Chapter 3

THEORETICAL BACKGROUND

3.1. Introduction

This chapter covers the theoretical background for template-matching by correlation-based filtering and Gabor filtering, and line detection using the Hough transform. One of the most fundamental techniques of object detection is template matching [Ritter96]. In a traditional template matching, a template represents a replica of an object of interest, and the template is compared to all parts of the image. If the template and a portion of the image are sufficiently similar, then the object is labeled as being an instance of the template object.

Gabor functions (GFs) are Gaussian functions modulated by complex sinusoids, and Gabor filtering has been shown to be a powerful technique for object detection in cluttered images [Casasent96]. The imaginary part of a GF is a good edge detector while the real part is an excellent blob detector. For these reasons, the GF was considered in this research. However, as shall be shown in Section 4.1, this approach did not yield good results.

The Hough transform is a versatile method for detecting shapes, such as lines, circles, and ellipses. However, only detection of straight lines will be discussed in this chapter.

3.2. Correlation-based Matching

Correlation-based matching is a simple, yet very popular technique in pattern recognition with images. The basic method consists of forming a correlation measure between a template T and an image I ($I \otimes T$) and determining the location of a correspondence by finding the location of maximum correlation. Usually I is much larger than the template.

A common measure of dissimilarity is

$$d = \sum (I - T)^2 = \sum I^2 - 2\sum I\, T + \sum T^2 \tag{3.1}$$

where the sum is over pixels in the template T and the image I. The value of d will be small when I and T are almost identical and large when they differ significantly. Since the term $\Sigma(IT)$ will have to be large whenever d is small, a large value of $\Sigma(IT)$ similarly indicates the location of a good match of the template. Notice that $\Sigma(IT)$ is the cross-correlation of the two signals I and T.

This simple cross-correlation is called unnormallized, and it has several major shortcomings. If a certain pixel in the image has an abnormally large value, the $\Sigma(IT)$ associated with this location is likely to have a large value even though no good match exists. And since zeros do not contribute to the correlation value, a good match containing zeros will possibly have a small value. To overcome these problems, normalized cross-correlation (NCC) is often used instead, in which the measure is formulated as [Abbott95]:

$$\rho(u,v) = \frac{\sum_x \sum_y \left[\tilde{T}(x,y) \tilde{I}(x+u, y+v) \right]}{\sqrt{\sum_x \sum_y \tilde{T}^2(x,y)} \sqrt{\sum_x \sum_y \tilde{I}^2(x+u, y+v)}} \tag{3.2}$$

In this equation, I is an image, and T is a template; $\tilde{I} = I - \mu_i$, and μ_i is the average intensity of I within the window being considered; $\tilde{T} = T - \mu_T$, and μ_T is the average intensity of T. This formula keeps ρ in the range -1 to 1, with a value close to 1 indicating a better matching.

For (3.2) to be effective, a prior accurate knowledge of the pattern T is required. For the detection of detonators, we assume that the shape of a detonator is fixed and known. However, the orientation of the detonator in images can be changing, and multiple templates with varying orientations have to be used.

The algorithm for object detection for a single orientation of the detector using NCC is as follows:

```
Algorithm:  NCC_Detection
Given: Input image $I$; orientation of the detonator.
1. Initialize an output array with the same size as the input image.
2. Set an initial threshold value for threshholding the correlation result.  Normally, it is
   set at 0.7-0.8.
3. Repeat the following steps until the result is satisfactory.
        3.1. Construct the correlation filter template T using an empirical model
             method.
        3.2. Perform normalized cross-correlation of $I$ with $T$, and store the result
             in the output array.
        3.3. Threshold these correlation values.  If any value is larger than the
             threshold, set the corresponding point in the output array to 1,
             otherwise to 0.
```

3.4. If the result is not satisfactory, adjust the threshold or the correlation filter template (see details in Section 4.2) and continue.

4. End repeat.

3.3. Hough Transform

The Hough transform uses templates described by a set of parameters, such as the slope and intercept of a line [Hough62, Duda72]. The standard Hough transform detects curves whose shape can be described using an analytical equation. In this section we will only cover the method for finding straight lines using the standard Hough transform.

A straight line in the sense of the Hough algorithm is a collinear set of points. Each point on a line votes for several combinations of parameters. Thus the number of points (pixels) along the line determines the quality of the straight line. The Hough transform is often represented as a mapping into the function space of sinusoidal functions defined by

$$r = x\cos(\theta) + y\sin(\theta) \tag{3.3}$$

Any straight line l_0 in the x-y plane corresponds to a point (r_0, θ_0) in the r-θ plane, where $\theta_0 \in [0,\pi)$ and $r_0 \in \Re$. Let n_0 be the line normal to l_0 that passes through the origin of the x-y plane. The angle n_0 makes with the positive x-axis is θ_0. The distance from (0, 0) to l_0 along n_0 is $|r_0|$. Fig. 3.1 illustrates the relation between l_0, n_0, θ_0, and r_0. Note that the x-axis in the figure corresponds to the point (0, $\pi/2$), while the y-axis corresponds to the point (0, 0) [Ritter96].

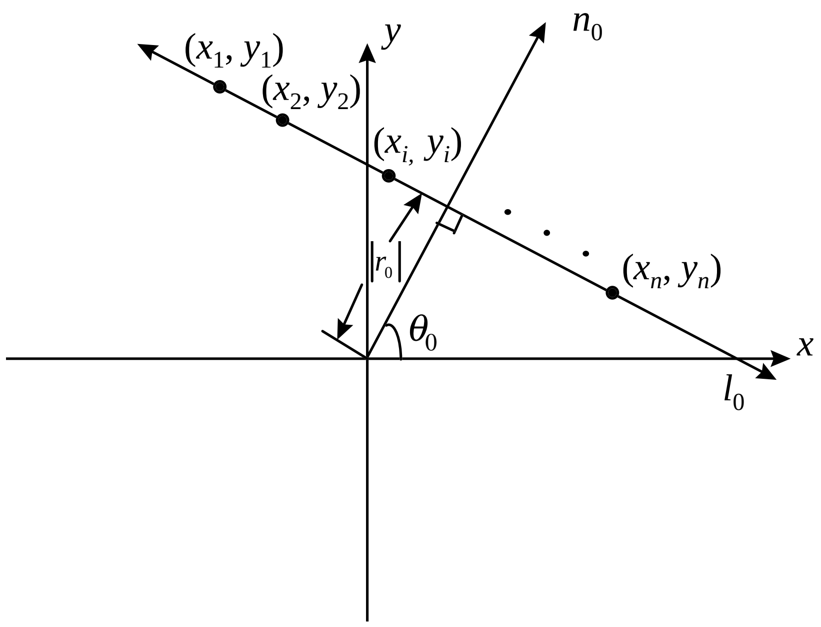


Fig. 3.1. Relation of rectangular to polar representation of a line [Ritter96].

Suppose (x_i, y_i) are points in the x-y plane that lie long the straight line l_0. The line l_0 has a representation (r_0, θ_0) in the r-θ plane. The Hough transform takes each of the points (x_i, y_i) to a sinusoidal curve $r = x_i \cos(\theta) + y_i \sin(\theta)$ in the r-θ plane. The property that the Hough algorithm relies on is that each of the curves $r = x\cos(\theta) + y\sin(\theta)$ has a common point of intersection, namely (r_0, θ_0). Conversely, the sinusoidal curve $r = x\cos(\theta) + y\sin(\theta)$ passes through the point (r_0, θ_0) in the r-θ plane only if (x, y) lies on the line $r = x\cos(\theta) + y\sin(\theta)$ in the x-y plane.

Each point (x, y) at a feature pixel in the domain of the image maps to a sinusoidal function by the Hough transform. Therefore the number of feature pixel points that lie along the line (r_0,θ_0) in the x-y plane can be determined by counting the number of sinusoidal curves in the r-θ plane that intersect at the point (r_0,θ_0).

Algorithm: Hough_Line_Detection

Given: Binary input image *I*.

1. Quantize the parameter space appropriately.
2. Assume that each cell in the parameter space is an accumulator, i.e. the parameter space is an accumulator array. Initialize all cells to zero.
3. For each foreground point (x, y) in the image space, increment by 1 each of the accumulators that satisfy the equation $r = x\cos(\theta) + y\sin(\theta)$.
4. Maxima in the accumulator array correspond to the parameters of the model instances.

3.4. Gabor Filters

The Gabor wavelet filter function (GF) can be written as [Casasent96]

$$g_x = \frac{w}{2\pi|A|^{-\frac{1}{2}}}\exp\left[j\omega_0^T(x-x_0)\right]\exp\left[-\frac{1}{2}(x-x_0)^T A^{-1}(x-x_0)\right] \tag{3.4}$$

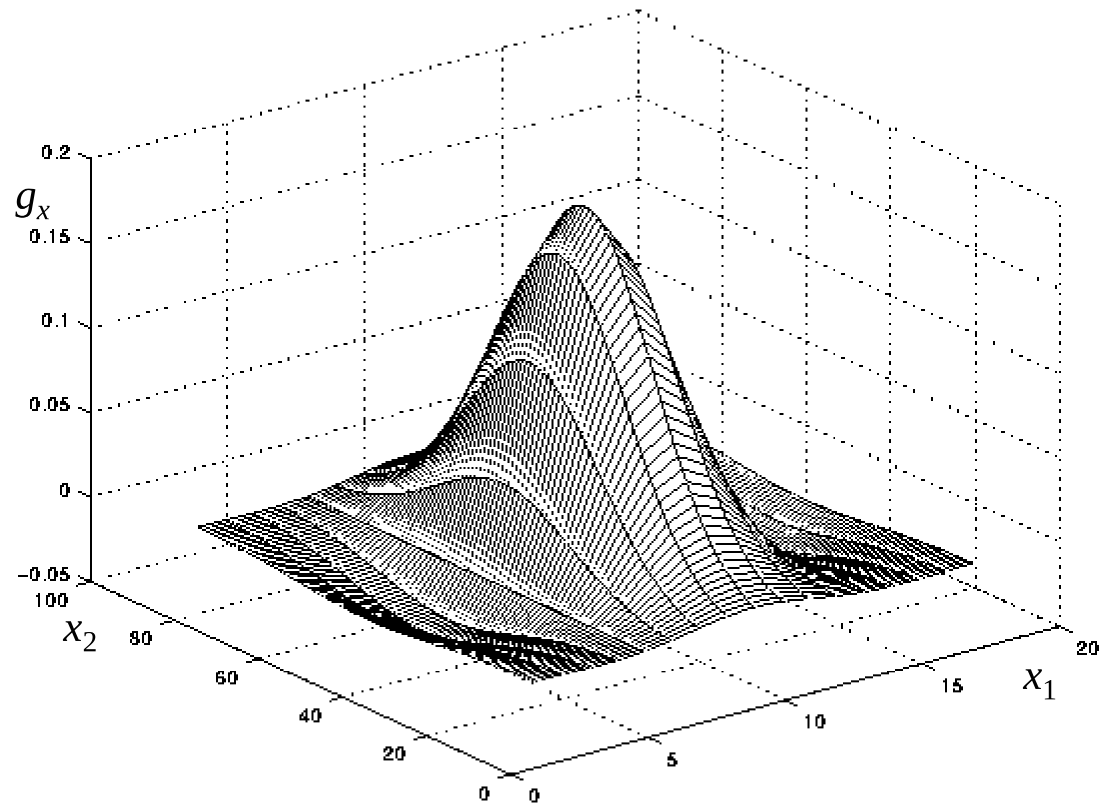
The vector $x = [x_1, x_2]^T$ describes the two dimensional spatial Cartesian domain, and the scalar variable w is a weight parameter. The normalization factor $1 \Big/ \left(2\pi |A|^{-\frac{1}{2}} \right)$ keeps the area under the Gaussian part of (3.4) constant as we change A. The vector x_0 shifts the function, and $A = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}$ is a matrix describing the shape of the Gaussian part of the Gabor function, where a and b determine the variance of the Gaussian part of the function. The modulation frequency $\omega_0 = [\pi/2a, 0]$ is only in one direction (x_1 axis) since we use GFs to detect only object height or width (RGFs) or edges in one direction (IGFs) [Weber95]. To obtain GFs with different orientations, we apply the coordinate transformation $\tilde{x} = Jx$ and $\tilde{x}_0 = Jx_0$ with

$$J = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{3.5}$$

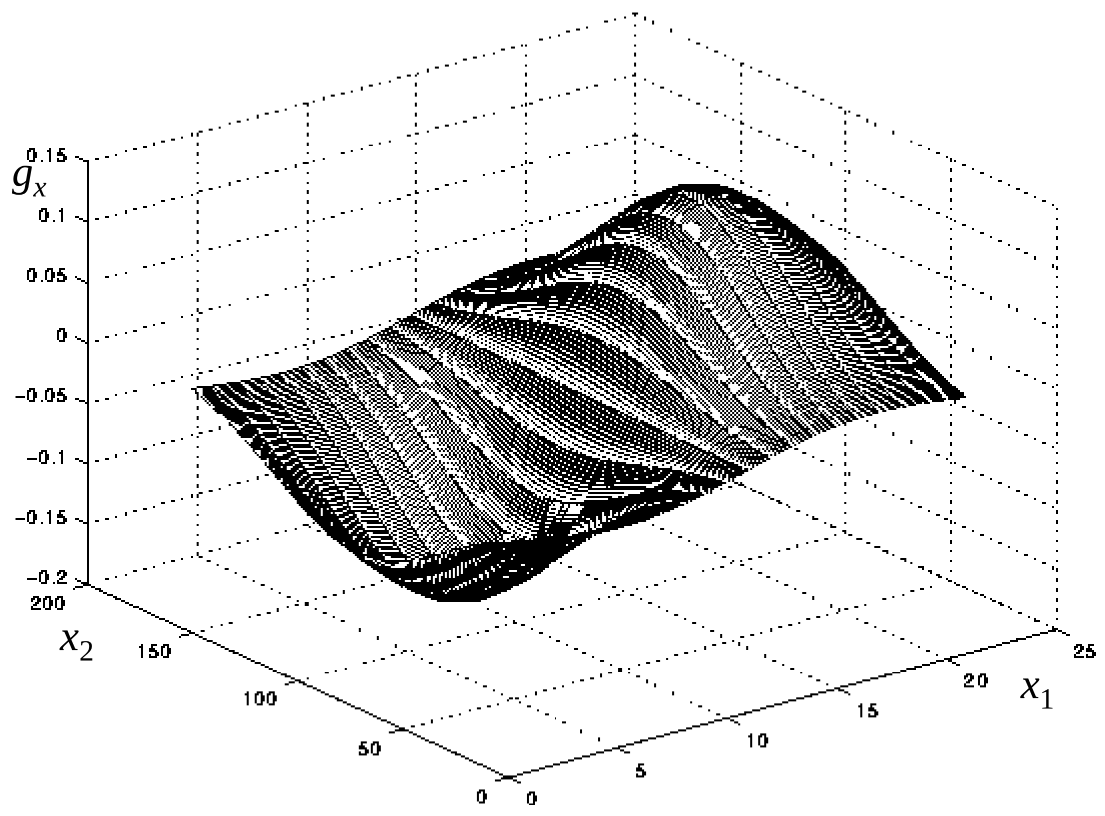
This rotates the GF by angle ϕ.

The four parameters controlling the GFs can be collectively described by a vector $n = \left[a\ b\ w\ x_0^T \right]^T$. Different choices of vector n produce different Gabor functions. The cross-correlation of g_x with an input image I produces one Gabor wavelet output, and the outputs for all g_x are the Gabor transform of I. The correlation $I \otimes g_x$ produces a 2-D gray scale output. The value at each output pixel indicates the amount of a given radial spatial frequency ω at an angle ϕ present in I in a local region (defined by a and b) about each pixel.

A Gabor function g_x consists of a real part and an imaginary part. The real part of g_x (RGF) with $\phi = 0$ has the form shown in Fig. 3.2a, which can be used as an object blob detector since the central positive lobe detects objects and the two negative lobes detect the background around the objects. The imaginary part of g_x with $\phi = 0$ has the form in Fig. 3.2b. It has a zero mean with a positive and a negative lobe and serves as a good oriented edge detector. Normally, two directional RGFs were used to detect object height and width, and four directional IGFs to detect the left, right, top and bottom edges of the object [Weber95]. However, due to the elongated shape of the detonator the imaginary Gabor function will not be used here.



(a)



(b)

Fig. 3.2. Gabor function plots for RGF (a) and IGF (b). Here, a = 2, b = 10, ω_0 = [$\pi/2a$, 0]. Note the large positive lobe and two smaller negative lobes each side of the RGF, and one large positive and one large negative lobe of the IGF.

A RGF can be formulated as:

$$g_r(x) = \frac{w}{2\pi |A|^{-\frac{1}{2}}} \cos\left(\omega_0^T (x - x_0)\right) \exp\left[-\frac{1}{2}(x - x_0)^T A^{-1} (x - x_0)\right] \tag{3.6}$$

and its output automatically gives a peak at the center of an object. For RGFs, a wider template for objects is desirable because a wider RGF is less sensitive to variation in object width/height[Weber95]. This poses an intrinsic limitation on its applicability in this problem because of the extremely narrow width of the detonator.

The algorithm for the RGF is:

Algorithm: Real_Gabor_Filtering

```
Given: Input image $I$.
1. Initialize the parameters in the real Gabor function.
2. Initialize an output array with the same size as the input image.
3. Repeat the following steps until result is satisfactory.
    3.1. Generate the 2-D template corresponding to the real Gabor function.
    3.2. Cross-correlate the input image $I$ with the filter template.
    3.3. The maximum peak of the correlation result should correspond to the object that we are detecting.
    3.4. If the result is not satisfactory, adjust the parameters of the real Gabor Filter (see details in Chapter 4) and continue.
4. End repeat.
```

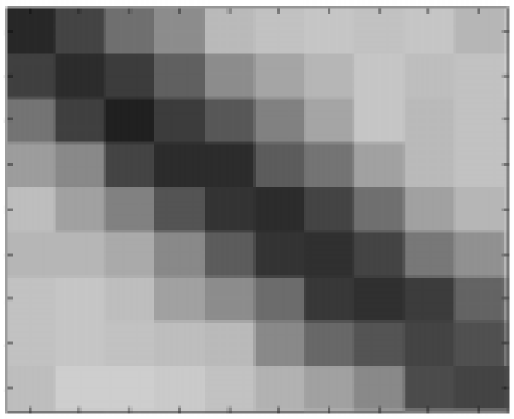
3.5. Information Fusion

Although simple template matching techniques may work well for some cases, there are several drawbacks. One is that simple template matching can require a relatively large template and can be computationally intensive. In our problem, since the size of the detonator is about 32×23 pixels for some orientations, the size of the template is also the same. This reduces the calculation speed dramatically. The second drawback is that it is very difficult to distinguish two similar objects by using simple template matching only. For example, in our problem, because of the elongated shape of the detonator, false alarms can occur when the objects with a similar shape is encountered. The third drawback is that it is very sensitive to the orientation of the detonator. A slight change in the orientation will cause the method to fail. To overcome these problems of simple template matching, a more sophisticated algorithm using information fusion technique will be developed. Details of the approach will be described in the next section.

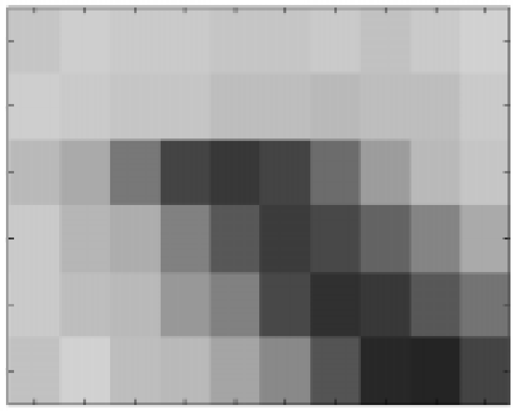
3.6. Approach of this Thesis

The detection task can be carried out in two steps: a detection step and a fusion step. In the detection step for a given orientation, three templates will be used (Fig. 3.3). One template is used for detecting the middle part of the detonator and the other two for detecting the left end and the right end respectively. Because now the middle part and the end points are the target objects that we are detecting instead of the whole detonator, the size of the template is significantly reduced and the problem of orientation sensitivity is mitigated. The block diagram is shown in Fig. 3.4.

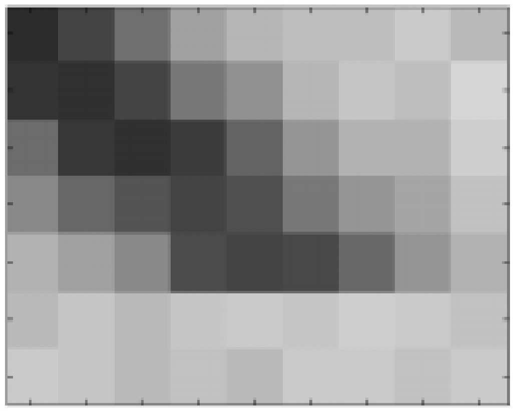
Let h_1 represent the middle template, and h_2 and h_3 the two end templates. First, each individual template is scanned over the whole input image, and three outputs are generated. By appropriate threshholding, the final locations of a possible good match for end and middle parts can be determined. This gives three binary outputs, which are called the middle output, the left-end output, and the right-end output, respectively. When a non-zero pixel is found in the middle output, this point, say $(\tilde{x},\tilde{y})$, determines a window in which the Hough transform is used to search for lines. The window may be expressed as $x \in [\tilde{x},\tilde{x}+dx]$ and $y \in [\tilde{y},\tilde{y}+dx]$, where [$dx$, dy] is the size of the window. This is shown in Fig. 3.5. The window size is chosen to be similar to that of the detonator at that orientation. The Hough transform is calculated using only outputs generated by the middle template, and by using $r = x_i \cos(\theta) + y_i \sin(\theta)$. The number of feature pixel points that lie along the line (r_0,θ_0) in the input image can be determined by counting the number of sinusoidal curves in the r-θ plane that intersect at the point (r_0,θ_0). After appropriate threshholding, a possible good match of a straight line can be detected. This process is repeated for all the non-zero output pixels in the middle output.



(a)



(b)



(c)

Fig. 3.3. Example templates used in this research. These were obtained empirically with an in-plane detonator orientation of 38°. (a) Middle template h_1. (b) Left-end template h_2. (c) Right-end template h_3.

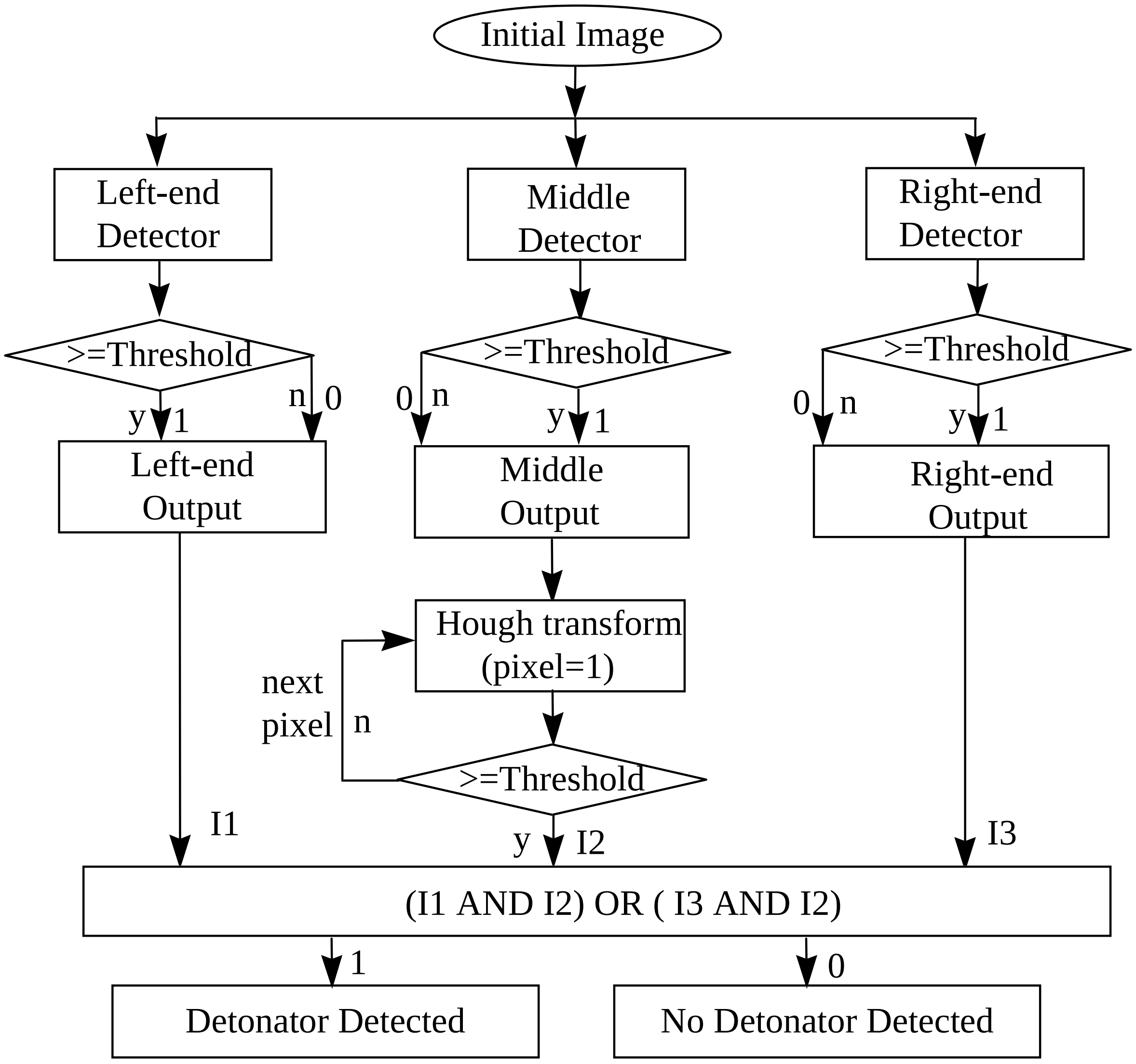


Fig. 3.4. Block diagram for the detonator detection algorithm.

After the locations of a possible good match are determined by the middle detector and the Hough transform, the algorithm moves on to the fusion step, in which this output from the middle detector is combined with the outputs from the end detectors. If there are no middle outputs that satisfy a given threshold, we say that there is no detonator in the image. If there are one or more middle outputs that satisfy a given threshold, the end outputs are used to see whether there are some end points at the corresponding end positions along the line. When this is true, we say that a detonator is detected.

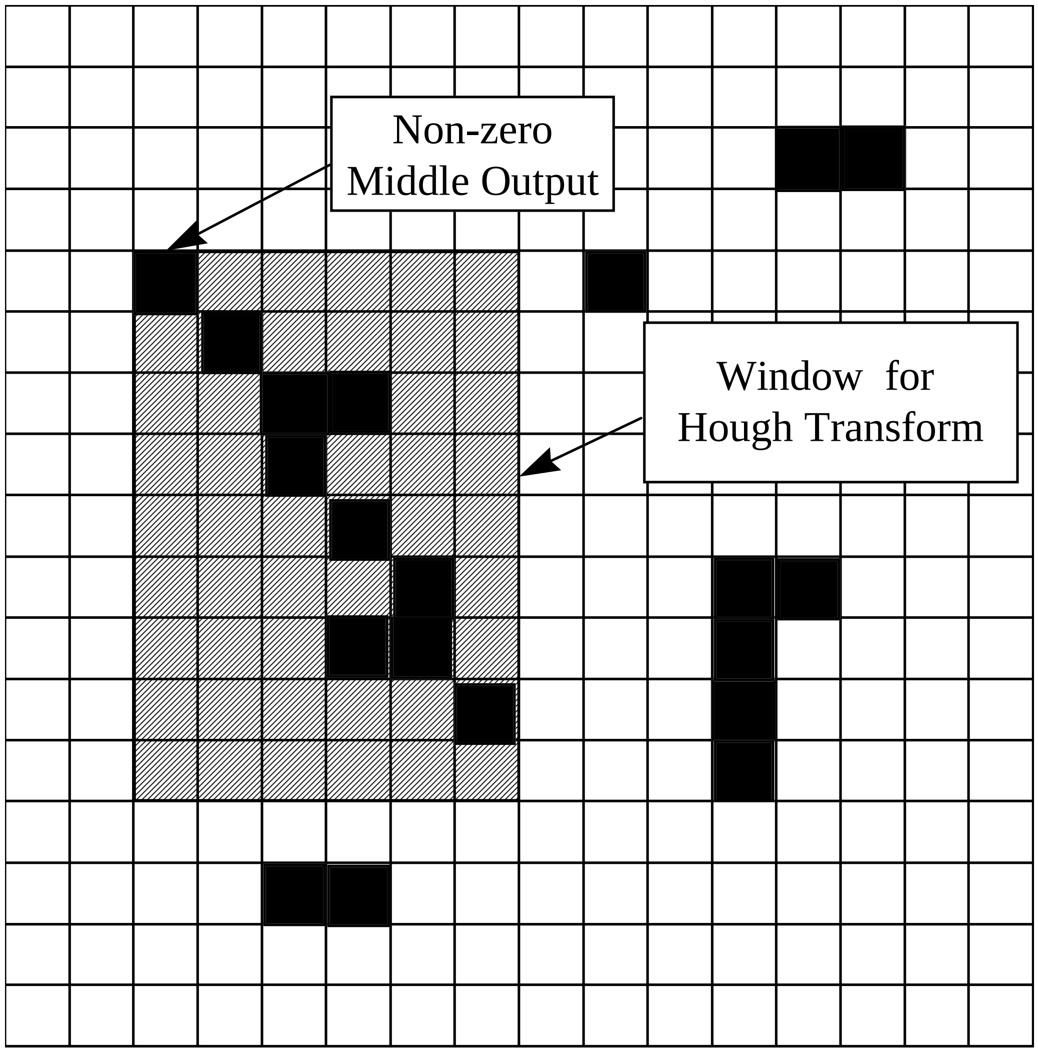


Fig. 3.5. Schematic of cutoff window for the Hough transform (the shaded area). The dark blocks indicate pixels with non-zero middle outputs.