpose hardware (in our case, the DVP of the DeAnza), we have performed the convolution computation (Step 1) on both the VAX and the DVP of the DeAnza. On the VAX, the total of all 15 convolutions took 73:08 CPU time (87:13 wall clock time). Breaking this CPU time down into user time and system time, 4171 seconds was spent in user time and 217 seconds was spent in system time; i.e., 95% of the time was spent in the primary algorithm, and only 5% of the time was spent in system support. On the DVP of the DeAnza, the corresponding computation took 6:28 CPU time (8:17 wall clock time). Breaking this CPU time down, 76 seconds was spent in user time and 313 seconds was spent in system time; i.e., only 20% of the total time was spent in the primary VAX algorithm, and 80% was spent in system support. The increased system time here (over the VAX-based convolution) is attributable to image movement between the VAX memory and the DeAnza memory and reformatting the images into PDS [6] format. The total CPU time here represents the VAX time required to (a) load images into the DeAnza memory, (b) extract images from the DeAnza memory (and reformat those images into PDS format on the VAX), and (c) generate and send instructions to the DVP of the DeAnza. Of the total CPU time here, 110 seconds was spent in reformatting raw images (on the VAX) into PDS format. While this is of practical interest to us, it is not conceptually necessary; the image is the same no matter what its format. If we subtract this 110 seconds from the DeAnza-based convolution time, we see a factor of 15 in speedup from the VAX-based convolution to the DeAnza-based convolution. In other experiments (specifically, a standard deviation over a 15×15 window), we have found speedup factors of close to 100.

The averaging over a 15×15 macrowindow (Step 2) was only performed on the DVP of the DeAnza. Execution time for all 15 images (combined) requires 6:58 CPU time (8:56 wall clock time). A significant speedup can be derived by combining Steps 1 and 2 directly on the DeAnza and not retrieving the intermediate results of Step 1 into the VAX memory. (Since the result of

the convolution is already in DeAnza memory, it can be immediately averaged over the 15×15 macrowindow without retrieving the intermediate result.) When Steps 1 and 2 are combined on the DeAnza, the execution time is 8:32 CPU time (11:12 wall clock time).

Collecting the texture space information for the 208 points of the training set (Step 3) takes in 0:39 CPU time (0:51 wall clock time). This is a very simple "gluing" algorithm. In fact, 20 of the 39 seconds of CPU time is spent in reading in the 15 images.

The discriminant analysis and center calculation in S (Step 4) requires 5:28 CPU time (7:55 wall clock time). This time could be significantly reduced by implementing a C program which performs the same function. However, this would require understanding all the details of the discriminant analysis, which is not the central issue here. Since the training sets are relatively small and the absolute time required to perform this analysis in S is small, we have chosen to retain flexibility by keeping this function inside S.

Classification of the entire 512×512 image **image.r** and writting the appropriate files (Step 5) requires 146:42 CPU time (165:21 wall clock time). This is the major computational component in the entire analysis. The two major algorithmic components are (a) multiplying a 15×2^{16} array by a 15×15 transformation, and (b) computing the distance between each of the 2^{16} 15-vectors (in discriminant space) and the 47 category centers.

Converting a classification "image" to a pseudocolor (`colorize`) image (Step 6) requires 0:17 CPU time (0:23 wall clock time). This must be done for both the **class** "image" and the **superclass** "image". It is also done while producing the binary display for each of the separate supercategories in **superclass**.

4.4. Conclusions

A relatively good job was done in classifying these two images. For **image.r**, we estimate the accuracy to be in the 75-85% range; for **image.o**, we estimate the accuracy to be in the 70-80% range. These estimates are based on visual inspection. A more quantitative analysis would require some form of hand segmentation of the images; this has not been done.

During the development of this system, we have developed a methodology for refining the classification. The file `center.dist` can be used to identify categories whose centers are close in discriminant space, and therefore may be "too close" to one another to be able to be discriminated. Specifically, one or more categories within a supercategory may be "close" and the user can combine them. If categories in different supercategories are "close", the user may change some of the points in the training set to attempt to separate the categories more. In any event, the user is made aware of what categories may be misclassified. This type of analysis can be applied prior to the actual classification or display of any specific image.

The display portion of the methodology allows the user to overlay classifications (either as a whole or separately) over the original image to quickly identify regions of the image which may have been misclassified. The **certainty** image can also be overlaid on either the original image or the colorized classification image to identify potentially misclassified portions of the image. Once misclassified regions of the image have been identified, the user can modify the training set by adding (or deleting) categories and/or training set points.

In the case of these two images, we immediately find that the regions around water bodies are misclassified. The user can create a new category, say water-edge, select training set points (for instance, using the data in the `etts.r` file), place water-edge in the supercategory of water, and attempt another classification (using Steps 3 through 6).

Besides the direct texture classification processing described in this working paper, postprocessing may be applied to increase accuracy. Specifically, the processing presented here performs a pixel-by-pixel classification of an image, with *no* abstract concepts (such as continuity) involved. However, we know that certain physical phenomenon cannot occur in discontinuous ways. For example, we may choose to apply some postprocessing which eliminates small patches of isolated texture fields when surrounded by large massive dominating texture fields, replacing the classification of the isolated pixels with the dominant texture in the region. Many other postprocessing techniques may also be applicable.

The major computational component in this system is the classification process (`classify`) itself. The two major algorithmic components here are the transformation into discriminant space (performed by a large matrix multiplication) and a search for the closest category center. This could be significantly speeded up if these two components can be "moved" to the DVP of the DeAnza.

By visual inspection of the classified images, we find that those categories which have "wide" texture fields in the images (such as water, grass, and tree) have a relatively high accuracy of correct classification, and those categories that have "narrow" texture fields in the images (such as roof, shadow, and road) are classified somewhat less accurately. This is not a surprising result. We suspect that accuracy will increase in images with "wider" texture fields.

5. Summary

In general, Laws' TEM seems to be a viable method for texture classification. A relatively high accuracy of correct texture classification can be obtained across images with similar photographic properties. Accuracy across images with dissimilar photographic properties is low enough that it is probably inappropriate to attempt such classifications. Methodologies for determining "good" training images and "good" training sets can be developed. Such

methodologies may incorporate human visual inspection of displays and quantitative automated algorithms.

---

1. Kenneth Ivan Laws, *Textured Image Segmentation,* University of Southern California (January 1980). Ph. D. Dissertation.

2. Will D. Gillett, *An Evaluation of Two Texture Classification Methods,* Dept. of Computer Science, Washington University (October 1986). Working Paper WUCS-CV-86-2.

3. Sing-Tze Bow, *Pattern Recognition,* Marcel Dekker, Inc., New York (1984).

4. Bob Bickler, *IP-8500 Image Array Processor Hardware Reference Manual,* Gould, Inc. (September, 1983).

5. Richard A. Becker and John M. Chambers, *S: An Interactive Environment for Data Analysis and Graphics,* Wadsworth, Belmont, CA (1984).

6. A. P. Reeves, *PPS-PDS Users' Manual, Version 2,* School of Electrical Engineering, Cornell University, Ithica, NY (1983).