1995

Vision system development for a surface-mount chip placement machine

Thomas P. Walpole
Lehigh University

AUTHOR:

Walpole, Thomas P.

TITLE:

Vision System Development for a Surface-Mount Chip Placement Machine

DATE: May 28, 1995

Vision System Development for a Surface-Mount Chip Placement Machine

by

Thomas P. Walpole

A Thesis

Presented to the Graduate Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical Engineering

Lehigh University

May 2, 1995

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science

5/9/95
Date

Thesis Advisor

Chairperson of Department

Acknowledgments

Terry Banyard - Sigma Systems International
for the opportunity to work on a real world project.

Dr. Alastair McAulay - Lehigh University EECS Department
for the opportunity to attend graduate school through the acquisition of funding.

Brian Costello - Lehigh University - undergraduate
for the work he has done.

Table of Contents

Abstract

To be able to place surface mount chips accurately and rapidly onto a circuit board, it is necessary to have a description of where the chip is to be placed on the board, where the chip is now, where the board is now, and a method to move the chip from point x to point y accurately. In this paper we describe the steps we have taken so far to implement the first three of the requirements listed above using multiple cameras and a vision system. The final component has already been developed and is planned for development into a commercial product.

Introduction

This project deals with the implementation of a machine for placing surface mount components on circuit boards at approximately twice the rate of any other commercially available machine. To this purpose the hardware for a machine has been constructed using multiple linear motors and a vision system with multiple cameras . This thesis deals with some of the things needed to implement the vision system portion of the machine.

The vision system is based on a Coreco F-64[1] image capture board with the ability to switch among multiple cameras. The board also hosts a Texas Instruments DSP chip to enable images to be processed without the need for transfer across the bus of the machine. The system is planned to be run on a minimum of an Intel 486DX2-66 machine running the Microsoft Windows operating system.

The vision system needs to be able to complete two basic tasks. The first task is the inital scanning of the circuit board. In this step the board needs to be scanned into the computer and stored so that the locations for placement of components can be entered either interactively by a user or through a definition file. If entered through a definition file the image will also be used to verify that there are contact pads at the area specified in the file as a way to check for a valid file. The maximum size circuit board this machine will handle is twelve inches by eighteen inches. This board is scanned in images of approx. one half inch by one half inch, with each image being a 512x512 8-bit grayscale image. This corresponds to 864 images of 262,144 bytes each, or 216 megabytes of data plus some overhead for image headers. This amount of information would be easier to manage, store, manipulate if some form of compression was utilized.

The second task that needs to be performed is when a component is picked up, we need to know the exact orientation and location of that component. This is needed to allow for accuracy in placing the component on the board. We know the orientation within plus or minus two degrees and we also know the location within a few pixels, however we need a better level of accuracy.

Both of these tasks are discussed in the following sections, with the results obtained towards the end. This machine is an ongoing project and I hope to see a completed prototype in the near future.

<u>Board Image Compression</u>

Fractal Image Compression

To be able to use the scanned image of the circuit board we needed to compress it down to a more usable size. One of the methods tried for this was automatic fractal compression. The "automatic" refers to the fact that no interactive geometrical modeling is required. This automatic compression is accomplished by extending IFS theory from global to local and restricting attention to affine symmetry transformations.

To begin fractal compression we need three basic ingredients[2] which lead to four theorems and expectations. The three ingredients needed are

1. A model Y for the space R of real world images where each "point" in Y represents a real world image, has a support □ , chromatic attributes, resolution independence, and closure under the application of arbitrary invertible affine transformations applied to clipped parallelograms within images, chosen to yield rectangular images. . The support □ is a set

$\square \subset R^2$, where R^2 denotes the Euclidean plane. $\square$ is defined by

$$\square=\{(x,y)\in R^2 : a\leq x\leq b,\ c\leq y\leq d\} \tag{1}$$

where $a<b$ and $c<d$ are real constants.

2 A metric d on the space Y, such that (Y,d) is a complete metric space.

3.A contractive operator O, which acts upon the space (Y,d). That is, the operator O is such that there exists a real number s with $0\leq s<1$ and

$$d(O(\phi),O(\psi)) \leq s\cdot d(\phi,\psi) \text{ for all } \phi,\psi\in Y \tag{2}$$

With these three ingredients we consequentially have the following four theorems and expectations[3]

(1) Theorem (Existence of Attractors)

Since O is contractive and the metric space Y is complete, there exists an unique image $\phi \in Y$ such that

$$O(\phi)=\phi \tag{3}$$

(2) Expectation (Fractal Character of Attractors)

We anticipate that ϕ has a resolution independent character because of the contractivity of the functions from which O is constructed: the whole invariant image is the same as a sum or union of contractions applied to it, and thus it is made of shrunken copies of (parts of) itself. Depending on the way in which the contractions act, the focus may be on spatial contractivity, intensity contractivity, or measure theoretic contractivity, and we expect that the attractor ϕ will inherit corresponding fractal characteristics.

(3) Theorem (Computation of Attractors)

To compute ϕ, we can use the fact that if $\psi \in Y$ then the result of repeatedly applying O to ψ converges to the attractor ϕ; that is

$$\lim_{n=\infty} O^{\circ n}=\phi \tag{4}$$

Moreover, if there exists a real constant C such that

$$d(\phi_1,\phi_2)<C \text{ for all } \phi_1,\phi_2 \in Y, \tag{5}$$

then we have the error estimate

$$d(O^{\circ n}(\psi),\phi)\leq s^{n}C \tag{6}$$

(4) Theorem (General Collage Theorem Estimate)

The distance between $\psi \in Y$ and the attractor ϕ of O is bounded by the estimate

$$d(\phi,\psi) \leq \frac{d(\psi,O(\psi))}{(1-s)} \tag{7}$$

In fractal transform theory the basic ingredients 1,2 and 3 are provided by local transformations which are assembled in various ways to produce operator O and a corresponding system for generating resolution independent fractal attractors.

We now consider applying IFS theory to local transformation in the case of binary images.

Definition Let(X,d) be a compact metric space. Let R be a non-empty subset of X. Let w:R->X and let s be a real number with $0 \leq s < 1$. If

$$d(w(x),w(y)) \leq s \cdot d(x,y) \textit{ for all } x,y \textit{ in } R \qquad \textbf{(8)}$$

then w is called a local contraction mapping on (X,d). The number s is a contractivity factor for w.

Definition Let (X,d) be a compact metric space and let w_i:$R_i \rightarrow$X be a local contraction mapping on (X,d), with contractivity factor s_i, for I=1,2,...,N, where N is a finite positive integer. Then

$$\{w_i : R_i \Rightarrow X: i = 1,2,\ldots,N\} \qquad \textbf{(9)}$$

is called a local iterated function system (local IFS). The number s=max{s $_i$:i=1,2,...,N} is called the contractivity function of the local IFS.

A simple algorithm to compute the attractor A of a local IFS is the escape time algorithm[4].

$$Let \;\; D = \bigcup D_i \;\; and\ define$$
$$f : D \to \square \tag{10}$$
$$by \;\; f(x) = w_i^{-1}(x) \; for \; x \in D_i \; for \;\; i = 1,2,3,\ldots,N.$$

This says that D_i is a piece of the image D, and we have defined a function f as an inverse affine transformation w_i^{-1} such that $f(D_i)=R_i$. Because of this we use the notation R_i for the domain of w_i, and D_i for its range rather than the other way around. The piecewise affine function

$$f : D \subset \square \to \square \tag{11}$$

where $\square$ is the support of D, provides a dynamical system whose repelling set is the attractor associated with the local IFS. The regions D_i are called domain blocks while the regions R_i are called range blocks.

The attractor of the IFS , A, and a decreasing sequence of approximations A_n to A are computed as follows:

(0) Initialize a counter to zero, and specify a maximum number of iterations n.

(1) Input x ∈ □

(2) Is (x ∈ D)? If yes , replace x by f(x), If no output "x ∉ A" and exit

(3) Increment counter; if counter = n, output "x ∈ A_n and exit

(4) GOTO (1)

In practice this algorithm yields,

$$A_n = W_{local}^{on}(D) \supset A$$

assuming n is finite. If n=∞ and the input belongs to A then the algorithm will run on endlessly.

Taking this algorithm and applying it to image compression of binary images gives us the following algorithm.

(0) Input a binary image G, a subset of □⊂ R^2

(1) cover G with domain blocks, D_i, . The complete set of domain blocks (D_i I=1,2,...,n) must cover G. Blocks do not overlap one another.

(2) Introduce a collection of possible range blocks $R\subset\Box$, such that $R\cap G\neq 0$. These are squares whose sides are twice as long as those of the domain blocks. The possible coordinate of the lower left corner of each possible range block (R_x,R_y) are restricted to lie in a finite set L. Correspondingly, we define a collection T of local contractive affine transformations, mapping from range block R to the domain block D_i. That is for I=1,2,...n,

$$T_i = \{w(D_i, R_x, R_y, j):(R_x, R_y)\in L \;;\; j = 0,1,2,...7\} \tag{13}$$

Where $w(D_i,R_x,R_y,j)$ is the contractive affine transformation with domain R, range D_i, of the form

$$0.5A(j)\cdot R + t \tag{14}$$

where A(j) denotes the jth symmetry in Table-1 and t is a constant.

Symmetry	Matrix	Description
0	0 1 1 0	identity
1	-1 0 0 1	reflection in y-axis
2	1 0 0 -1	reflection in x-axis
3	-1 0 0 -1	180° rotation
4	0 1 1 0	reflection in line y=x
5	0 1 -1 0	90° rotation
6	0 -1 1 0	270° rotation
7	0 -1 -1 0	reflection in line y=-x

Table-1 Symmetry Matrices[5]

(3) Carry out the fractal transform process as follows. For each i choose $w_i \in T_i$ to minimize the Hausdorff distance.

$$h(w_i(R \cap G), D_i \cap G) \qquad \textbf{(15)}$$

That is, for each domain block, one chooses a corresponding range block and symmetry, so that the transformed part of the image in the range block looks most like the part of the image in the domain block.

The set $$W_{local}(G) = \bigcup w_i(R \cap G) \qquad (16)$$

is called the collage of the image G corresponding to the local IFS, while the number

$$h(w_i(R \cap G),\ D_i \cap G) \qquad (17)$$

is called the corresponding collage error.

(4) Write out the compressed data in the form of a local IFS code.

(5)Apply a lossless data compression algorithm to the local IFS code, to obtain a compressed local IFS code.

Now that we have an algorithm for binary images we can extend this to be applied to grayscale images. We use the model for the space of real world images R that consists of the space Y of all real-valued functions

ϕ: □→ I. Here, I=[a,b] ⊂ R is a real interval that represents the possible grayscale intensity values in images, such as [0,255].

We convert Y into a complete metric space by defining the distance between two functions ϕ_1, $\phi_2 \in$ Y as

$$d(\phi_1,\phi_2) = sup\{|\phi_1(x,y) - \phi_2(x,y)|:(x,y)\in\square\} \qquad \textbf{(18)}$$

where sup denotes the suprenum, the smallest number M with the property that x≤ M for all x elements of S.

Now Let D denote a partition of □ consisting of a finite collection of sets $D_i \subset \square$, I =1,2,...,M., that is

$$\square = \bigcup_{M}^{i=1} D_i \qquad \textbf{(19)}$$

where D_i intersect D_j = 0 for i ≠ j. For each i let f_i:D_i→□, with $f_i(D_i)$= R_i, and let v_i:R→R be a contractive transformation with contractivity factor s, with 0<=s<1 that is

$$|v_i(z_1) - v_i(z_2)| < s \cdot |z_1 - z_2| \text{ for all } z_1,z_2 \in R \qquad \textbf{(20)}$$

for i=1,2,...,M.

$$(F\psi)(x,y) = v_i(\psi\ (f_i(x,y))\ when\ (x,y)\in D_i \qquad \textbf{(21)}$$

Then we define F:Y→Y by

We say that s is the contractivity factor of F. We call F the fractal transform operator.

Theorem (Convergence of fractal transforms)[6]

Let the complete metric space (Y,d) and the operator F:Y->Y be defined as above. Then F is a contraction mapping on Y; that is, for all $\phi_1,\phi_2 \in L^\infty(\square)$,

$$d(F(\phi_1),F(\phi_2)) \leq s\cdot d(\phi_1,\phi_2), \qquad \textbf{(22)}$$

where s is the contractivity factor of F.

This gives us the three ingredients from before needed for a fractal image compression system using F for the operator O. In particular, there exists a unique function $\phi \in$ Y such that F(ϕ)=ϕ. The function ϕ is called the attractor of the fractal transform . To compute ϕ, we can use the fact that if $\psi \in Y$ then the result of repeatedly applying F to ψ converges uniformly to the attractor Φ, that is

$$\lim_{n \to \infty} F^{on}(\psi) = \phi \tag{23}$$

which gives the error estimate

$$|F^{on}(\psi)(x,y) - \phi(x,y)| \leq s^{n} |b-a| \textit{ for all } (x,y) \in \square, \textit{ where } I=[a,b] \tag{24}$$

The distance between $\psi \in Y$ and the attractor of the fractal transform operator is bounded by the estimate

$$d(\phi,\psi) \leq \frac{d(\psi,F(\psi))}{(1-a)}, \tag{25}$$

This is the collage theorem for the fractal transform operator F.

Huffman/RLE encoding

Huffman coding is a lossless form of compression in which the input symbols are replaced with output symbols based on the probability of occurrence of the input symbol. The more probable an input symbol the shorter the output symbol that represents it should be. In our case the inputs symbols were the gray level values of the image being compressed, where the probability of occurrence of each symbol can be derived from a histogram of the input image..

An algorithm to generate the optimum Huffman code for a given input string is shown below[7]

(1) List the input symbols in order of probability

(2) Make a tree whose branches, labeled zero and one, are the two symbols with the lowest weight.

(3) Remove the two symbols just used from the list and add to the list a new symbol representing the newly formed tree with probability equal to the total weight of the branches.

(4) Make a tree whose branches, labeled zero and one, are the two symbols with lowest weight in the new list. This tree may

consist of two other symbols, or it could consist of a symbol and the tree just constructed.

(5) Repeat this procedure until one large tree is formed

.

At each stage in the merging of the symbols into a complete tree the branches are arbitrarily labeled 0 or 1, although following a simple rule such as the lowest probability tree is 0 will make it easier to follow.

As an example consider an input alphabet with five symbols {A,B,C,D,E} with probabilities

P_A=.20, P_B=.05, P_C=.34, P_D=.16, P_E=.25

The steps taken are illustrated in Figure-1.

Step 1- Merge {B,D} to form a tree with weight PBD=.21

Step 2- Merge {BD,A} to form a tree with weight PBDA=.41

Step 3-Merge {C,E} to form a tree with weight PCE=.59

Step 4-Merge {BDA,CE} to form the final tree with weight PABCDE=1

From Figure-1 we can see that the codewords for the output symbols are

A=00, B=010, C=11, D=011, E=10

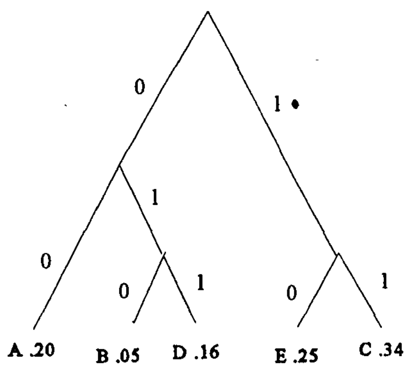


Figure 1-Construction of Huffman Code

Component Orientation

Hough Transform

To calculate the orientation of the chip in the captured image we used the Hough algorithm[8]. We knew that we were looking for straight lines (edges of the pins) and we knew a range of angles in which these lines would lie.

All points on a line will satisfy the equation for that line, $y=mx+c$, where m is the slope of the line, change in y per unit change in x, and c is the y-intercept. To try to find these lines we therefore take each image points' coordinates and find which line equations, within the specified limits for the lines , they will satisfy. The procedure is outlined below[9].

(1) Define space

Quantize parameter space between appropriate minimum and maximum values for c and m.

(2) Setup

Form an array A(c,m) with elements initially zero to act as an accumlator.

(3) Process

For each point (x,y) in a gradient image (see below) such that the strength of the gradient exceeds a set threshold, increment all points in the accumulator array where x,y satisfy the line equation. That is

$$A(c,m) = A(c.m)+1 \textit{ for } m,c \textit{ satisfying } c=-mx+y \textit{ for } C_0 \leq c \leq C_1, M_0 \leq m \leq M_1. \quad \textbf{(26)}$$

(4) Results

Local maxima in the accumlator array now correspond to collinear points in the image array, with the values in the accumlator array providing a measure of the number of points on the line.

A gradient image is an image to which the gradient operator has been applied. For an image function f(x), the gradient magnitude s(x) and direction ϕ(x) can be computed as

$$\begin{aligned} s(x) &= (\Delta_1^2+\Delta_2^2)^{1/2} \\ \phi(x) &= \tan^{-1}(\Delta_2/\Delta_1) \end{aligned} \qquad \textbf{(27)}$$

where

$$\begin{aligned}\Delta_1 &= f\,(x+n,\ y)-f\,(x,y)\\ \Delta_2 &= f\,(x,\ y+n)-f\,(x,y)\end{aligned} \tag{28}$$

The operators used to calculate these on a given image are[10]

Δ_1

0	1
-1	0

Δ_2

1	0
0	-1

in the case that n=1. These operators are convolved with the input image with the results giving the gradient image.

The calculation of the Hough transform, although it provided the required results to the desired degree of accuracy, proved to be computationally intensive and proved to take too long to be useful in this project.

Linear Regression

In trying to reduce the computation we realized that we knew approximately where the edge we were looking for was located and that we could find the edge points relatively rapidly. Once we had the edge points we needed a way to calculate the equation, and therefore the slope and intercept, of the line these points lay on. To do this we used first order linear regression[11].

A straight line relating two quantities x and y can be described by the equation

$$Y = m x + c \tag{29}$$

where c is the intercept, the y value when x=0, and m is the slope. There will also be some error introduced into the equation of the line due to the sampling that takes place in capturing of the image in digital form. Because of this our model for the line becomes[12]

$$y_i = c + m x_i + e_i \quad i = 1,2,\ldots,n \tag{30}$$

Now to correctly obtain the parameters for the desired line we need to minimize the error term. For the ith case the observed residual (fitting error) is

$$\hat{e}_i = y_i - (\hat{c} + \hat{m}x_i) \quad i=1,2,\ldots,n \tag{31}$$

, note the hats over m and c refer to the estimates of m and c while we are using the same hat notation to specify the observerd error, which can be compared to the equation for statistical errors,

$$e_i = y_i - (c + mx_i) \quad i=1,2,\ldots,n \tag{32}$$

The method we used to minimize the errors is least squares estimation. In this method we select c and m to make the residual sum of squares, RSS, as small as possible where

$$RSS = \sum_{i=1}^{n} \hat{e}_i^2 = \sum_{i=1}^{n} [y_i - (\hat{c} + \hat{m}x_i)]^2. \tag{33}$$

Now to minimize this we can differentiate with respect to c and m. set the derivatives to zero, and solve the resulting equations[13].

$$\frac{\delta RSS}{\delta\hat{c}} = -2\sum(y_i-\hat{c}-\hat{m}x_i)=0$$
$$\frac{\delta RSS}{\delta\hat{m}} = -2\sum x_i(y_i-\hat{c}-\hat{m}x_i)=0 \tag{34}$$

Which can be rearranged to

$$\hat{c}n+\hat{m}\sum x_i=\sum y_i$$
$$\hat{c}\sum x_i \ +\hat{m}\sum x_i^2=\sum x_iy_i \tag{35}$$

Now the estimates for c and m can be found by solving these two equations simultaneously to obtain

$$\hat{c}=\sum y_i/n \ -\hat{m}\sum x_i/n$$
$$\hat{m}=\frac{\sum(x_i-\sum x_i/n)(y_i-\sum y_i/n)}{\sum(x_i-\sum x_i/n)^2} \tag{36}$$

Now we define

$\bar{x}=\sum x_i/n$	*Sample average for the* $x_i^/s$	
$\bar{y}=\sum y_i/n$	*Sample average for the* $y_i^/s$	(37)
$SXX=\sum(x_i-x)^2$	*Corrected sum of squares for the* $x_i^/s$	
$SXY=\sum(x_i-\bar{x})(y_i-\bar{y})$	*Corrected sum of cross products*	

Which leads to

$$\hat{m} = \frac{SXY}{SXX} \qquad (38)$$
$$\hat{c} = \bar{y} - \hat{m}\bar{x}$$

Thus given a set of sample (x, y) pairs we can calculate all items needed to obtain c, and m.

Results

Board Image Compression

The Images below (Figure-2 through Figure-5) are sample images that were compressed using both Huffman/RLE encoding and Fractal compression followed by Huffman/RLE. Table-2 shows the sizes of the files before and after compression using the different compression methods. The table also shows the size of a file after converting to binary and then running Huffman/RLE on it.

As can be seen from Table-2 the fractal compression performed much better than the Huffman/RLE alone, as expected, and as well as if not better than converting to binary and then running Huffman/RLE. The times listed in Table-2 are times on a 486DX2-66 machine. Although the fractal compression is lossy, there were no noticeable artifacts, nor additives left on the circuit board images tested, Figure 6 through Figure 9,

	Orig. Size (bytes)	Huff./RLE % comp.	Fractal % comp.	Binary Huff./RLE %comp.	$T_{Huff.}$	$T_{Frac.}$
Board 1	262,930	37%	90%	93%	5	127
Board 2	262,930	30%	87%	95%	4	121
Board 3	262,930	29%	91%	91%	3	142
Board 4	262,930	28%	84%	89%	3	157

and all images were fully usable for component placement. The detriment of fractal compression was the time it took to compress the images to these ratios. When scanning a full size twelve inch by eighteen inch circuit board there will be a total of eight hundred and sixty four images taken. With compression times for one image running into the multiple minutes fractal compression creates a time burden.

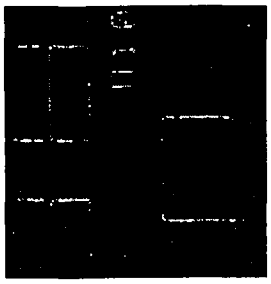


Figure 2-Board 1

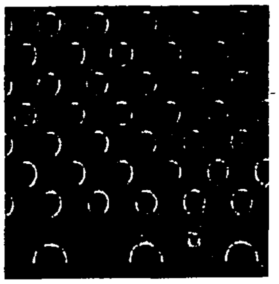


Figure 3-Board 2

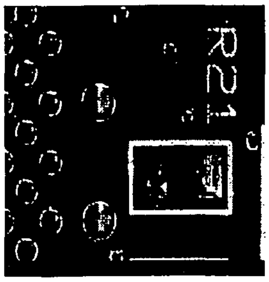


Figure 4-Board 3

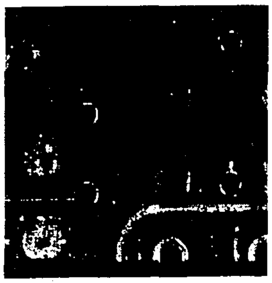


Figure 5-Board 4

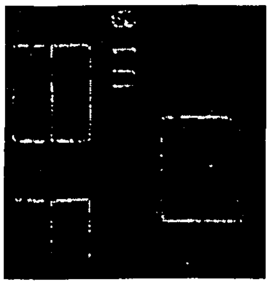


Figure 6-Board 1
after reconstruction
from Frac. Comp.

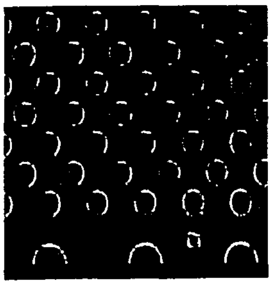


Figure 7-Board 2
after reconstruction
from Frac. Comp.

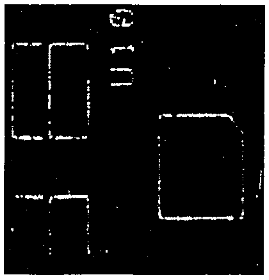


Figure 2-Board 1

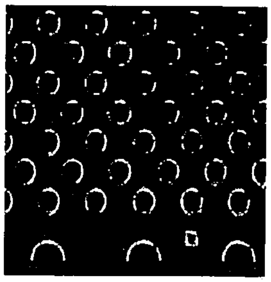


Figure 3-Board 2

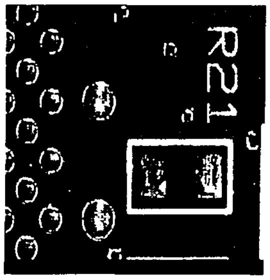


Figure 4-Board 3

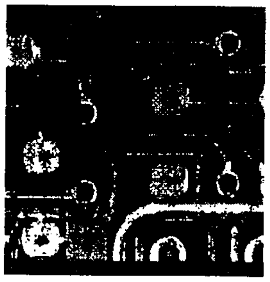


Figure 5-Board 4

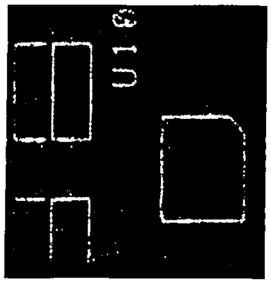


Figure 6-Board 1
after reconstruction
from Frac. Comp.

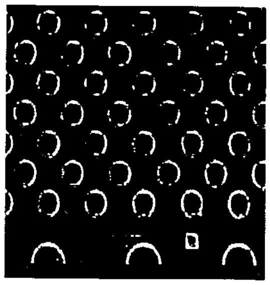


Figure 7-Board 2
after reconstruction
from Frac. Comp.

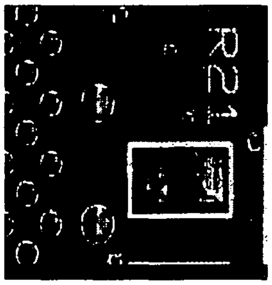


Figure 8-Board 3 after reconstruction from Frac. Comp.

Figure 9-Board 4 after reconstruction from Frac.Comp.

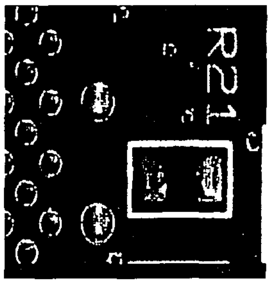


Figure 8-Board 3 after reconstruction from Frac. Comp.

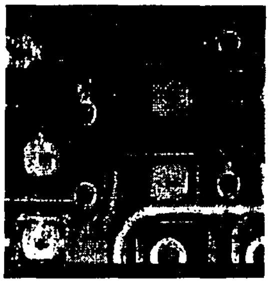


Figure 9-Board 4 after reconstruction from Frac.Comp.

Component Orientation

Below can be seen images of a chip (Figure-10 through Figure-12) scanned at a few angles around forty five degrees from horizontal and Table-3 lists the angles and intersect points determined by our edge hunting-linear regression algorithm. All images are of a surface mount chip against a piece of non-reflective black paper.

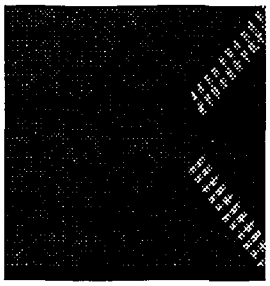


Figure 10-Chip 1

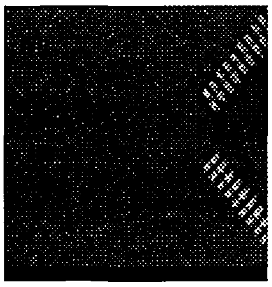


Figure 11-Chip 2

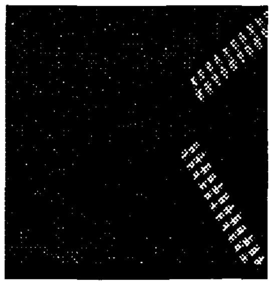


Figure 12-Chip 3

Figure 13 shows the chip orientation used to obtain the images in Figure-10 through Figure-12.

Component Orientation

Below can be seen images of a chip (Figure-10 through Figure-12) scanned at a few angles around forty five degrees from horizontal and Table-3 lists the angles and intersect points determined by our edge hunting-linear regression algorithm. All images are of a surface mount chip against a piece of non-reflective black paper.

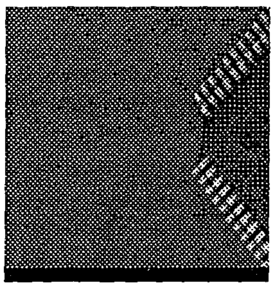


Figure 10-Chip 1

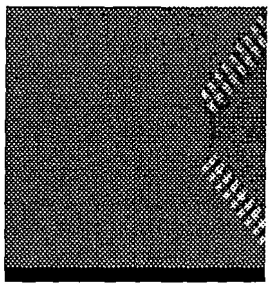


Figure 11-Chip 2

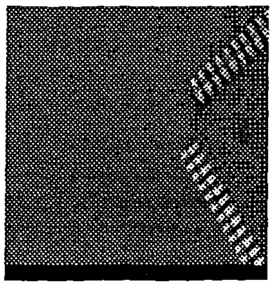
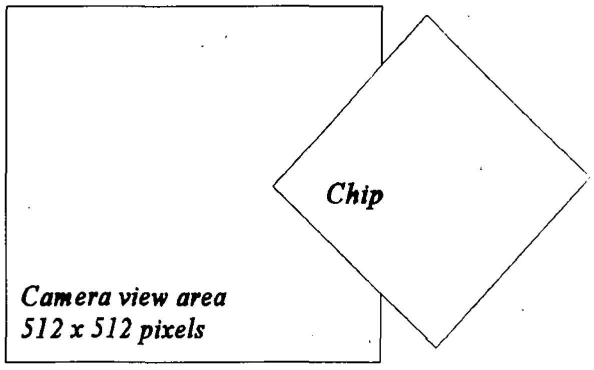


Figure 12-Chip 3

Figure 13 shows the chip orientation used to obtain the images in Figure-10 through Figure-12.



Background is black non-reflective paper

Figure 13- Chip position for capture

	Chip 1	Chip 2	Chip 3
Angle (Degrees)	44.1	43.3	45.1
(x,y) coord pixels	(260.3,258.1)	(270.3,262.2)	(257.8,261.2)

Table-3 Numeric results of component orientaion

As can be seen from Table-3 we managed to calculate both the angle and the corner point (the intersection of the lines along the outside edges of the pins). These values were calculated within .1 degrees and within the dimensions of 1 pixel. With these two pieces of information we can rotate the chip to the desired angle of rotation and also know the amount of x,y offset to compensate for during placement of the component on the board.

References

1.Coreco Inc.
6969 Trans Canada Highway
Unit 113
Saint-Laurent
Quebec, Canada
H4T 1V8

2. M. Barnsley and L. Hurd, *Fractal Image Compression*. AK Peters, Ltd, Wellesley, MA (1993) p. 174

3. M. Barnsley and L. Hurd, *Fractal Image Compression*. AK Peters, Ltd, Wellesley, MA (1993) p. 175-176

4. M. Barnsley, *Fractals Everywhere*, Academic Press, Boston (1988).

5. M. Barnsley and L. Hurd, *Fractal Image Compression*. AK Peters, Ltd, Wellesley, MA (1993) p. 180

6. M. Barnsley and L. Hurd, *Fractal Image Compression*. AK Peters, Ltd, Wellesley, MA (1993) p. 186-187

7. R.W. Hamming, *Coding and Information Theory*, Prentice Hall, Englewood, NJ (1980)

8. Duda, R. O. And P. E. Hart. "Use of the Hough transformation to detect lines and curves in pictures/" *Commun. ACM 15*, 1, January 1972,11-15

9.D. Ballard and C. Brown, *Computer Vision,* Prentice-Hall Inc., Englewood Cliffs, NJ (1982)

10.D. Ballard and C. Brown, *Computer Vision,* Prentice-Hall Inc., Englewood Cliffs. NJ (1982) p. 77

11.Weisberg, *Applied Linear Regression*, john wiley & sons, New York, NY (1980)

12.K.Shanmugan and A. Breipohl, *Random Signals - Detection, Estimation and Data Analysis,*John Wiley and Sons, Inc., New York, NY, (1988)

13.Weisberg, *applied linear regression*, john wiley & sons, New York, NY (1980) pp.242-243

Vita

Thomas P. Walpole was born in London, England on December 13, 1971 to Mr. Thomas M. Walpole and Mrs. Pauline M. Walpole. He enrolled at Lehigh University in 1989, graduating with a Bachelor of Science in Computer Engineering in January of 1994. He continued his education attending graduate school at Lehigh while working as a research assistant for Dr. Alastair McAualay in the Lehigh University Advanced Electronic Manufacturing Laboratory. His main interests lie in the the areas of design and application of vision system hardware and software. After completing his M.S. in Electrical Engineering in May, 1995 he plans to enter into industry.

END
OF
TITLE