Integrated Circuit Detection from a PCB: QC Application

Steps to build+run+test:

- make detect
- ./detect ic_1.png

In order to implement the recognition of the IC's I tried implementing a different set of approaches and finalized on the one that suited the best. The description of our approaches has been explained in brief in the following:

Linear Filtering

I used a low-pass filter to perform the linear filtering in the spatial domain in order to remove the noise and the unnecessary elements from the image as a result of the blurring operation. To do so, I experimented with 2 types of smoothing filters:

- Gaussian Blurring filter
- Box Blurring filter

The results of the Gaussian blurring filter were more significant in the case of our application as it reduced the white noise in the image better. The mean (Gaussian) filter was then convolved with the grayscale image of the PCB. In order to handle the border conditions properly during the convolutions, I have *padded* the image on *all 4 sides with Zeros*. Therefore I are omitting the 1st few pixels in the image before detecting the IC's as they do not contribute to the edge pixels for rectangular images.

To scale the result in the intensity domain of the output image, the result of the convolution was decided by the criterion which fit the requirement better:

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Edge Detection (Calculating the Gradients of the Image)

In order to detect the edges of the grayscale image, I segregated the edge pixels by computing the gradient of the image intensity function. To do so, I made use of the Sobel-Feldman Operator which is an Isotropic 3x3 discrete differentiation operator.

The operator was used to obtain both vertical and horizontal edges in the grayscale image using G_x as the Sobel operator for the x direction and G_y as the operator in the y direction. I first convolved the image with the whole 3x3 filter in the X and Y directions to check the results and then converted each of G_x and G_Y into separable filters. The Sobel Operators and their separable forms are as follows:

G_x =

1	0	-1
2	0	-2
1	0	-1

$\Rightarrow [1 \quad 2 \quad 1]^T [1 \quad 0 \quad -1]$

G_y =

1	2	1
0	0	0
-1	-2	-1

$\Rightarrow [1 \quad 2 \quad 1] [1 \quad 0 \quad -1]^T$

The G_x and G_y were convolved with the image (say A) in a sequential manner such that the results of the intermediate image transformations were also stored(X *and* Y) as follows:

$X = G_x * A$ and $Y = G_y * A$

To obtain the spatial intensities of the image back from the original image, the gradient magnitude was calculated as follows:

$$G = \sqrt{G_x^2 + G_y^2}$$

After doing so, I applied a threshold on the resultant to again filter out the unnecessary components from the image. On a few experiments (on the given 9 PCB images) I fixed this Threshold 'T' to a value of 50.

Final Thresholding

After applying the linear filter and applying the Sobel Operator, I used the images to come up with a model to find an adaptive threshold. The data I used to come us with this function to define the adaptive threshold is:

IC Number	Dimensions	Average Pixel Intensity	Calculated Threshold
1	571 x 571	2.26534	10
2	300 x 360	3.89601	16.79
3	300 x 360	2.55	11
4	900 x 900	2.82	12.1
5	500 x 640	5.97	25.7
6	1318 x 2565	3.15	13.61
7	1371 x 2413	4.1207	17.78
8	1359 x 2192	1.96 ~ 2	9
9	1374 x 1308	3.6897	15.89

Adaptive Threshold = (Average Pixel Intensity) * 3.0

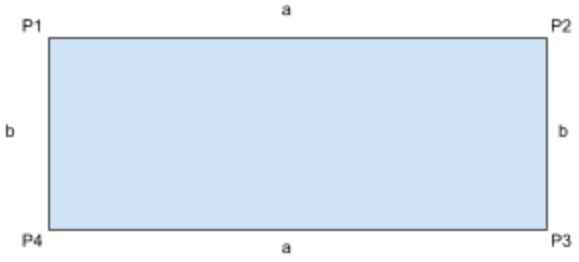
Hough Transform

After removing the edge pixels which were below the threshold, I made use of the Hough Transform to detect the edges and enforce which edge belonged to which IC. To implement the Hough Transform I transformed the image from the Cartesian Coordinate system into the Hough Space which is represented in Polar Coordinates, i.e. Angle-Distance parameter space.

$$\rho = x\cos\theta + y\sin\theta$$

Here ρ represents the distance from the origin to a line and θ represents the angle of the line with the origin (taken as the center of the image as given in the textbook (u_c, v_c) in [1]). The value of ρ ranges from $-diag\_len$ to $diag\_len$ [refer code] and the values of θ have been incremented from 0^o *to* 180^o by an increment of 1 degree to best calculate each edge with maximum number of pixels. I have made use of an accumulator which is a 2D array that stored the values $\rho\, vs\, \theta$ where the number of rows = number of ρ values and the number of columns = number of θ values.

Rectangle Patterns in Hough Space



To find the rectangular pattern, with (P1,P2) and (P3,P4) being parallel sides with length a, as well as (P2,P3) and P4,P1) with length b.

In the Hough Space, I will get 4 peaks $H_1(\rho_1,\theta_1))$, $H_2(\rho_2,\theta_2)$, $H_3(\rho_3,\theta_3)$, $H_4(\rho_4,\theta_4)$. The basic idea of the proposed algorithm is to search every pixel (x,y) of the image, compute the Hough

Transform of the edge image in a certain neighborhood centered at (x,y), find relevant peaks of the HT, and use the following:

Let H1 = (ρ1,θ1), H2 = (ρ2,θ2),..., Hm = (ρm,θm) denote the m peaks. To find the 4 peaks, i.e. the parallel lines that form our Rectangular IC, all peaks are scanned, and peaks Hi , Hj are paired together if they satisfy the following conditions:

➔ $\Delta\theta = |\theta i - \theta j| < T\theta$

➔ $\Delta\rho = |\rho i + \rho j| < T\rho$

➔ $|C(\rho i, \theta i) - C(\rho j, \theta j)| < TL * C(\rho i, \theta i) + C(\rho j, \theta j)/2$

- Tθ is an angular threshold, and determines if peaks Hi and Hj correspond to parallel lines (i.e. θi ≈ θj).
- Tρ is a distance threshold, and is used to check if the lines related to Hi and Hj are symmetric with respect to the θ axis (i.e. ρi ≈ −ρj).
- TL is a normalized threshold, and determines if line segments corresponding to Hi and Hj have approximately the same length (i.e. C(ρi ,θi) ≈ C(ρj ,θj))
- Each pair of peaks Hi and Hj satisfying the above equations generates an extended peak **Pk = (±ξk,αk)**, where

$$\alpha k = 1/2 * (\theta i + \theta j) \text{ and } \xi k = 1/2 * |\rho i - \rho j|$$

It should be noticed that Pk encodes information about both peaks Hi and Hj , because θi ≈ αk,θj ≈ αk,ρi ≈ −ξk and ρj ≈ ξk.

The final step of the proposed technique is to compare all pairs of extended peaks Pk and Pl , and retrieve those that correspond to orthogonal pairs of parallel lines (hence, related to a rectangle).

A rectangle is then detected if:

Δα = ||αk − αl| − 90◦| < Tα … [2]

where Tα is an angular threshold that determines if pairs of lines Pk and Pl are orthogonal. Finally, the vertices of the detected rectangle are obtained through the intersection of the two pairs of parallel lines. The orientation of the detected rectangle is given by αk, and the sides are given by 2ξk and 2ξ.

After a number of simulations, I set our thresholds to Tθ = 3◦ , Tρ = 3 and TL = 0.4

Removing Repeating/ Duplicate Rectangles:

The thresholds that I have used to compute the degree of parallelism between the sides of the rectangle Tθ, Tρ, Tα and TL may generate duplicated rectangles for neighboring centers.

Our approach was to compute an error measure for each detected rectangle, and to choose the rectangle for which the error is the smallest. Each rectangle is represented by a pair of extended peaks Pk and Pl , and there is an orthogonality error measure Δα which I computed in the previous steps.

Furthermore, each of these extended peaks were obtained by comparing a pair of regular peaks Hi and Hj , and their parallelism and distance error measures Δθ and Δρ, according to conditions where the thresholds were declared.

Thus, there are five error measures related to each rectangle: Δθk, Δθl , Δρk, Δρl and Δα. The proposed error measure is given by:

E(Pk,Pl) = $\sqrt{a\,(\Delta\theta\,2\,k\,+\,\Delta\theta\,2\,l\,+\,\Delta\alpha 2)\,+\,b\,(\Delta\rho\,2\,k\,+\,\Delta\rho\,2\,l\,)}$

This E(Pk,Pl) obtained was used to calculate the **confidence value** of detection for the given IC. The confidence was calculated as follows:

Confidence = [E(Pk,Pl)/10000]

where a and b are weights for angular and distance errors, respectively. In our application, I have taken the a =1 and b = 4. To implement this, I have used **Non Maximal Suppression** to find the unique pairs of the points in the Hough Space that minimize our error function. This is carried out by determining for every cell in Accumulator: Acc(θ, r) whether the value is higher than the value of all of its neighboring cells. The coordinates of the remaining peaks are potential line

parameters, and their respective heights correlate with the strength of the image space line they represent.

Modified accumulator updating

Since the purpose of the accumulator array is to find the intersections of 2D curves. Due to the discrete nature of the image and accumulator coordinates, rounding errors usually cause the parameter curves for multiple image points on the same line not to intersect in a single accumulator cell.

Therefore for a given angle $\theta=i\theta*\Delta\theta$, I incremented not only the corresponding accumulator cell Acc(iθ, ir) but also the neighboring cells Acc(iθ, ir−1) and Acc(iθ, ir+1). This makes the Hough transform more tolerant against inaccurate point coordinates and rounding errors.

REFERENCES:

1. Principles of digital image processing: core algorithms / Wilhelm Burger, Mark J. Burge.
2. Rectangle Detection based on a Windowed Hough Transform Claudio Rosito Jung and Rodrigo Schramm - IEEE paper
3. https://alyssaq.github.io/2014/understanding-hough-transform