Chapter 4

ALGORITHM DEVELOPMENT

In this chapter, the development of a detection algorithm will be presented. First both correlation and Gabor filters were evaluated for the problem of detonator detection, and their performances were compared. Second the Hough transform was tested for line detection on the output images from correlation-based filters, and finally an information fusion technique was employed to combine the output from the Hough transform with those from correlation filters and to judge the presence of a detonator.

4.1. Gabor Filter

As discussed in Chapter 3, the real part of a Gabor function (RGF) can be used as a blob detector, and the imaginary part of a Gabor function (IGF) can be used as an edge detector. The IGF is not applicable to this problem due to the elongated shape of the detonator; hence only the RGF along the orientation of the detonator is tested.

From Chapter 3, we know that a RGF can be formulated as:

$$g_r(x) = \frac{w}{2\pi|A|^{-\frac{1}{2}}}\cos\left(\omega_0^T\left(x - x_0\right)\right)\exp\left[-\frac{1}{2}\left(x - x_0\right)^T A^{-1}\left(x - x_0\right)\right] \tag{4.1}$$

and its output automatically gives a peak at the center of an object. The four parameters controlling the GFs can be collectively described by a vector $n = \left[a\ b\ w\ x_0^T\right]^T$. By selecting different *n*, a set of RGFs along a certain orientation are generated.

Fig. 4.1 shows an original input image and Fig. 4.2 through Fig. 4.5 show the RGFs with different values for *a*, and the corresponding output results. The size of the filter is $(2a+1)\times(2b+1)$ (in pixels). Due to the elongated shape of the detonator, the left end and the right end information has only a slight effect on the output result. In these results, an object with a similar shape and orientation as the detonator but with a different length, a clothes-hanger, is included in the luggage. It was found that no matter how we changed the parameter *a* , both the hanger and the detonator resulted in peak points, and the clothes-hanger had a even higher peak value than the detonator. Hence it is difficult to distinguish the hanger and the detonator even after threshholding.

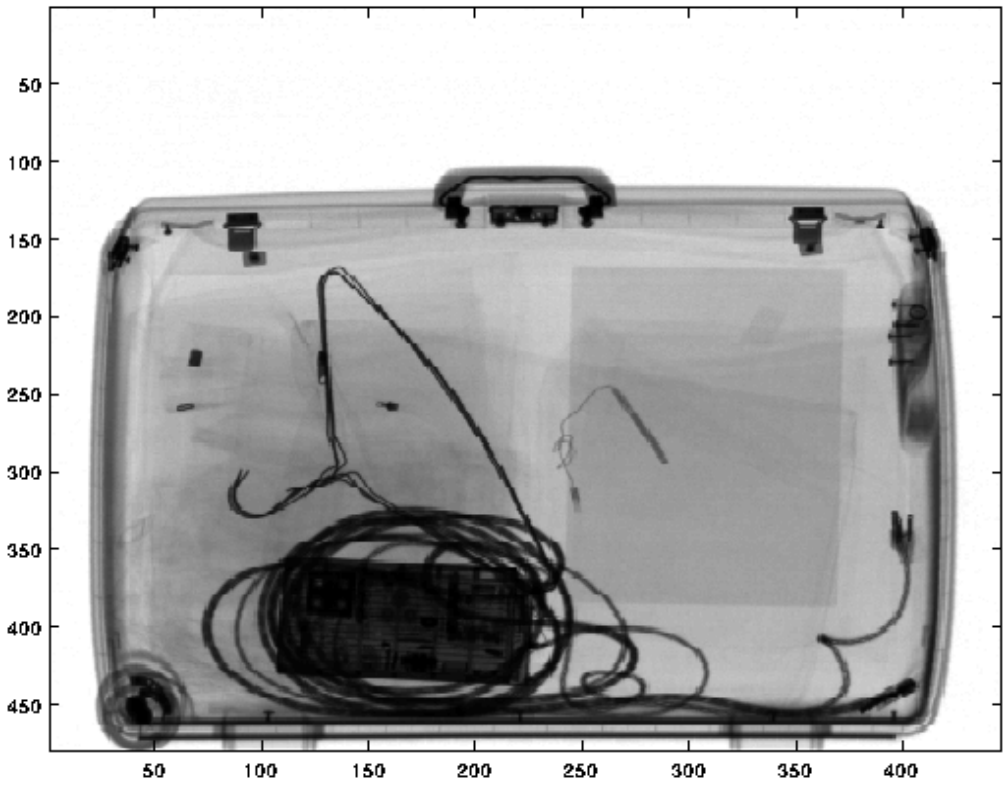
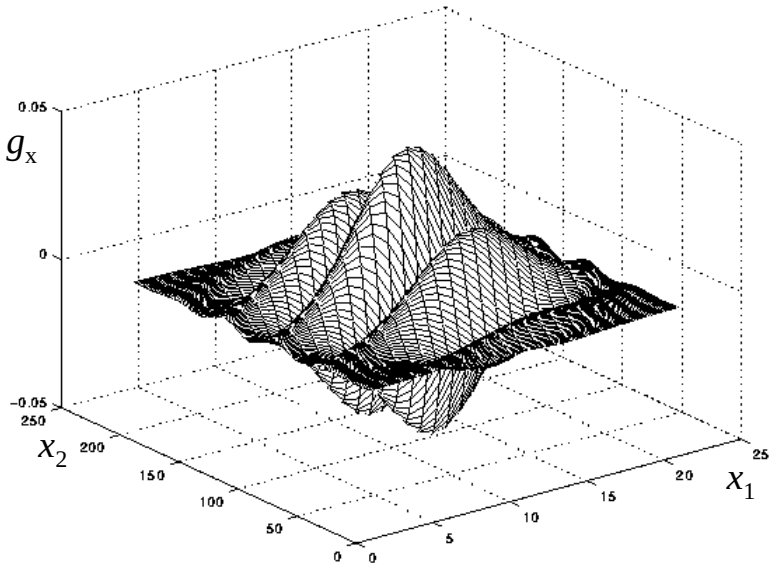
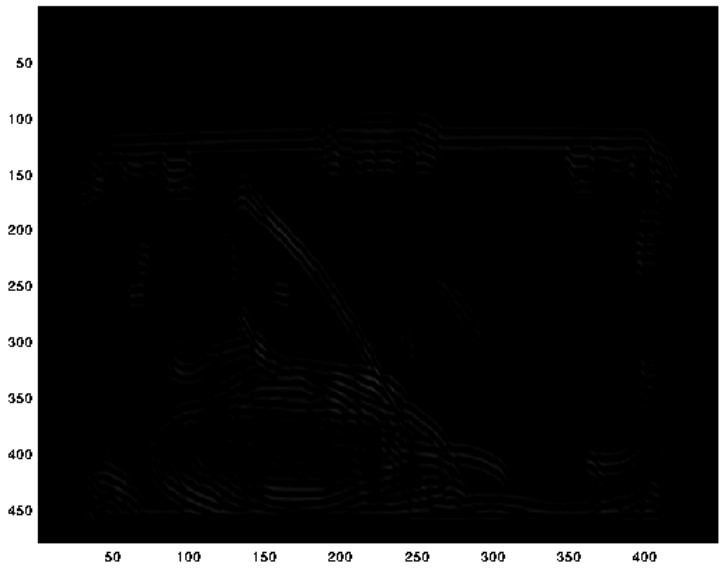


Fig. 4.1. An original image with the detonator at row 250 and column 250.

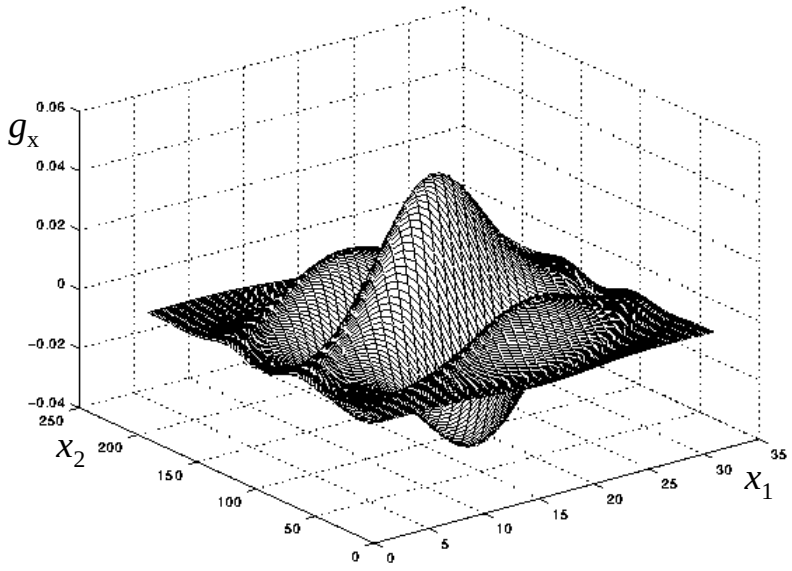


(*a*)

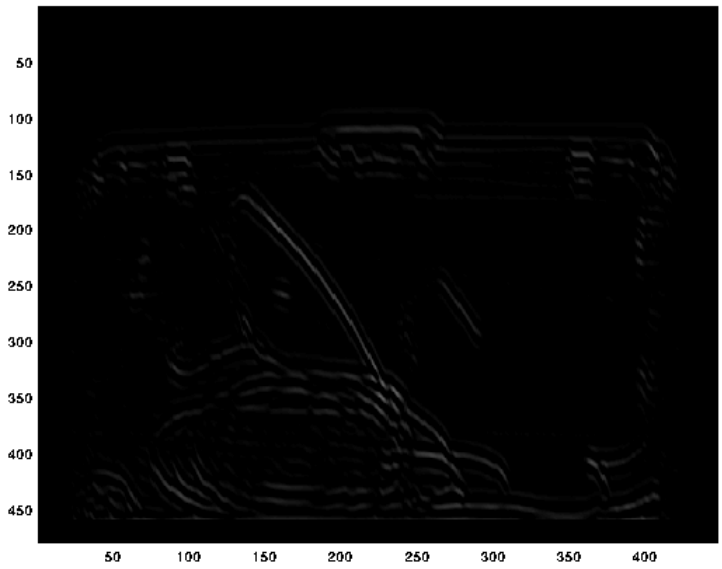


(*b*)

Fig. 4.2. The RGF with $a = 2$, $b = 20$, $\phi = 25.4°$ and the corresponding output result.

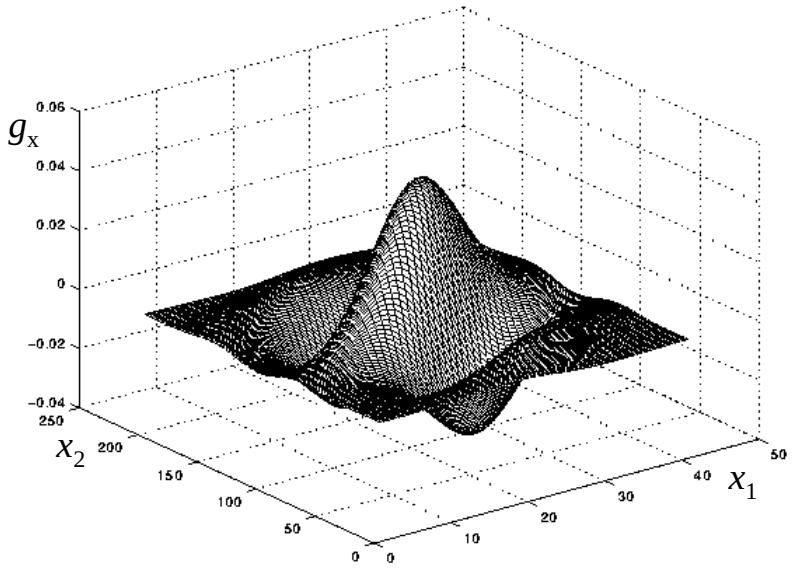


(*a*)

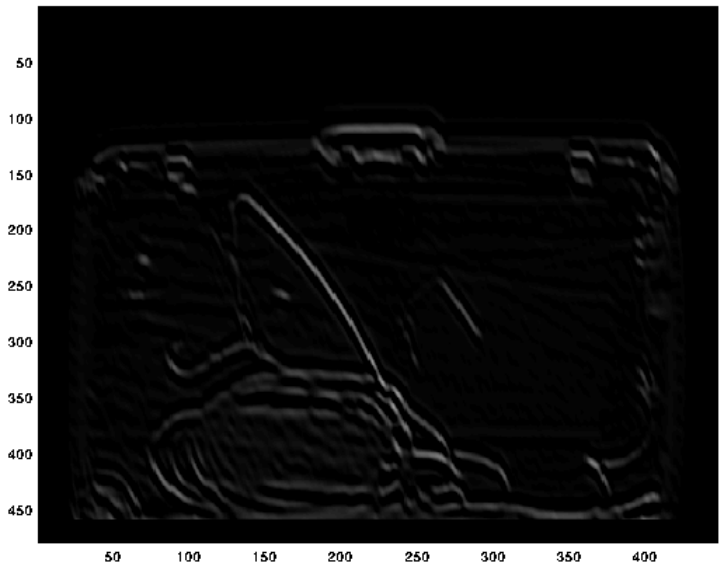


(*b*)

Fig. 4.3. The RGF with $a = 3$, $b = 20$, $\phi = 25.4°$ and the corresponding output result.

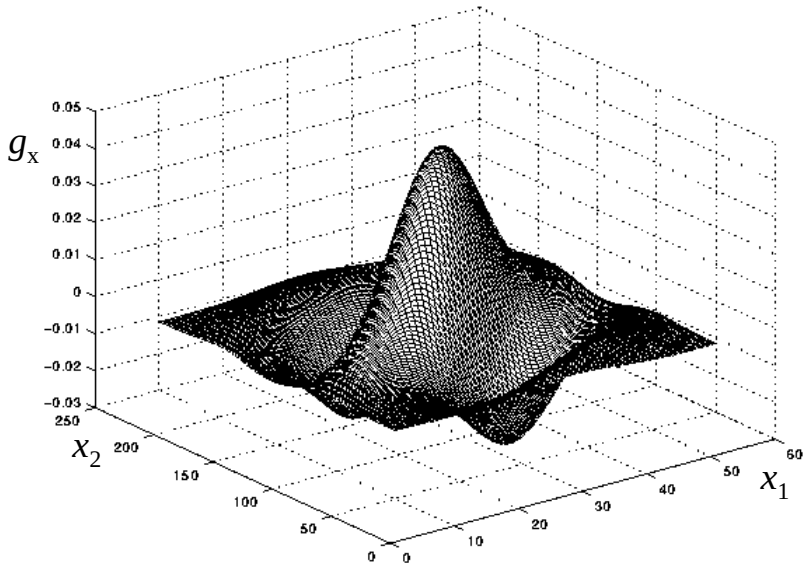


(*a*)

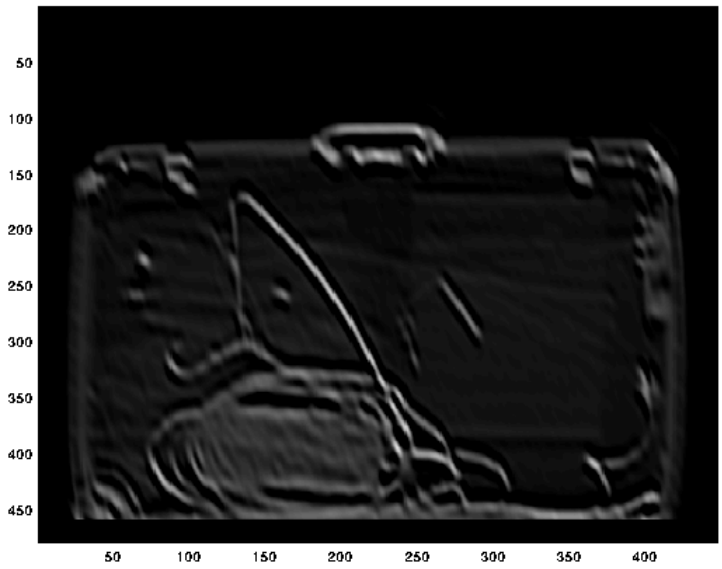


(*b*)

Fig. 4.4. The RGF with $a = 4$, $b = 20$, $\phi = 25.4°$ and the corresponding output result.



(*a*)



(*b*)

Fig. 4.5. The RGF with $a = 5$, $b = 20$, $\phi = 25.4°$ and the corresponding output result.

By changing ϕ and keeping a and b constant, the corresponding output results are shown in Fig. 4.6 through Fig. 4.8. Still using Fig. 4.1 as the input image (the orientation of the detonator is set at 25.4°), Fig. 4.6 shows the result by using the RGF with ϕ set to match the detonator orientation ϕ = 25.4°. It can be seen that the peak points are present near the detonator. After a slight change in ϕ ($\Delta\phi$ = 5°), however, it becomes difficult to see the peak points of the detonator (Fig. 4.7). When $\Delta\phi$ is further increased to 10°, the peak points at the location of the detonator can barely be seen (Fig. 4.8). Obviously, the RGF is very sensitive to the orientation of the detonator.

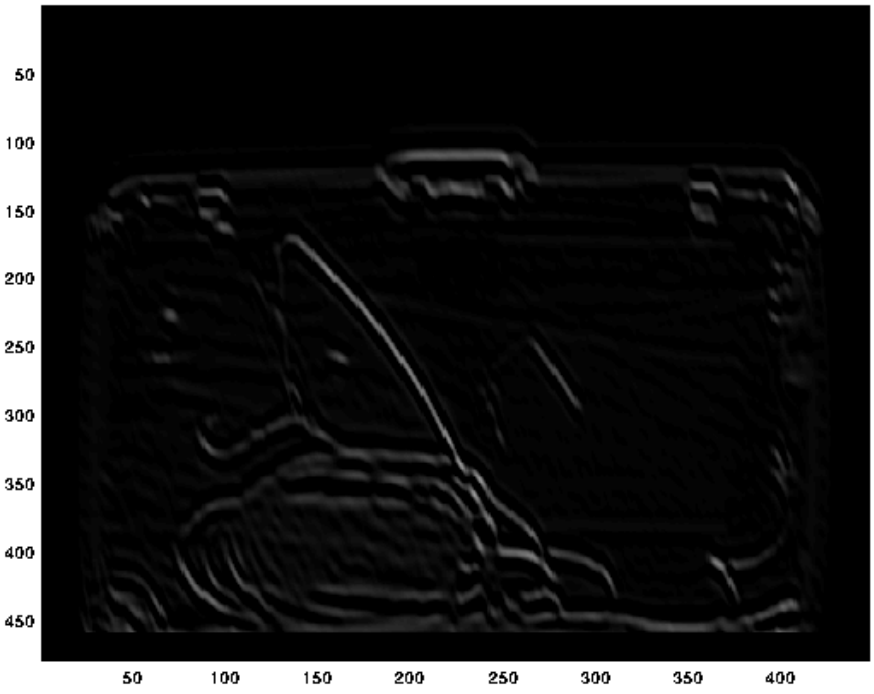


Fig. 4.6. The corresponding output result of the RGF with a = 4, b = 20, ϕ = 25.4°.

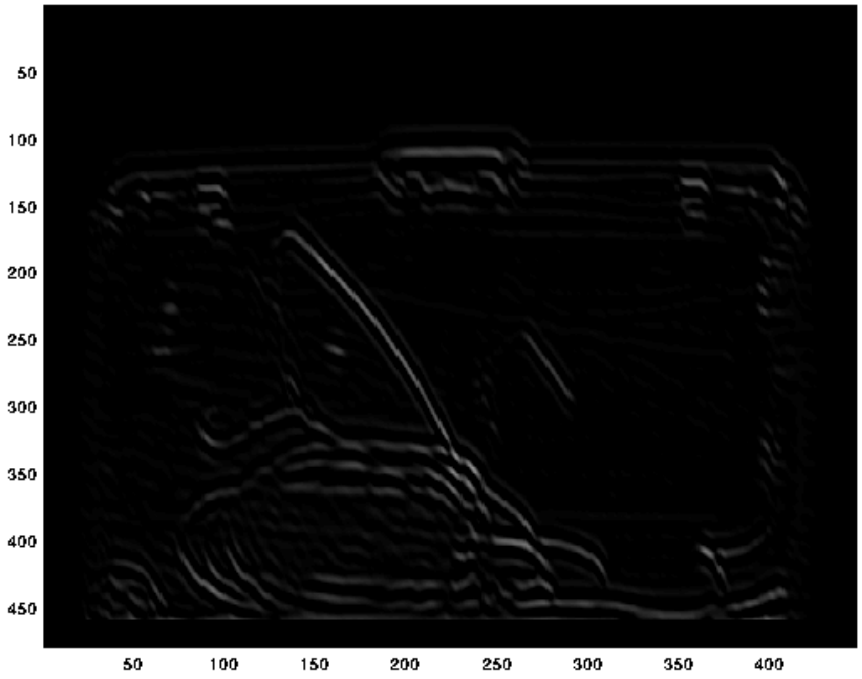


Fig. 4.7. The corresponding output result of the RGF with a = 4, b = 20, ϕ = 30.4°.

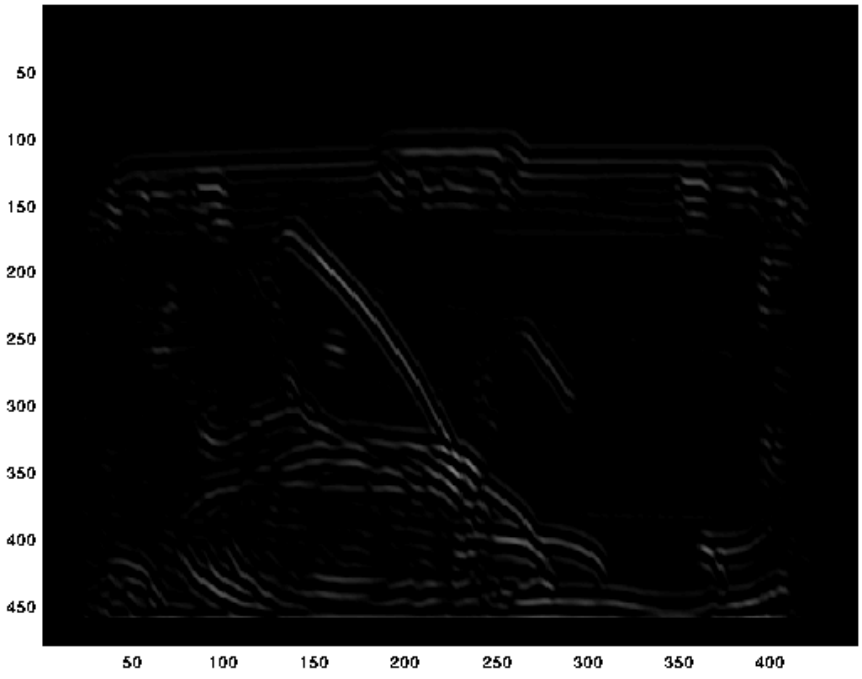


Fig. 4.8. The corresponding output result of the RGF with a=4, b=20, ϕ =35.4°.

4.2. Correlation-based Detection

In this section, the correlation filters are evaluated for the detection of detonators.

4.2.1. Select the Template

Three templates were used to detect the middle and end parts of the detonator correspondingly instead of one whole detonator. Obviously, by using one template with the same size of the detonator, the detonator can be detected in a single step. However due to the elongated shape of the detonator, the detection is vulnerable to changes in the object orientation. A small variation in the object orientation will cause the template matching to fail. Also, because of the size of the detonator, the calculation speed is slow.

To solve these problems, three templates were used to detect the middle-part, left-end and right-end parts respectively. Compared with the size of the detonator, the sizes of these three templates were reduced significantly. At $\phi = 40°$ orientation, the size of the detonator was about 32×23 pixels, while the sizes of these three templates were chosen to be around 10×10 pixels. This reduces the amount of calculation significantly. Another advantage of using three templates is that it increases the tolerance for small orientation changes. This is because that now we are looking for middle and two end points rather than the whole detonator. In the following section, it can be seen that for the middle point template, when the orientation changes between +16.6° to –12.6°, the filter gives good output results. For left- and right-end point templates, the end points are also detected well for the same amount of orientation change.

When the size of the three templates decreases, both the calculation speed and the angle tolerance increase. But the drawback is that false matches also increase. We optimized the size of the template by carefully observing the results obtained while changing the template sizes. For the original image shown in Fig. 4.9, the outputs of the middle-point templates with different sizes are shown in Fig. 4.10. The output of the template with the size of 4×6 pixels gives more false points than the output of the template with the size of 9×11 pixels, i.e., false match points increase with reduced template sizes.

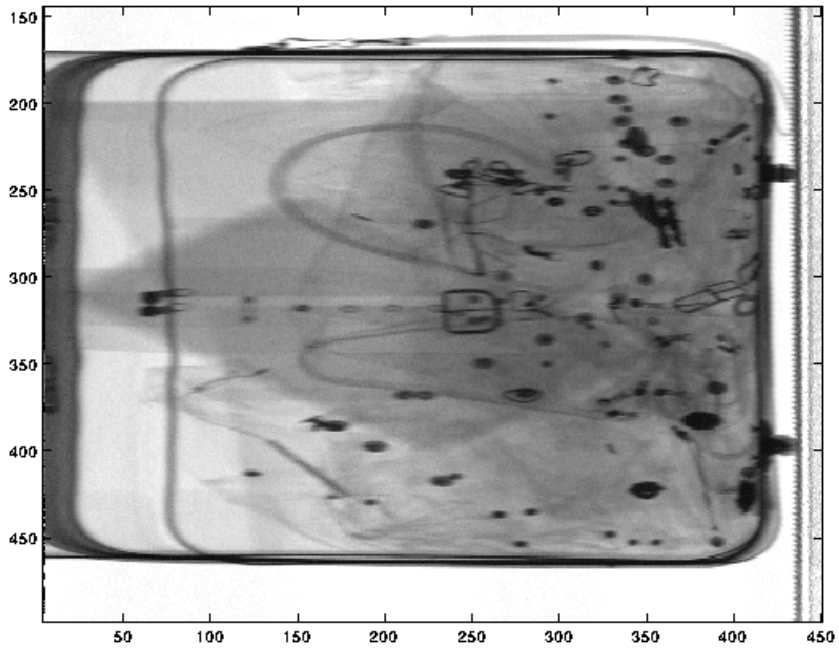
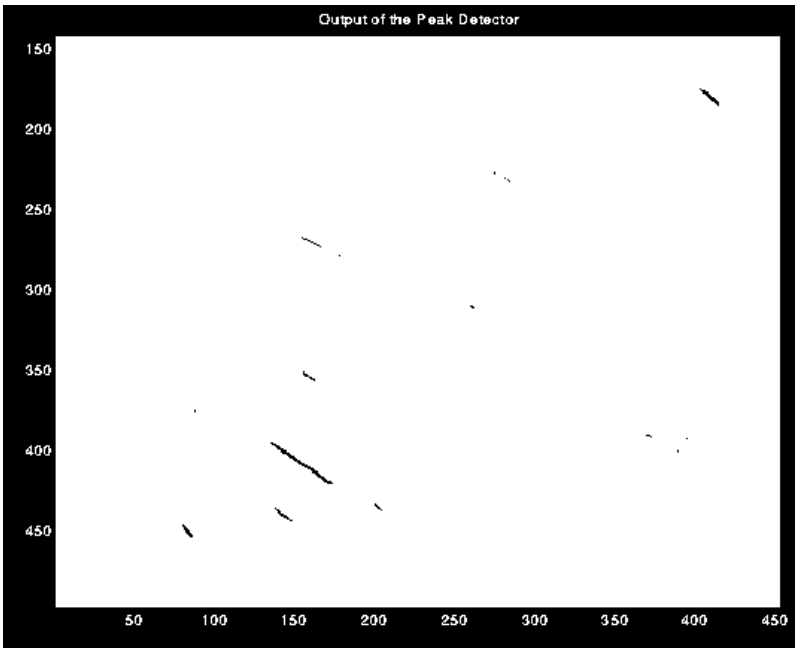
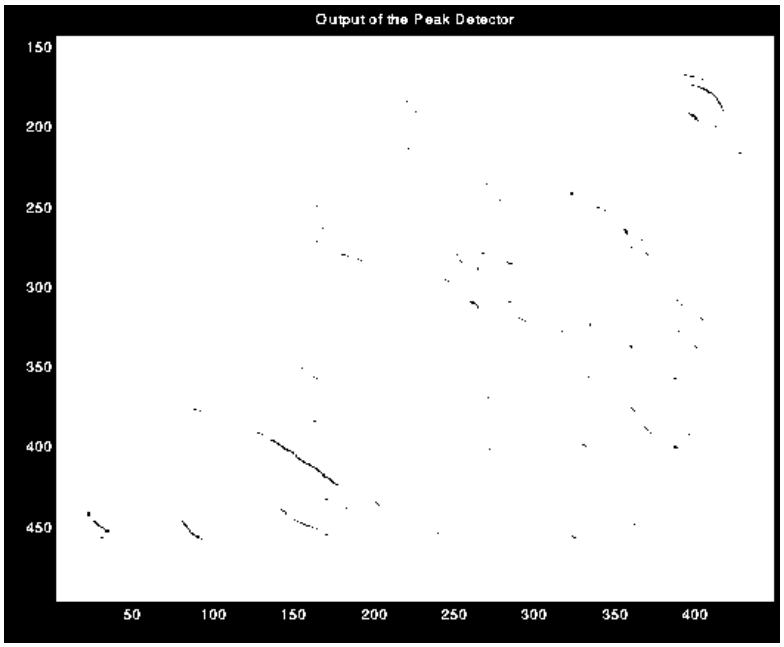


Fig. 4.9. An original image with the detonator around row 400 and column 140.



(*a*)



(*b*)

Fig. 4.10. The output results of the middle-point templates with different sizes: (*a*) 9×11 pixels. (*b*) 4×6 pixels.

4.2.2. Template Construction

Analytical as well as empirical models can be used to describe a template. Normally, the analytical model has some advantages. It is easier to design than the empirical model, and it is also convenient to design the templates with varying orientations.

We tried to formulate the template by using a cylindrical model. The function of the cylinder in Cartesian coordinates is

$$y'^2 + z'^2 = r^2 \tag{4.2}$$

where r is the radius of the column.

By using the x-y plane rotation,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \& \quad z' = z \tag{4.3}$$

where θ is the orientation of the column corresponding to the x-axis, we get $\left(-x\sin\theta + y\cos\theta\right)^2 + z'^2 = r^2$. Since in our problem z is corresponding to the gray scale values, I is used instead of z.

So the function for the analytical model is shown below:

$$I(x,y) = \begin{cases} \left| k\sqrt{r^2 - \left(-x\sin\theta + y\cos\theta\right)^2} \right| & , \ for\, r^2 > \left(-x\sin\theta + y\cos\theta\right)^2 \\ 0 & , \ otherwise \end{cases} \tag{4.4}$$

where k is a constant. The middle-point template designed using this model is shown in Fig. 4.11 and Fig. 4.12*b* shows the output result using this template with Fig. 4.12*a* as the original image. To compare the analytical template with the empirical template, the empirical template for the middle-point template is shown in Fig. 4.13, and the corresponding output result is shown in Fig. 4.12*c*. It can be seen that both analytical and empirical templates give

good output results. However, we had built all the templates for multiple orientations before the analytic template model was considered, hence the empirical template is used in this thesis.

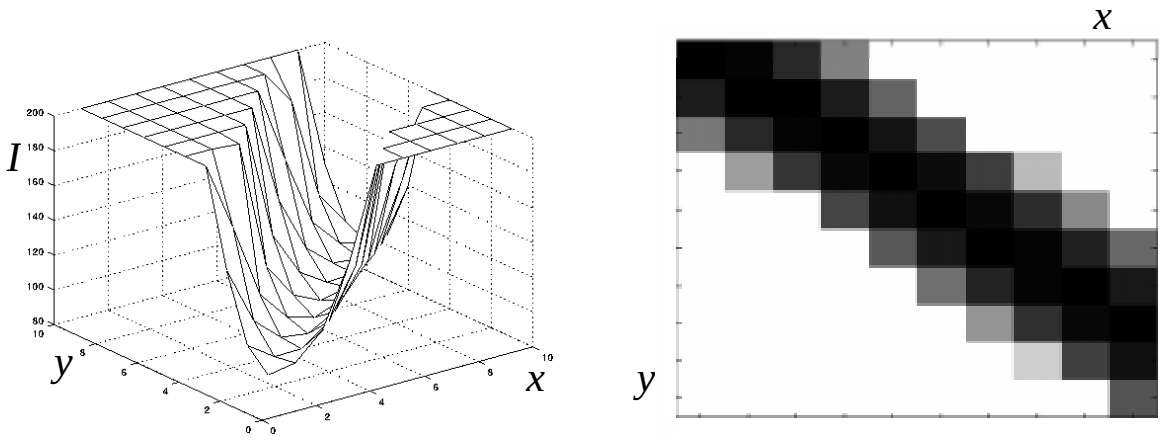
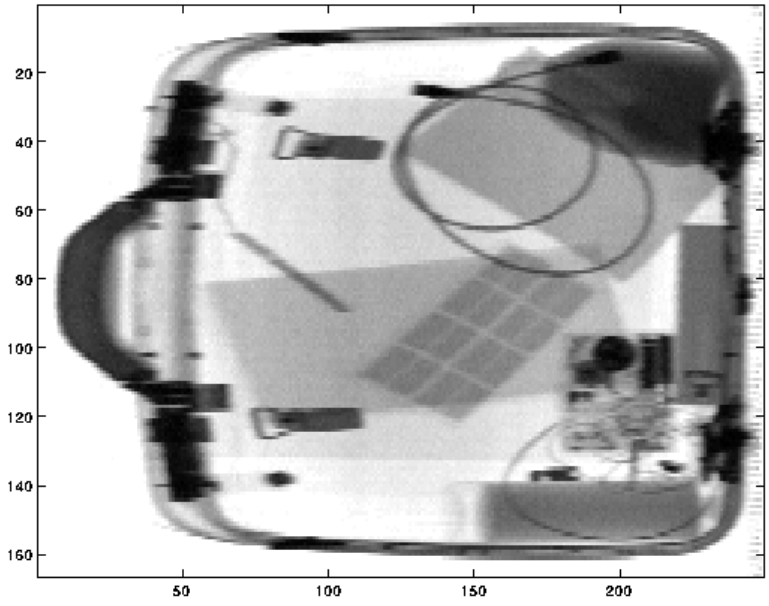
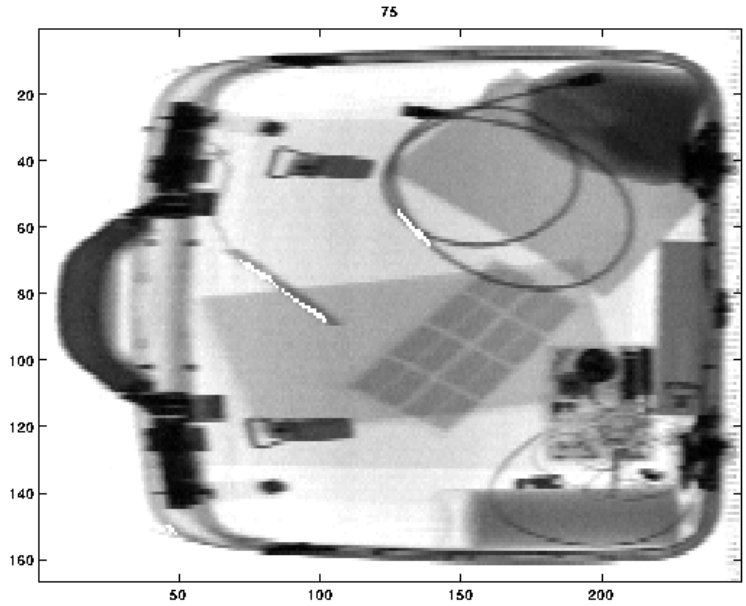


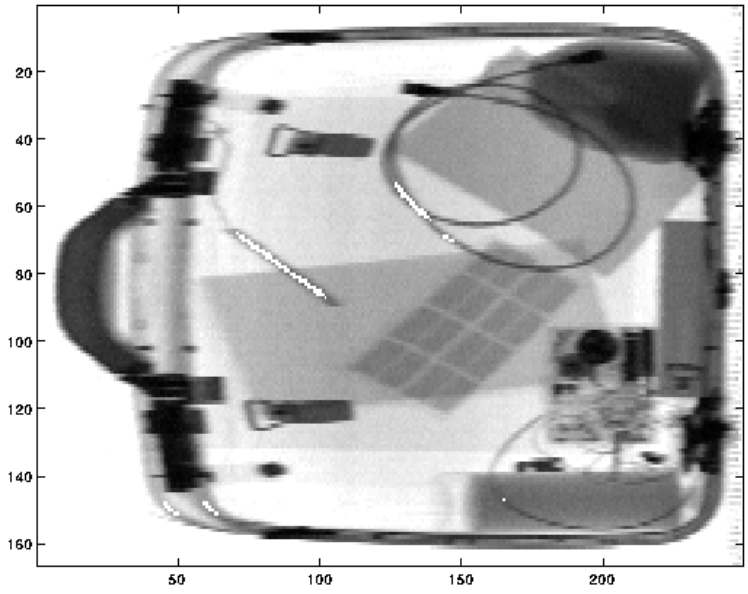
Fig. 4.11. The analytical middle-point template. The size of the template is 10×10.



(*a*)



(*b*)



(*c*)

Fig. 4.12. Comparison of analytical and empirical templates in matched filtering. (*a*) Original image. (*b*) The output obtained using the analytical template. (*c*) The output obtained using the empirical template. The same threshold was used (0.75). The empirical result is slightly better in this case.

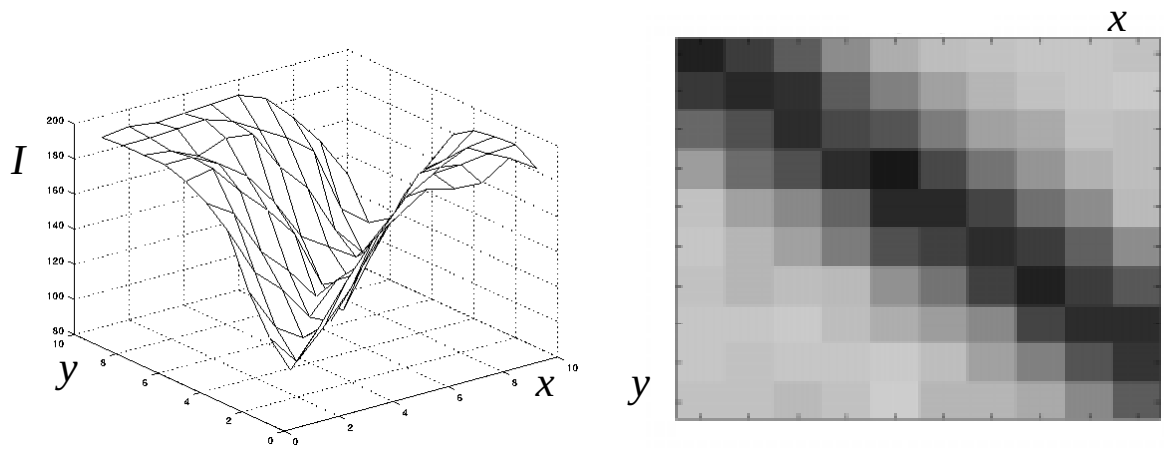
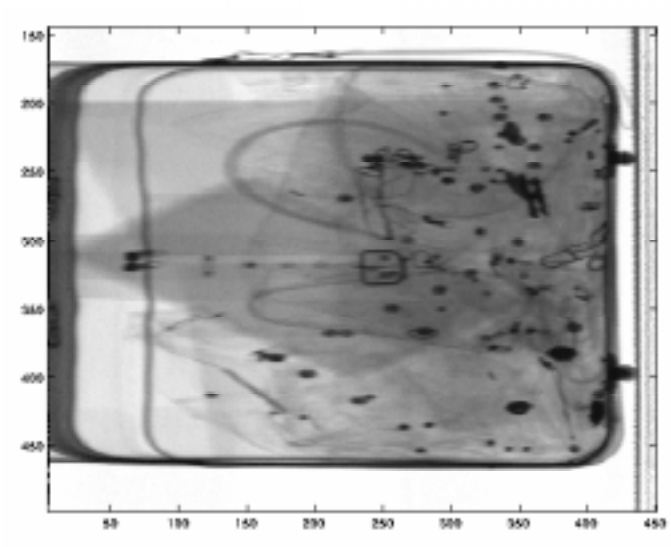


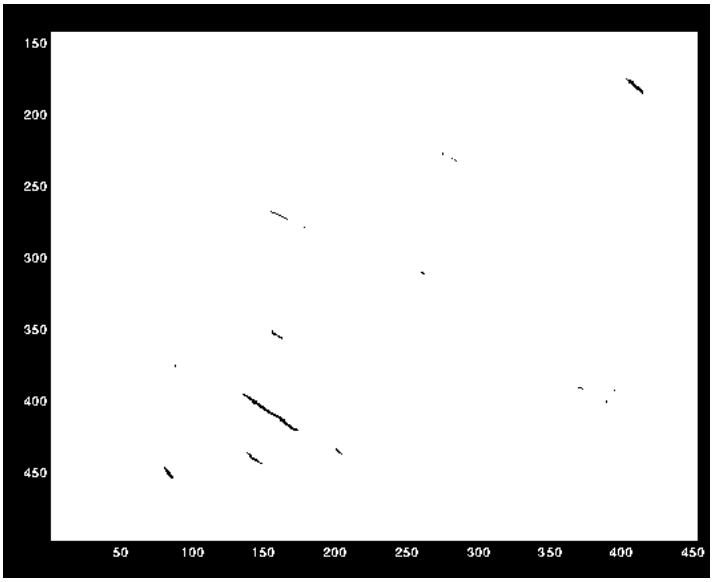
Fig. 4.13. The empirical middle-point template. The size of the template is 10×10.

4.2.3. Select the Threshold

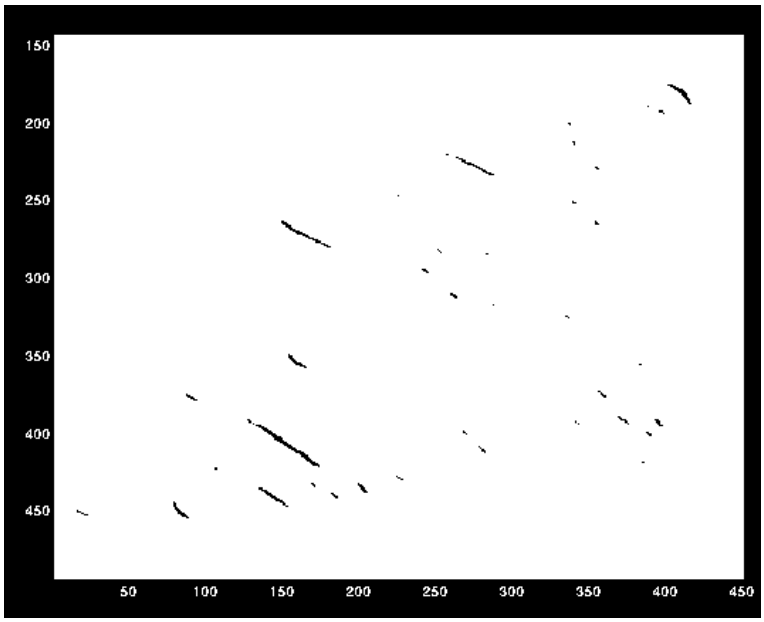
After the correlation filter, the proper threshold needs to be selected in order to get the binary result. From Chapter 3, we know that the output of the correlation is between -1 and 1, and a larger value indicates a better matching. Consequently, a larger value for the threshold can reduce false outputs, but the angle tolerance is also reduced. On the other hand, a smaller value for the threshold can increase the angle tolerance, but false outputs also increase. Fig. 4.14 shows the output of the middle-point templates with thresholds 0.75 and 0.65. A appropriate threshold was determined through careful evaluation. For 40^0 orientation, the threshold for middle point detection was determined to be 0.80, and end point detection 0.85. When the correlation result was larger or equal to the threshold, the corresponding pixel on the output image was set to 1. Otherwise, it was set to 0.



(*a*)



(*b*)



(*c*)

Fig. 4.14. Example of the output of the middle-point templates with different thresholds. (*a*) The original image. (*b*) Output result with threshold 0.75. (*c*) Output result with threshold 0.65.

4.3. Hough Transform

We use the Hough transform $r = x\cos(\theta) + y\sin(\theta)$ as discussed in Section 3.4. Notice that we only calculate the Hough transform for a certain range of the angle θ. Say, if the three peak and end templates are set at the orientation $\phi = 40°$, then the range of the angle is from 24° to 54°. The reason for this is that the middle-point and end templates only tolerate about 25° change. Computing the Hough transform beyond this range is not necessary because of the assumed fixed orientation of the detonator. In other words, this algorithm will be effective only when the change of the detonator orientation is less than 25°.

Once a non-zero point is found, the Hough transform is carried out within a certain size window. We do not simply apply the Hough transform to the whole middle-point output

plane, because a certain amount of separated middle points that are located along a common line will cause a false match.

There is another simple method to count the number of the middle points along the detonator orientation in a certain size window. A binary template that has 1s along the detonator orientation and 0s on other pixels (Fig. 4.15) was built. The drawback of this method is that when the orientation tolerance is increased, false matches increase considerably. For example, when the template size is 20×20 pixels, and a orientation tolerance between 40°-50° is preferred, the number of the pixels that have a value of 1 must be large than 7. No matter how the middle points are located, as long as enough points (i.e. larger than the threshold) show up on those 1 pixels on the template, we will get a suspected detonator point. Comparing with this simple method, the Hough transform increases the orientation tolerance and reduces false matches dramatically.

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Fig. 4.15. Illustration of a binary template with 1*s* along the detonator orientation and 0*s* on other pixels

4.4. Image Fusion

Let $(\tilde{x},\tilde{y})$ be the origin (upper-left corner) of the current Hough transform window. 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. In the fusion step, we use the following rule: when the maximum value in the Hough array is larger than a given threshold, and at least one end point is detected near either

point $(\tilde{x},\tilde{y})$ or $(\tilde{x}+x_r,\tilde{y}+y_r)$, where ($x_r$, y_r) indicates the right end of the line in the current Hough window, we say that the detonator is detected.

To set the threshold for the Hough transform, we should notice that if the threshold is set based on the exact length of the detonator, since the overlap will reduce the number of the non-zero pixels in the middle output, a slight overlap will cause the detonator not to be detected. However, if the threshold is set at much lower than the length of the detonator, the tolerance for overlapping is improved but false matches will increase. Consequently, the threshold should be decided after careful observation.

Since each non-zero pixel in the middle output will be set as the original of the Hough transform, by assuming that the left end is around the original, the detection for the left end is straightforward. The pixel around the origin of the current window $(\tilde{x},\tilde{y})$ is detected to see whether there are some end points showing up in this area, under the assumption that the line, it presents, passes through $(\tilde{x},\tilde{y})$.

In order to detect the right end, we assume that once the maximum value in the Hough array is larger than the threshold, in other words, when a line is detected, there are little or no middle points that are not on this line. By using this assumption, in the current window, the average row location and the average column location of the middle points should be the middle of the line. Since the left end is the origin of the current window, the location of the right end point can be easily extrapolated, say, (x_r, y_r). Then the right end point will only be searched around (x_r, y_r). If one or more end points are found, a match will be declared.

Mathematically, the image fusion that was developed in this research may be expressed as follows:

Classify pixel (x, y) as "detonator" if

$$(I \otimes h_1)(x,y) > T_1$$

AND

$$\left[(I \otimes h_2)(x',y') > T_2 \text{, for any } (x', y') \text{ satisfying } (x'-x)^2 + (y'-y)^2 < R_2^2\right.$$

OR

$$\left.(I \otimes h_3)(x',y') > T_3 \text{, for any } (x', y') \text{ satisfying } (x'-x_r)^2 + (y'-y_r)^2 < R_3^2\right]$$

AND

$$H(x,y) > T_4,$$

Otherwise, classify (x, y) as "non-threat".

In this expression, T_1, T_2= T_3, and T_4 are thresholds, R_2 and R_3 are distance thresholds, and $H(x, y)$ represents the maximum value in the Hough array, computed with (x, y) as the upper left corner of an image window.