# Multiscale Relaxation Labeling of Fractal Images

Jeffrey A. Choate *(jachoate@eng.xyplex.com)*
Michael A. Gennert *(michaelg@cs.wpi.edu)*

Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

Category: Segmentation and perceptual grouping

**Abstract**

This paper describes the application of multi-scale relaxation to automatically detect pavement distress. Pavement distress detection is a difficult task which simple edge detection schemes perform poorly. We have chosen to use relaxation labeling to improve upon an initial edge-based segmentation. This work is based upon a fractal model of pavement distress. The scale-invariance property of fractals suggests that information at different scales of resolution may be combined to improve segmentation. Thus, we have developed a multi-scale relaxation technique for use in a pavement distress detection system. Straightforward linear interactions fail to capture the complexity of pixel interactions for this problem. To better model pixel interactions, we have included non-linear terms in the relaxation process. Symmetry arguments and careful engineering allow a 93% reduction in the complexity of this approach. To demonstrate the necessity of the multi-scale approach, examples with and without multi-scale relaxation are shown. We found that performance was greatly improved by multi-scale relaxation.

# 1 Introduction

This paper describes the application of multi-scale relaxation to automatically detect pavement distress. Pavement distress detection is a difficult task for several reasons:

- There is great variability in the appearance of both sound and distressed pavements.
- It is necessary to detect features as small as 2 pixels wide.
- There is typically much sensor noise present in pavement imagery.
- Traditional computer vision models fail to capture the complexity of pavement distress.

Because of these problems, simple edge detection schemes perform poorly.

We have chosen to use relaxation labeling to improve upon an initial edge-based segmentation. This requires some model of the appearence of pavement distress. Our approach is to model pavement distress as fractal structures. The scale-invariance property of fractals suggests that information at different scales of resolution may be combined to improve segmentation. Thus, we have developed a multi-scale relaxation technique for use in a pavement distress detection system. The system is trained on images whose labels are known; relaxation coefficients are estimated using a least-suqares optimization.

To demonstrate the necessity of the multi-scale approach, a version of the system was implemented that used only a single level of processing. It was found that performance was greatly degraded. Examples with and without multi-scale relaxation are given later in this paper.

An added complication is that straightforward linear interactions fail to capture the complexity of pixel interactions for this problem. To better model pixel interactions, we have included non-linear effects that require 819 coefficients. Fortunately, symmetry arguments and careful engineering reduce the number to 61.

## 1.1 Detailed Problem Description

Distress (cracking) occurs on paved surfaces for a variety of reasons. Since there are no generic sizes or shapes, cracks are particularly hard to detect. The images used in this image have been produced by the PASCO system [SOL91]. Images generated using this system have a resolution of .75mm/pixel [ELK91a]. The intensity ranges of the different regions in the image are often very close together, and distress pixels can be brighter, darker, or the same as the aggregate pixels [ELK90]. In order to successfully model the distress, another property of the distress can be used.

Cracks have been shown to be fractal in [GEN91]. This is a useful property of distress which can be exploited to differentiate the distress from the rest of the features in the image. A fractal is defined as a set of points whose fractional dimension is strictly larger than its topological dimension [MAN82]. An interesting property of fractals which can be exploited is self-similarity. A fractal object will appear to have the same fractal dimension, regardless of the scale that is used or the distance from the object [GLE87]. Since a crack is a fractal curve, it will look the same regardless of the scale at which the it is viewed.

Not all of the data in pavement images are of use for automatic distress detection. Pixels which correspond to distress larger than 1.5mm within

pavement images comprise only 0.1% of the image generated from a lane-km of pavement [ELK91a]. In order to detect such a small component of the image, the system must exploit the properties of the distress [ELK91b].

Standard image processing techniques applied to pavement inspection have only met with limited success [CHI83, CUR84, ELK91b, GOS91]. Highly non-uniform paving material comprises most of the distress image, and boundaries between these materials must be eliminated for successful detection of distress. This requires the use of context in image processing, which we provide through relaxation.

# 2 Relaxation

The main goal of this research is to derive a consistent set of labelings given an image which may or may not contain pavement distress. Data input to the system will not contain consistent labelings of the image but rather rough initial guesses. Consistently labeled images contain continuous distress regions, surrounded completely by boundaries which separate the crack from the aggregate. The initial labelings may be inconsistent. Relaxation is used to iteratively update the labels in an image, based upon labels at neighboring points in order to arrive at a more consistent set of labels [HUM83]. In this work, the labels are treated as probabilities, which imposes additional constraints on the label set: the labels must be non-negative and sum to one.

## 2.1 Least Squares Approximation

A least squares approximation is used as the update rule in the relaxation algorithm. The approximation is used to find the best fit to an ideal set of data. When the system is "trained", a set of data with known labels is used

to find coefficients which represent the best fit for the relations between the labels in the data. Neighbor relations, or influences which are inconsistent will have negative coefficients, relationships which have little influence will have small weights, and relations which are important have large coefficients. These coefficients are used as weights when performing the labeling of the input images.

A least squares approximation is well suited to this task for several reasons. First, a least squares approximation is used in order to find the relationships between variables, and to find the best fit to some data. This best fit approximation can be applied to an ideal set of data which represents the labeling that the system should derive. This solution to the best fit can then be applied to the un-labeled images by imposing the same fit onto the input data. In this system, we have a data point $p_x$, with a given label $i \in \{A, B, C\}$, and a collection of neighbors for that pixel, each with a separate label set. The label sets for both $p_x$ and the neighboring pixels are defined to be probabilities. Since we would like to find the relationship between $p_{xi}$ and the neighbors in the immediate neighborhood of $p_{xi}$, and then apply this relationship to later data sets where the correct labeling is not known, a least squares approximation seems to be an ideal mathematical tool to use.

Consider a model of a system with $k$ data points, $N$ neighboring influences, and $L$ possible labels for each data point. A data point $p_x$, has several neighbors which exert an influence on the current data point. $p_{n_x^{ij}}$ is the coefficient for the influence on $p_x$ from label $j$ of $p_x$'s $i^{\text{th}}$ neighbor. Using this notation, the coordinate vector equations can be rewritten as:

$$\vec{y} = \begin{bmatrix} p_{00} & p_{0C} & p_{0C} \\ \vdots & \ddots & \vdots \\ p_{k0} & p_{0C} & p_{kC} \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} p_{n_0^0 A} & p_{n_0^0 B} & p_{n_0^1 A} & \cdots & p_{n_0^N B} & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & 1 \\ p_{n_k^0 A} & p_{n_k^0 B} & p_{n_k^1 A} & \cdots & p_{n_k^N B} & 1 \end{bmatrix} \quad \vec{x_0} = \begin{bmatrix} p_{n_x^0} p_{xA} & p_{n_x^0} p_{xB} & p_{n_x^0} p_{\cdot} \\ \vdots & \ddots & \vdots \\ p_{n_x^N} p_{xA} & p_{n_x^N} p_{xB} & p_{n_x^N} p_{\cdot} \end{bmatrix} \tag{1}$$

We can then write the least squares approximation as a vector equation:

$$\mathbf{E} = \|\vec{y} - \mathbf{A}\vec{x_0}\|^2 \tag{2}$$

We assume that $\mathbf{A}^t\mathbf{A}$ is a linearly independent matrix, so the rank of $\mathbf{A}^t\mathbf{A}$ will be equivalent to the rank of $A$, or $NL$. If this condition is satisfied, then the matrix $\mathbf{A}^t\mathbf{A}$ is invertible, and so we can solve for $\vec{x_0}$ using the following equation:

$$\vec{x_0} = \left(\mathbf{A}^t\mathbf{A}\right)^{-1}\mathbf{A}^t\vec{y} \tag{3}$$

# 3 System Overview

Each pixel in the distress image can be labeled as either a crack pixel, a boundary pixel, or an aggregate pixel. Weightings are assigned to each of these labels, and are used to classify each pixel. A pixel will have a collection of these weightings, one for each possible label; this collection of weightings is referred to as the point's label set. Relaxation algorithms adjust these labels to make a pixel's label set more consistent in the context of the current neighborhood. In this work, the label set is further constrained by treating each label as a probability. A label's weighting therefore, is actually the probability that the data point is correctly labeled with the respective class.

Initially, zero crossings of a second directional derivative edge detector [TOR84, GOS91] are used to detect all of the boundaries in the image. After these boundaries have been detected, initial crack and aggregate regions are set up in the image using simple intensity value comparisons.

The self-similarity property of fractal distress curves can be exploited to aid in the labeling process. One way to exploit this property is to use multiple resolutions of the input data. Data at any given resolution will have neighboring pixels at the same resolution as well as neighbors at higher and lower resolutions. These scales are constructed by averaging the input data to generate a coarser resolution. Because the distress retains the same properties at all scales, relaxation can be performed at each scale using the same set of rules.

In order to perform relaxation labeling, a set of relaxation coefficients must be found. Therefore, a two phase process is defined. The first (training) phase produces the coefficient vector which is used by the relaxation algorithm in the second phase. The second (relaxation) phase uses the coefficient vector and input data to generate new labelings of the image. These updated labelings are then re-input to the system for further updates, until the system converges.

# 4 Algorithm Design

The goal of this work is to consistently label the distress (cracks) in input images. A model of correct pavement labeling must be defined for the system to use. This model defines the possible classifications, or labels for a data point, as well as the neighbors which will influence a data point.

Data points labeled as Crack are data points which lie directly over crack pixels in the input image. Data points labeled as Boundary are the points which lie on the immediate edges of Crack data points. Boundary points are also defined as points which have both Crack and Aggregate points as neighbors. Finally, data points labeled as Aggregate are any points which are neither Crack nor Boundary points.

Continuous labeling was chosen as the relaxation method, and an additional constraint was placed on the label set which makes each label in a data points label set a probability. Treating the label set as a probability offers added benefits over standard continuous relaxation labeling, since a data point will have a probability of belonging to a given class in the label set, rather than just a continuous label weighting. This allows classifications to be easily made by simply using the greatest probability in the label set as the current point's classification.

A data point will have a number of neighbors both at the same scale and at multiple scales which will each assert a different influence on the current data point's label set. The neighborhood that is considered is defined using the following model. A data point has eight neighbors on the same level as itself, one neighbor on the level above the current level, and four neighbors on the level below, for a total of thirteen neighbors (Figure 1).

## 4.1 Edge Detection

Detected edges are used to generate the initial set of data used by the relaxation algorithm. The zero crossings of the second directional derivative method [HAR84, TOR84] was chosen because it guarantees closed contours, keeps good localization of edges, and responds well to actual edges. The

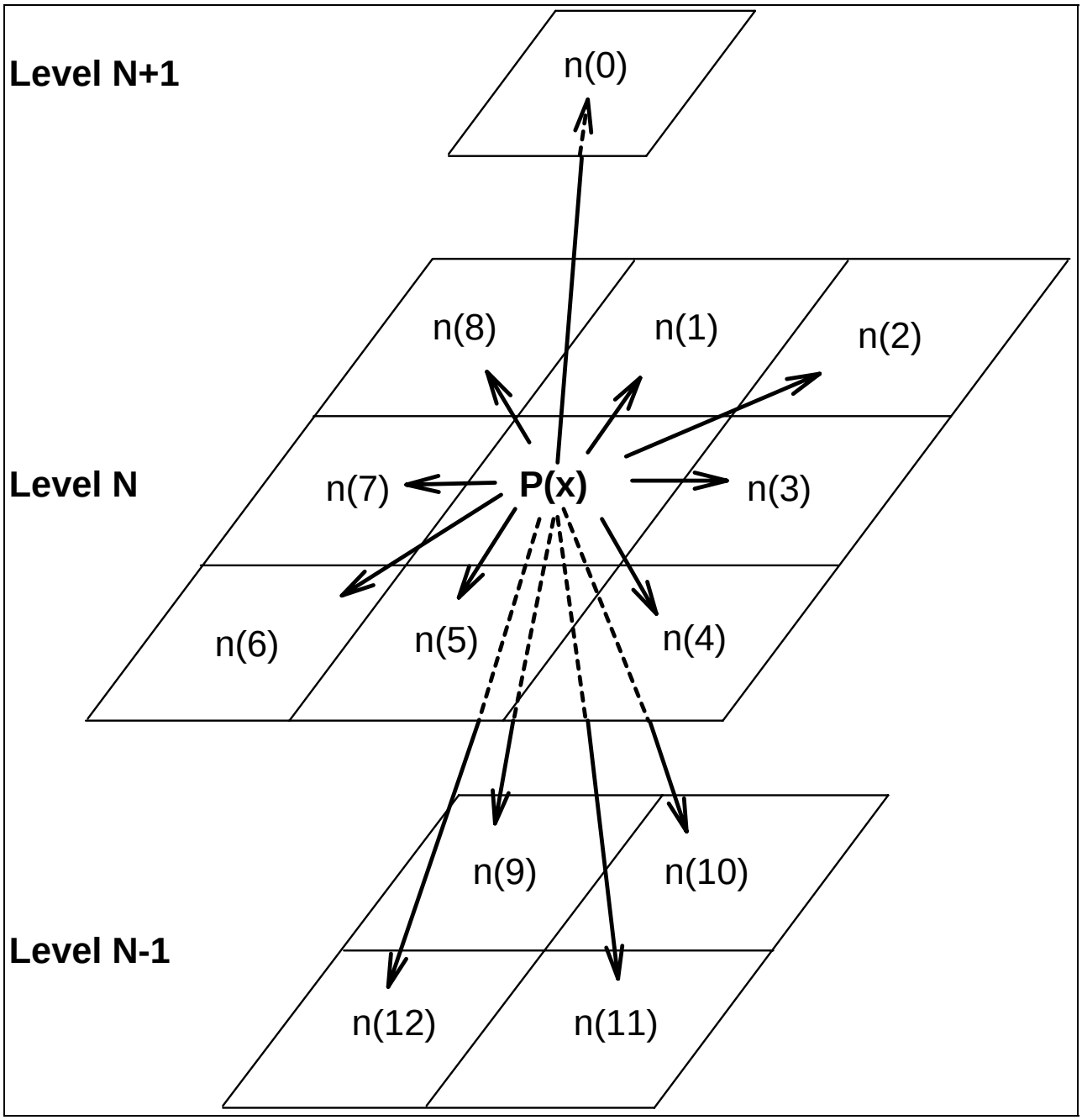


Figure 1: Neighbor Label Sets for Data Point X

magnitude of the gradient, $\|\nabla I\|$ is used to generate a weighted boundary image. This weighted boundary image is then normalized into the range of $[0 \cdots 1]$, so that edge strength is replaced by a probability.

## 4.2 Terms in Relaxation

We consider an image to be a set of data points, $p_i$. Every data point in the data set will have possible classes or labels. If there exist $L$ possible labels, then $p_{xj}$ represents one label $j$, for data point $p_x$. The $i^{\text{th}}$ neighboring pixel for $p_x$, represented as $p_{n_x^i}$ will have associated confidences for each label, $j$ in its label set. An additional constraint is placed upon $p_x$'s label set, allowing us to consider each label in the label set as a probability.

$$\sum_{j=0}^{L} p_{xj} = 1 \tag{4}$$

The goal of relaxation is to adjust the label set for each $p_x$ in the data set, based upon its current neighbors. Therefore, each pixel has $N$ neighbors, each with $L$ possible labels; there will be $NL$ influences for each label in $p_x$'s label set, with exactly one influence for each label in $P_{n_x^i}$'s label set.

One of the initial requirements of the relaxation algorithm is a constraint relation between the labels at neighboring data points and the labeling at the current data point. This constraint relationship is used as an update rule in order to adjust the label sets for each data point. In this project, a least sum of squares approximation is used to find coefficients which represent an ideally labeled image. Using the equation for the least squares fit, (Equation 5) the update rule minimizes the distance between the current label set and the best fit of the ideal data.

$$\mathbf{E} = \|\vec{y} - \mathbf{A}\vec{x_0}\|^2 \tag{5}$$

Standard relaxation algorithms work by considering the influences of neighboring points upon a given data point. These neighbors can consist of the immediate neighbors in a four-connected or eight-connected scheme, where points which are not "touching" the current point are not considered. Neighbors could also consist of pixels which surround the current pixel for some distance around the current pixel. In this work, additional considerations are needed since a neighborhood does not just consist of points which lie around a data point, but also neighbors which lie on the image at scales both above, and below the current scale at higher and lower resolutions respectively. Therefore a third dimension is added to the influences on a current data point. The possible neighbors that a data point can have are the eight data points immediately surrounding the current data point on the current level, the data point on the level above which the current data point influences, and the data points on the lower level which the current data point is comprised of. In this implementation, four data points on a lower level comprise one data point on the level above, so a data point has four neighbors on the level below the current level. In total, a data point at a given level has thirteen neighbors, as shown in Figure 1.

Let us assume a linear relationship between the current data point and each of its neighbors. Given thirteen neighbors, with three labels each, thirty-nine terms will enter the equation; these terms are described as the linear influences upon the current data point. Each neighbor effects each $p_{xj}$ differently, and therefore separate weighting sets must be used for each class of

$p_x$.

Linear influences do not provide enough information for suitable relaxation, since accurate classification is not possible using just information about any one surrounding neighbor. Most situations require more than one neighbor's influences for accurate classification. Consider Figure 2:

| | | |
|---|---|---|
| B | B | B |
| C | ? | C |
| B | B | B |

Figure 2: An Obvious Labeling

It should be clear that the best labeling for the unknown data point is Crack, using information on both sides of the unknown data point. Information about any single neighboring data point is not sufficient to determine the correct label of an unknown data point. Therefore,the interactions between *groups* of neighbors, called non-linear interactions, must be incorporated for the correct labeling of the system.

*Pair-wise* interactions are the influences that two neighbors in combination will have on the current data point. A pair-wise interaction, $P_{n_x^i k} P_{n_x^j l}$ is the relationship on the current point from label $i$ of the $k^{\text{th}}$ neighbor and label $j$ of the $l^{\text{th}}$ neighbor. As one might guess, there are substantially more influences when considering the pairwise influences, since we are combining the labels of neighbors. In the same system considered above, the number of pair-wise interactions are: $\binom{NL}{2} = \frac{(NL)!}{2(NL-2)!}$. These interactions do not include *square terms*. Square terms are terms which are considered as pair-wise interactions with themselves. If square terms were included in pair-wise

interactions, there would be an additional $NL$ terms. However, technical complications (matrix non-invertibility) make training the square terms impossible.

Higher-order interactions could be considered; In this work, only linear and pair-wise terms are considered. Therefore, including the $NL$ linear terms, and the $NL$ square terms, the total number of influences upon a single data point is given by:

$$2(NL) + \frac{(NL)!}{2(NL-2)!} = \frac{NL(NL+3)}{2} \tag{6}$$

## 4.3 Multiple Levels

The use of multiple scales offers several improvements over normal single-scale relaxation algorithms which do not incorporate information from several levels. Higher scales (coarser resolution) allow the system to re-connect discontinuous regions. At some given scale, two disconnected regions will both be represented by a single *parental data point*. Through the use of the relaxation algorithm, this information will eventually propagate down to the level where the two disconnected regions lie. This information will allow the regions to be reconnected based upon that information from the higher scale. Higher scales also remove noise and other non-relevant information.

## 4.4 Relaxation Updates

In order to update a data point, a coefficient matrix $\mathbf{A}$ must be supplied and a new vector $\vec{y}$ must be computed. The vector $\vec{y}$ is the current label set for the data point $p_x$. The matrix $\mathbf{A}$ represents the influence matrix for the neighbors of $p_x$. In a system with $L$ possible labels in a label set, and $N$ possible neighbors for a given data point $p_x$, the vectors are represented as:

$$\vec{y} = \begin{bmatrix} p_{x0} & \cdots & p_{xL} \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} p_{n_x^0 0} & p_{n_x^0 1} & \cdots & p_{n_x^N L} & 1 \end{bmatrix} \qquad (7)$$

To perform the relaxation labeling, we form the scalar product of the current neighbor influences, $\mathbf{A}$ by the coefficient vector, $\vec{x_0}$. In a linear system, this scalar product is a linear mapping of the current label set to a more consistent label set. The labels in the set will still sum to one, but the values can lie outside of the range between $[0, 1]$. This creates a need for a mapping function to map the values of the label set back to a probabalistic domain. In a non-linear system, values in the new label set are not necessarily mapped to points still within the probabilistic plane. Therefore, the new label set must be mapped back into probabilities.

## 4.5 Relaxation at Multiple Scales

Relaxation commences at the lowest scale and proceeds in the following manner:

1. Perform relaxation labeling at current scale, utilizing data at current scale, and scales above and below where appropriate.

2. Compute error measure using least squares approximation for the entire level.

3. Add level error measure to system error measure

4. Move to next level in the current direction of traversal

5. Repeat procedure until either at top or bottom level

6. If system error measure is below some threshold, stop

7. Start at current level, reversing the level traversal direction

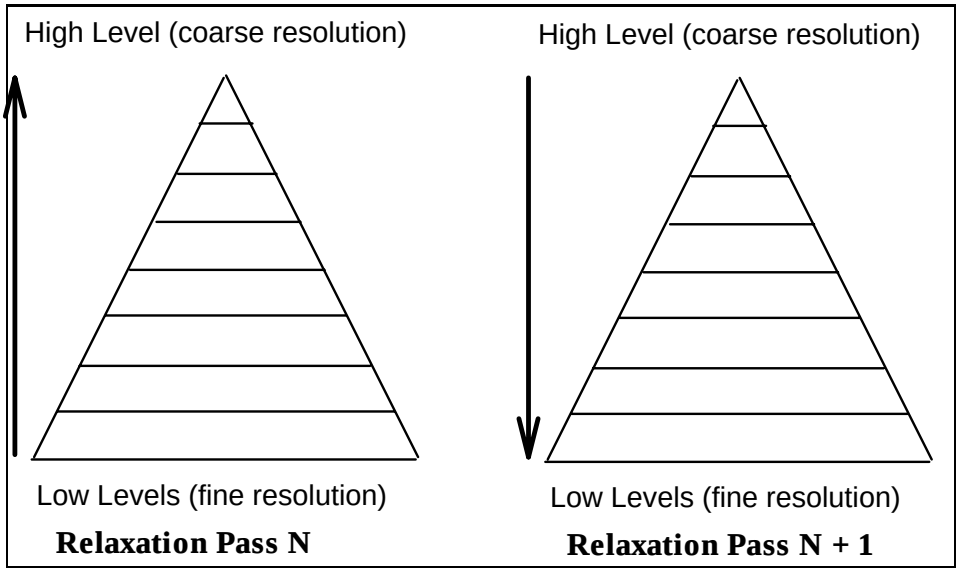


Figure 3: Traversal of Levels at Successive Iterations

This algorithm provides a hierarchical structure which allows the updated information to be passed from the lower levels upwards, and from the higher levels downwards as illustrated in Figure 3.

## 4.6 Linear and Non-linear Terms in the Relaxation

A simplistic model which considers only the direct linear influences upon the current point was implemented originally. We found this simple model to be inadequate, but recount it here for explanatory purposes. A given data point, $p_x$, is defined to have 13 neighbors which exert influences upon the current point's label set to bring it closer to the best fit to the ideal data. If just linear influences are considered, there will be $13 \times 3$, or 39 possible influences on $p_{xj}$. Since there are 3 labels in the $p_x$'s label set, we would have to solve for $\vec{x_0}$ which would be a $39 \times 3$ matrix. This matrix can be simplified

using arguments of symmetry and the probabilistic nature of the label sets, reducing the number of terms.

Consider the matrix of neighboring influences, $\mathbf{A}$, which is used to solve for $\vec{x_0}$, where $p_{n_x^i}$ represents the $i^{\text{th}}$ neighbor of $p_x$, and $p_{n_x^i A}$ represents the probability of $p_{n_x^i}$ being labeled as Aggregate, and likewise $p_{n_x^i B}$, $p_{n_x^i C}$ represents the neighbor's probability of being labeled as Boundary and Crack, respectively.

$$\mathbf{A} = \begin{bmatrix} p_{n_0^0 A} & p_{n_0^0 B} & p_{n_0^0 C} & \cdots & p_{n_0^{12} C} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ p_{n_k^0 A} & p_{n_k^0 B} & p_{n_k^0 C} & \cdots & p_{n_k^{12} C} \end{bmatrix} \tag{8}$$

Since $C = 1 - (A + B)$, we can substitute this eliminate all occurrences of $C$. After performing the multiplications in Equation 3 to solve for $\vec{x_0}$, and combining like terms, the crack terms will reduce to a single one in the last column of the matrix. Therefore, the new matrix, $\mathbf{A}$ is written as:

$$\mathbf{A} = \begin{bmatrix} p_{n_0^0 A} & p_{n_0^0 B} & p_{n_0^1 A} & \cdots & p_{n_0^{12} B} & 1 \\ \vdots & \vdots & \vdots & \cdots & \vdots & 1 \\ p_{n_k^0 A} & p_{n_k^0 B} & p_{n_k^1 A} & \cdots & p_{n_k^{12} B} & 1 \end{bmatrix} \tag{9}$$

Refering to figure 1, note that neighbors $n_7$ and $n_3$ should have the same influence, since they are symmetric about the current data point. Extending this a bit, since $\{n_1, n_3, n_5, n_7\}$ are all symmetric about $p_x$, they all should have the same influence for the same label in each of their respective label sets. Therefore, these four terms with the same label can be grouped together to form a single influence. Likewise, the eight connected diagonal neighbors, $\{n_2, n_4, n_6, n_8\}$ should be grouped together since they are also symmetric. These symmetry arguments can also be applied to group the lower level pixels, $\{n_9, n_{10}, n_{11}, n_{12}\}$ together. After combining similar terms, there will

be four influences from each label. With 2 labels, the 26 total terms are simplified to 8 groups of terms. The additional term of 1 is added in to account for the crack influences, bringing the total number of terms to 9.

Extending the model to incorporate pair-wise influences increases the number of terms dramatically. Using the existing definitions of 13 neighbors, and 3 labels in the label set, the total number of influences is found using Equation 6 to be 819 terms. We can immediately drop all references to Crack terms and add a single 1 to the end of the matrix. Grouping similar influences together in order to simplify **A** further is not as straightforward as in the linear model. For example, when considering the pair-wise term $n_3 n_7$, if we use the same groupings as in the linear model, this term would be grouped with $n_1 n_5$. However, $n_1 n_7$ cannot be grouped in this set, since the two pairwise influences are not symmetric about $p_x$. After like terms are grouped together, there are 61 terms in the **A** matrix. Therefore, the size of the matrix is decreased by about 93%!

Since the training data are ideal, the probabilities for a given label will be either 1 or 0. This creates problems in the solution of the least squares approximation. The quantity $\mathbf{A}^t\mathbf{A}$ must be invertible. If linear influences as well as the square terms are considered, and $n^2 = n$ for $n \in \{0, 1\}$, then each of the square terms will be equal to some linear term and the matrix would not be linearly independent. Therefore, the square terms were dropped from consideration and the linear influences were left.

## 4.7 Training

The least squares approximation requires a set of training data to derive the coefficient vector from. In the training phase, the data set is considered

*ideal*. Ideal data are data which have been correctly and consistently labeled by some previous method. Guaranteeing that the data are consistent and correct allows the system to mathematically "learn" the allowable (consistent) label combinations. Allowable label combinations are represented as positive weights in $\vec{x_0}$, which serve to strengthen correct combinations of neighboring label sets that arise when performing the relaxation labeling. Likewise, inconsistent (forbidden) label combinations will have a negative weighting in $\vec{x_0}$, which serves to weaken the inconsistent labelings that exist in the data.

Inconsistencies are not "hard-wired" into the system through explicit domain knowledge. Inconsistent labels are represented negatively in the $\vec{x_0}$ vector because such inconsistent combinations never arise in the ideal case. When the system uses the fits to the ideal data set in the relaxation algorithm, points which lie far from the ideal fit are considered inconsistent, and can be adjusted to be closer to the ideal data.

# 5 Results

Image 4T was used as the training image. The image is a $512 \times 512$ image of portland concrete. This image was hand labeled for training purposes.

A single-scale system was implemented to demonstrate the need for a system which incorporates information from the multiple scales. Image 4B is the input image and image 5T is the initial labeling to the system. Note the regions of aggregate which are incorrectly labeled as crack, and the regions of the aggregate which lie in the crack. After four iterations, the system is hopelessly lost, as shown in image 5B.

Since the two dimensional relaxation system was unable to correctly label

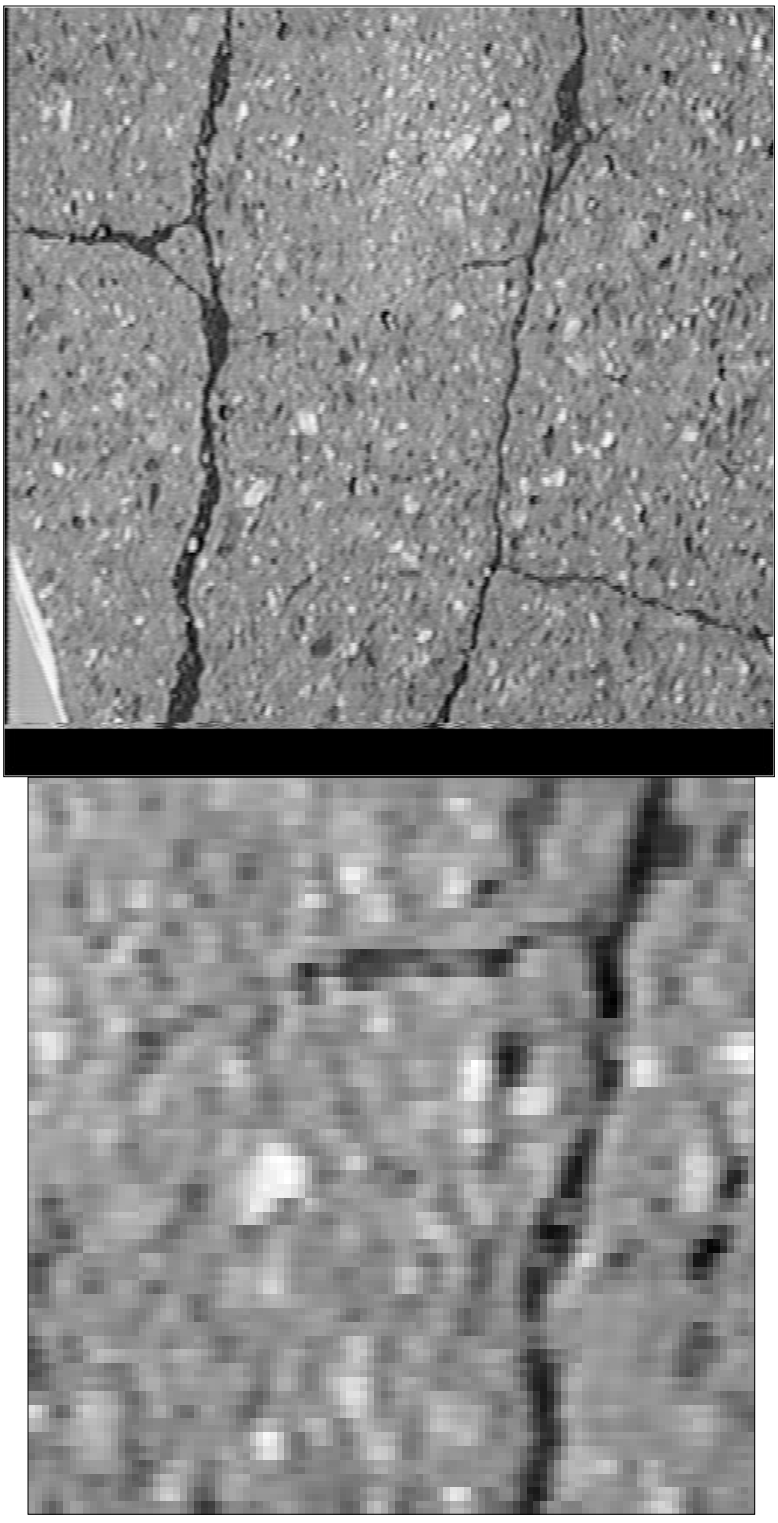

Figure 4: Top: Training Image. Bottom: Portland Concrete Distress Image.

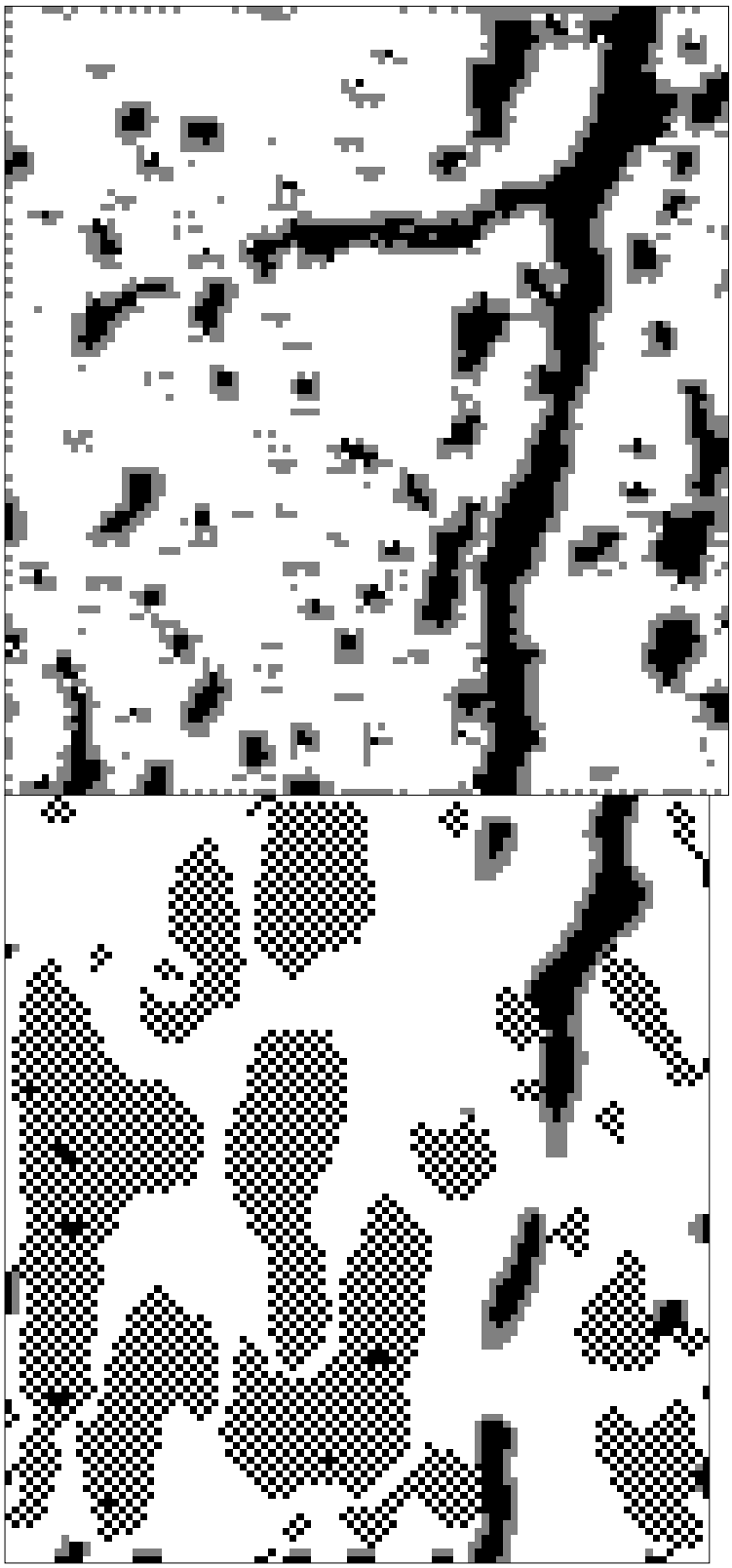

Figure 5: Top: Initial Solution. Bottom: After Fourth Iteration.

the images correctly, the multiple scale relaxation system was used instead. The same image and initial guess are used as in the single-scale experiment (Images 4B and 5T, respectively). Figure 6 depict the progress of the relaxation updates 5 and 25 iterations.

# 6 Conclusions

Single-scale relaxation methods were shown to be inadequate. This can be attributed to several factors. First of all, the fractal property of distress images was not capitalized upon. Second, distress images are very noisy. Third, in single-scale relaxation, if erroneous adjustments are made, there is no information to help revert the label set to its previous state, and therefore any bad changes get perpetuated.

When the multi-scale relaxation method was implemented, it was shown to address each one of these problems. By using multiple levels, the fractal property of the distress could be exploited. The use of multiple levels helps to handle image noise Finally, since the method is hierarchical in nature, there is some degree of history. This allows the system to stay consistent between levels. Even a simplified linear model using the hierarchical, multi-scale system far outperforms single-scale relaxation.

Finally, non-linear influences were considered. When the non-linear coefficients are applied to the images, there is an immediate improvement over all previous methods considered. Noise in the image is removed in just a few iterations, larger pieces of "non-crack" area are eaten away and eventually removed. Small cracks are detected, and do not get eaten away due to their fractal property. Also, the quality of the input images is not a major factor;

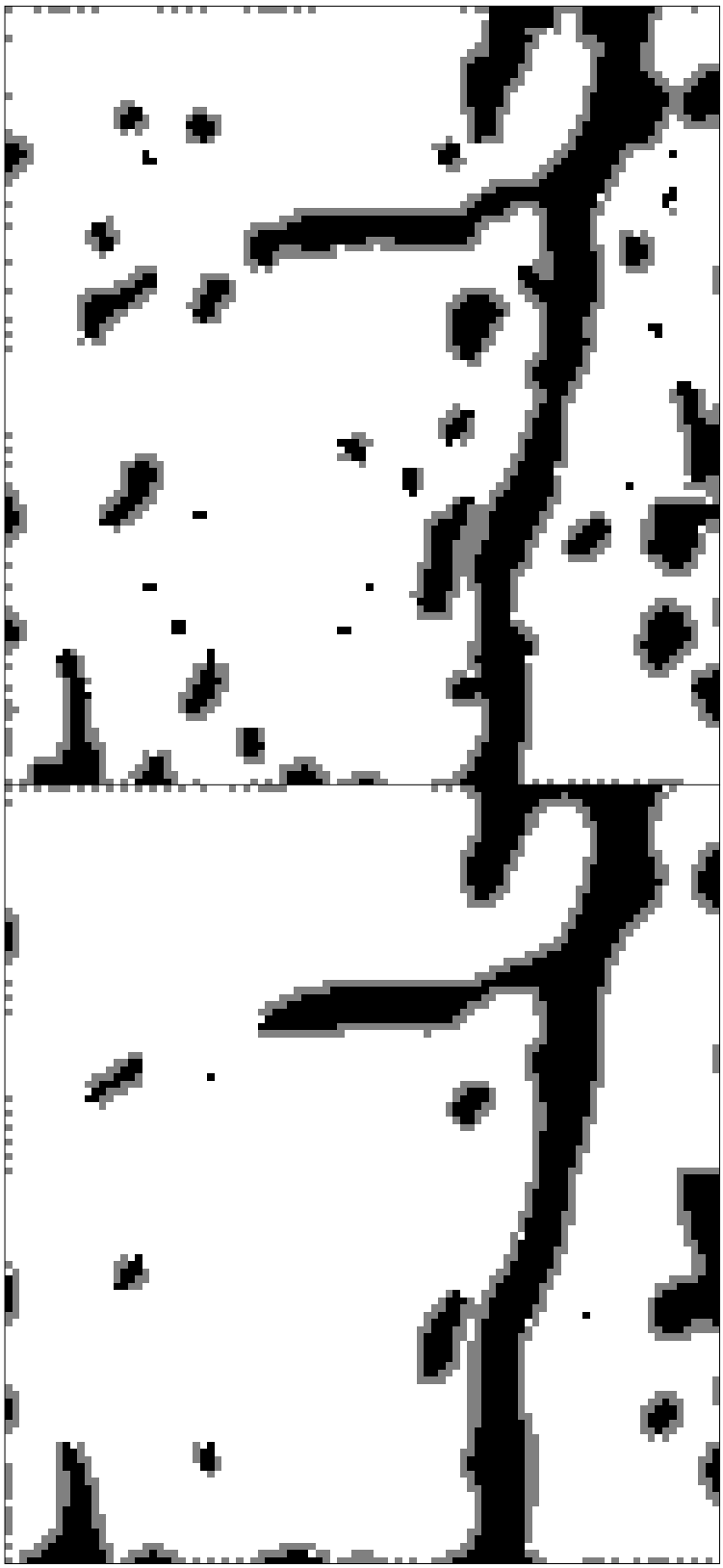

Figure 6: Top: After 5 Iterations. Bottom: After 25 Iterations.

a better initial image just makes the system converge to the ideal fit quicker.

The process of accurately detecting and isolating distress within pavement images has been demonstrated to be a non-trivial task. Our method demonstrated encouraging results, and although only a subset of types of distress were used in this work, we believe that such a system could be expanded to work on many types of distress with similar success.

# References


[CHI83] Chien CH, Martin WN, Meyer AH, Aggarwal JK "*Detection of Cracks on Highway Pavements*," Report No. FHWA/TX-83/7+256-3; 1983.

[CUR84] Curphey DR, Fronek D, Shan WJ, Wilson JE "*Pavement Evaluation Using Image Processing*," Proc. ISA Intern. Conf., pp. 1043-1052; 1983.

[ELK90] El-Korchi T, Wittels N "*Visual Appearance of Surface Distress in PCC Pavements: I Crack Luminance*," Trans. Resch. Rec., 1260, pp. 74-83, 1990.

[ELK91a] El-Korchi T, Gennert MA, Ward MO, Wittels N "*Distinguishing Aggregate from Distress in Pavement Images*," SHRP IDEA Proposal on Pavement Performance, September 1991.

[ELK91b] El-Korchi T, Gennert MA, Ward MO, Wittels N "*Lighting Design for Automated Pavement Surface Distress Evaluation*," Transportation Research Record, Vol. 1311, pp. 144 - 148, 1991.

[GEN91] Gennert MA, Wittels N, LeBlanc J, Gosselin D "*Analysis and Generation of Pavement Distress Images Using Fractals*," Trans. Resch. Rec., February 1991.

[GLE87] Gleik J, "*Chaos: Making a New Science*," Viking, New York, 1987.

[GOS91] Gosselin DR "*The Use of Second Directional Derivatives in Pavement Distress Images*," WPI Computer Science Technical Report; WPI-CS-TR-91-6, April 1991.

[HAR84] Haralick RM "*Digital Step Edges from Zero Crossing of Second Directional Derivative*," IEEE PAMI; vol. 6, pp. 58-68, 1984.

[HUM83] Hummel RA, Zucker SW, "*On the foundations of relaxation labeling processes*," IEEE Trans PAMI, vol 5, no 3, 1983, pps 267-286.

[MAN82] Mandelbrot B, "*The Fractal Geometry of Nature*," W. H. Freeman, New York, 1982.

[SOL91] de Solminihac H, Roper H "*Using Strip Films to Record Pavement Distress in the SHRP-LTPP Study*," Transportation Research Board Annual Meeting, Paper 910397, pp. 2-4, January, 1991.

[TOR84] Torre V, Poggio T "*On Edge Detection*," MIT AI Lab Memo 773, 1984.