Chapter 1

Trip Wire Detection for Land Mines

Chris Jessop, *University of Calgary* (cjessop@acs.ucalgary.ca)
Shabnam Kavousian, *Simon Fraser University* (skavousi@math.sfu.ca)
Michael Lamoureux, *University of Calgary* (mikel@math.ucalgary.ca)
Joshua Madden, *University of British Columbia* (jmadden@iam.ubc.ca)
Edo Nyland, *University of Alberta* (edo@phys.ualberta.ca)
Bruce Rout, *Simon Fraser University* (routfam@island.net)
Arunas Salkauskas, *University of Calgary* (arunas@math.ucalgary.ca)
David Saunders, *University of Toronto* (saunders@math.toronto.edu)
Bill Vetter, *Calgary* (vetterb@cadvision.com)
Satoshi Tomoda, *University of Calgary* (tomoda@math.ucalgary.ca)

Written by
Chris Budd, *University of Bath* (cjb@maths.bath.ac.uk)
John Stockie, *Simon Fraser University* (jms@cs.sfu.ca)

1.1 Introduction.

The safe detection of anti-personnel land mines is essential for the recovery of nations after an armed conflict. Land mines are often left as booby traps in populated areas, and so it is not appropriate to simply blow them up. Rather, the land mines need to be detected and then deactivated. A common mode of setting triggers for anti-personnel land mines is with trip wires which are hidden behind foliage or camouflaged in some other way. In order to minimise the risk to the people involved, a method is needed to identify suspect wires quickly and accurately. There are also many other practical situations where it is important to be able to detect wires of this sort; for example, helping light aircraft to avoid power cables.

The method of land mine detection being considered by ITRES Research Ltd. (who posed the problem for the PIMS Workshop) is to mount a detecting camera on a boom ahead of a slowly moving truck. The camera looks vertically downward at the ground beneath the boom and uses a charge-coupled device (CCD) detector to survey the resulting image[a]. In this detector, the image is acquired as a sequence of lines, which are constantly updated as the truck moves forward. The image is considered in several illuminating wavelengths simultaneously, spanning the visual and infrared spectra. Although the trip wires are often purposely camouflaged by being made of an optically transparent material, it is reasonable to assume that they are opaque in at least one of the scanned wavelengths. It is then hoped that an automatic algorithm (running in a well-protected computer on the truck) can be used to find the trip wires appearing in the images. If any wire (or suspect wire) is detected, then the algorithm will sound an alarm which will be a signal to the human operator to halt the truck and initiate a response by mine disposal experts.

1.2 Detailed Problem Description.

The trip wire detection problem can be reduced to the following:

> *Can a computer algorithm be designed that will detect a trip wire in a single image, assuming that the wire can be recognised with the naked eye?*

A typical image with sevreal wires hidden in foliage is given in Figure 1.1. In this picture there are two trip wires: one dark, horizontal wire located just below the centre of the image; and a second light-coloured one, obscured by the foliage, and running at an angle of approximately 30° to the first. We can see immediately some of the difficulties inherent in detecting a trip wire, namely

- wires can be partially covered by foliage;
- wires are not uniformly illuminated;
- many features, such as the stems of leaves, are also linear in nature and so might easily be mistaken for a wire;
- the image can be blurred or noisy, either making the wires difficult to detect, or introducing spurious linear features (such as the dark, grainy feature at the upper right of the image).

The problem faced in this Workshop was to develop an algorithm for detecting a trip wire which satisfied the following conditions:

1. it is *reliable*, and can detect a wire partially hidden and non-uniformly illuminated,

[a]More information about this type of detector and its applications is available at ITRES' web site – http://www.itres.com/.

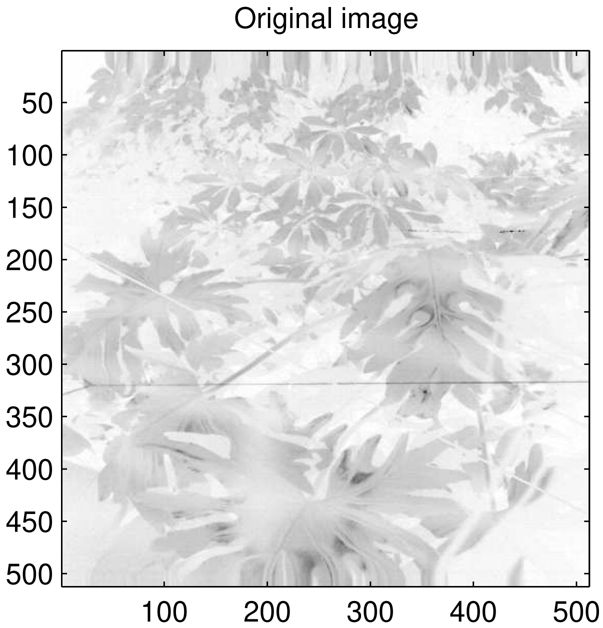


Figure 1.1: A test image provided by ITRES containing two trip wires and a third suspect linear feature.

2. it is *fast* and can process the image and detect the wire in real-time, before the advancing truck runs over the mine,

3. it has a low rate of *false detections* so that it does not get confused between trip wires and leaf stems (or other naturally-occurring linear features).

An algorithmic approach which has already been investigated by ITRES is one based on the *Hough transform*, which is a common technique used in image processing for detecting lines in images [3]. However, this method has apparently not been very satisfactory.

During the Workshop, we considered three algorithms for detecting trip wires:

- The first used a generalisation of the Hough transform called the *Radon transform*. To enhance the detection capability of this method, the image was first pre-processed in a way that exploits various properties of wires (which we discuss in Section 1.3.1). This method proved reasonably successful and the key aspect of implementing it turned out to be the development of an image-independent thresholding condition to distinguish between linear and nonlinear features.

- The second algorithm was based upon taking a series of one-dimensional slices through the image and looking for the movement of a sharp feature across this slice as the position of the slice was changed. This method was implemented, but was found to be very much affected by external clutter (such as foliage) and much less effective than the Radon transform method.

- The final approach was similar to the previous one, but attempted to take advantage of the fact that the CCD detector acquires the image in thin, horizontal strips. By looking at a sequence of contiguous strips, we attempted to locate the trip wire by looking for coherence from one slice to the next. This method suffered from the same problems as the previous one.

In this report, we will concentrate primarily on the Radon transform method, since it was the most successful of the three. When tested on several images, some supplied by ITRES and other generated artificially during the Workshop, it proved fairly effective in detecting hidden wires. The algorithm was implemented in MATLAB and proved rather slow. However, we believe that our implementation is far from optimal (indeed, there is a difference between what MATLAB claims it can do from the manual and what it does in practice!), and a dedicated implementation will certainly run *much* faster. We also believe that there is potential for a much faster implementation of the Radon transform that makes use of the fact that the image is updated one line at a time.

1.3 Identifying Linear Features using the Radon Transform.

Suppose that we look at a two-dimensional image which is made up of pixels. The image intensity can be regarded as a function $u(x, y)$ of the position (x, y) in the image. Before describing the Radon transform and the algorithm based on it, we first need to define what is meant by a trip wire, or a *linear image feature*, in a more mathematical way.

1.3.1 What is a wire?

We identify three characteristics of the intensity function $u(x, y)$ which make a wire distinct from the surrounding background:

1. A wire is *narrow* and represents a portion of the image in which the function $u(x, y)$ has a *high curvature*; that is, where $u(x, y)$ changes very rapidly.

2. A wire is locally straight and presents a *sharp edge*.

3. A wire can be partially hidden by shade, foliage or other natural obstacles, and as such it consists of *a sequence of co-linear line segments*, rather than an uninterrupted, straight line.

Although one or two of these properties may be satisfied by other natural features such as the stem of a leaf, the combination of the three appears to define a wire uniquely.

1.3.2 The Radon Transform.

The Radon transform of a 2D image $u(x, y)$ is commonly written as [7]

$$U(\rho, \theta) = \int_{\mathbb{R}^2} u(x, y) \cdot \delta(x \cos\theta + y \sin\theta - \rho)\, dx\, dy \tag{1.1}$$

with ρ and θ defined as shown in Figure 1.2 We assume that the origin for the measurement of ρ is always chosen to be the centre of the image. The advantage of using the Radon transform for the

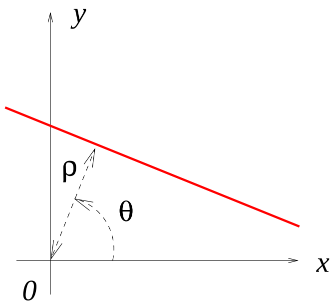


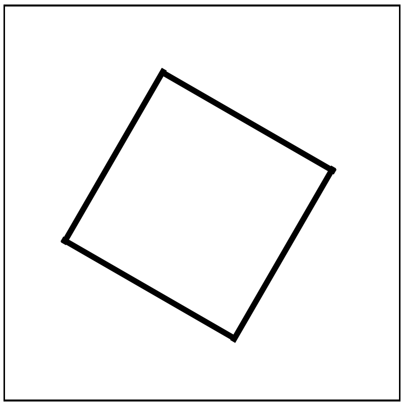
Figure 1.2: Definition of ρ and θ in the Radon transform.

detection of a trip wire is that a linear feature in the image, approximated by the straight line with equation

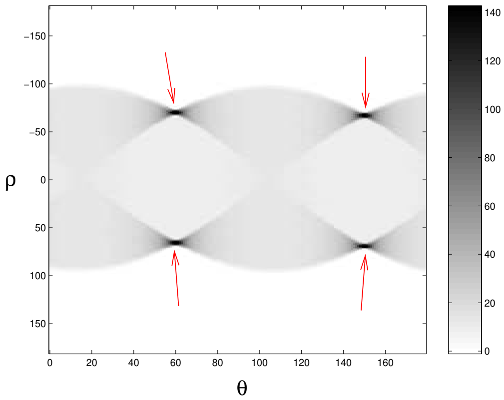
$$x \cos \theta + y \sin \theta = \rho,$$

corresponds to a point where $U(\rho, \theta)$ is large. Furthermore, the integral operation in (1.1) is insensitive to whether the ray along which the integration is performed has a linear feature that is contiguous or broken. The maximum value of $U(\rho, \theta)$ should thus correspond to the "most significant" linear feature in the image.

As an example, consider the simple test image in Figure 1.3(a), consisting of four line segments arranged in the shape of a square. The Radon transform clearly indicates four peaks corresponding



(a) Test image.



(b) Radon transform.

Figure 1.3: A simple test image and its Radon transform.

the four sides of the square: two at an angle of $\theta = 60°$ (with approximate offsets $\rho = 65$ and -65 from the centre of the image) and two more at an angle of $\theta = 150°$ (with the same offsets).

1.3.3 The algorithm.

The Radon transform is well-known for its ability to pinpoint linear features even in very noisy images [6, 2]. However, for the images supplied by ITRES, we found that the transformed intensity $U(\rho, \theta)$ often failed to detect real wires, and also exhibited spurious maxima totally unrelated to trip wires that were clearly visible to the naked eye. Therefore, the Radon transform by itself is not appropriate for trip wire detection.

The question now is this: *How can we exploit the three characteristics of a "linear feature" listed in Section 1.3.1?* The Radon transform makes use of only the third item in our list, by identifying multiple segments that lie on a single straight line. We will now describe an algorithm that uses two pre-processing steps which exploit the fact that trip wires are associated with edges, and high image curvature. Starting with a two-dimensional image $u(x, y)$, with values in the range $[0, 1]$, we proceed as follows:

Trip Wire Detection Algorithm.

---

1. Apply a *Laplacian filter*

$$L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

to each point in the raw image, which is analogous to taking the second derivative of the image, and is designed to accentuate regions of high curvature.

2. Perform *edge detection* on the filtered image (we use the *Sobel* method in most cases since it is the default in MATLAB).

3. Apply the *Radon transform.*

4. *Threshold* the transformed image, by looking for all points (ρ, θ) with $U(\rho, \theta) > T$ for some threshold T, in order to differentiate candidate lines from surrounding clutter.

5.* (for testing only) Apply the *inverse Radon transform* to locate candidate trip wires, and then compare them manually with the original image.

Step 5 is not required, but is very useful in terms of testing the success of the algorithm to see which linear features are recognized. In an actual mine detection scenario, a transformed image which has a value that exceeds the threshold T in Step 4 will set off an alarm that causes the human operator to stop and take action. The main conceptual difficulty in implementing this algorithm is the selection of an appropriate threshold, which will be described in detail in the next section.

This algorithm has the advantage of eliminating background intensity variations and bright spots and identifying linear features which may be both noisy and disconnected in the original image. We experimented with leaving out individual steps, but our overwhelming feeling is that steps 1–4 are all required for an effective algorithm.

The algorithm was very straightforward to implement in MATLAB, since all the components (except the "optional" inverse Radon transform) are built-in procedures in MATLAB's *Image Processing Toolbox* (see the Appendix for a listing of the code). We observed that steps 1, 2, 4 and 5 are all reasonably fast and that most of the computational time is taken up by the Radon transform in Step 3. This is partly because the MATLAB implementation is very slow and does not use the very important result that the Radon transform can be computed rapidly by using a Fast Fourier Transform (FFT) [1]. Even though the FFT algorithm is mentioned in the MATLAB documentation [5], it is not implemented in the current version of the *Toolbox* (communications with a technical representative of The MathWorks Inc. have confirmed this fact).

We also implemented the trip wire detection algorithm using another set of routines in the Kansas University Image Processing Library (KUIM) [4], but found them to be equally slow.

1.3.4 Approaches to thresholding.

As mentioned before, the choice of an appropriate threshold in Step 4 of the algorithm on page 6 is of paramount importance. When applying the threshold T, the only points (ρ, θ) retained in the transformed image are those satisfying $U(\rho, \theta) > T$. This produces a relatively small set of points (if T is chosen appropriately), and so inverting the thresholded image to give a set of candidate trip wires is then a fast and easy procedure (indeed it is very much faster than doing a direct inverse Radon transform).

There are several aspects of the thresholding problem that make the choice of T particularly problematic:

- If the threshold level is chosen too high, then no wires are found; alternately, if T is too low then too many false wires will be detected. Thus, there should be some "best" intermediate value in which only true trip wires are detected.

- The Radon transform integral (1.1) can be interpreted in the sense of Riemann sums as summing image intensities along linear slices. Therefore, an image that has bright patches throughout will have much larger transform values than an identical image that has no such patches. Some sort of scaling of the transformed image U based on average or maximum intensity is hence necessary if the linear features in both of these images are to be detected at the same threshold level.

- Equation (1.1) should also be insensitive to the dimensions of the image, and so U may also have to be scaled by some size factor. For example, if a trip wire is detected in an $M \times N$ image, then the same threshold should be sufficient to detect the wire in a smaller sub-image that fully encompasses the wire. In fact, the MATLAB implementation of the Radon transform assumes the pixels have dimension 1 square unit (that is, $\Delta x = \Delta y = 1$), and so a rescaling based on image size *is necessary*.

We now propose several alternatives for scaling the transformed image. Assume for the remainder that the image intensity values $u(x, y)$ lie between 0 and 1[b], and that the image has dimensions $M \times N$ pixels. Our approach to thresholding will be to look for points in a *rescaled* Radon transform $\tilde{U}(\rho, \theta) = U(\rho, \theta)/\alpha_u$ that satisfy $\tilde{U} > T$, and where α_u is a scaling factor that depends on the image. Based on the above observations, we will consider the following choices for the scaling factor:

S-1	no scaling	$\alpha_u = 1$
S-2	scale by the maximum of U	$\alpha_u = \max(U)$
S-3	scale by the average of U	$\alpha_u = \text{mean}(U)$
S-4	scale by the number of pixels	$\alpha_u = M \cdot N$
S-5	scale by the largest image dimension	$\alpha_u = \max(M, N)$

The reason for including **S-5** is that the **S-4** scaling actually serves to distort pixels in non-square images, while **S-5** weights both directions the same regardless of image size. The five different scalings will be applied in the next section to a variety of images. Our hope is to find an "ideal," universal threshold that is independent of image properties, and does not have to be tuned for a particular image.

1.3.5 Results.

At the start of the Workshop, we were provided with a single digital trip wire picture (Image 5), with a well-defined wire passing along its entire length. To help in the testing process, we created several additional images based on the given one, with "artificial wires" having different intensities, orientations and noise levels. One such image, which we label Image 5c, was the one pictured earlier in Figure 1.1. The output of the trip wire detection algorithm for this image is pictured in Figure 1.4.

Scaling S-1, S-2 and S-3:

The original image has three linear features: one an obvious bright trip wire running horizontally across the bottom of the image, one artificially created dark trip wire running obliquely from left to right, and a third noisy horizontal bright wire at the upper right. Using any of these three scaling methods, all of the linear features could be identified. A series of other images were constructed

[b] The image at this stage has already been through the edge detection process, and so is already a binary (0/1) image.

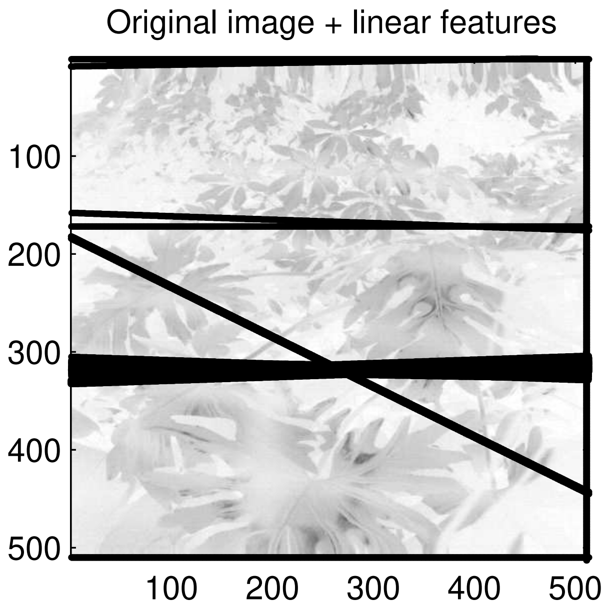


Figure 1.4: Sample inversion for Image 5c, with $T = 2.4 \times 10^{-4}$ and scaling **S-4**. The lines along the edges of the image are a result of the boundary conditions on the Laplace filter, and can be ignored.

manually from this one, that contained other trip wires and/or more noise. Scaling **S-3** produced the best results, but the threshold had to be tuned for each individual image.

Scaling S-4:

Table 1.1 summarises the number of linear features found by the algorithm with scaling **S-4** compared to what we'd like to find (in the "Actual" column). It is clear that the rescaling based on image size seems to be fine for square images of the same size. There is a fairly consistent value of the threshold $T \approx 2.4 \times 10^{-4}$ which identifies the linear features we're looking for.

However, the **S-4** scaling is not adequate when considering non-square images (which are probably more typical of what would be obtained from a CCD camera). This can be seen by looking at the last column, where the maximum value of the scaled Radon transform has significant variation with image strip width N. Therefore, we cannot expect the thresholding in this case to be independent of the image size.

[d] With the addition of noise in these images, the feature in the upper right of the image was much less coherent and thus hard to detect.

[e] This image has a lot of noise compared to the original one, which is what affects the $\max(\tilde{U})$ value. The artificial dark line is not picked out until the threshold is reduced to 2.1E-4. And there are also some other bogus vertical linear features picked out on the side of the image at this threshold.

[f] For each of these three subimages, the threshold detects far too many linear features to be useful.

[g] This is another noisy image like Image 5a, but the trip wire detection works better in this case.

Image	Actual	# of linear features found with threshold $T =$			$\max(\tilde{U})$
		2.4E-4	3.0E-4	5.0E-4	
Image 5 (512×512)	2	2	1	1	8.4E-4
Image 5a (512×512)	2-3[d]	1[e]	1	0	3.1E-4
Image 5b (512×512)	3	3	1	1	8.5E-4
Image 5c (512×512)	3	3	2	1	8.5E-4
200×512 sub-image	2	–[f]	–	–	2.4E-3
150×512 sub-image	2	–[f]	–	–	2.8E-3
100×512 sub-image	2	–[f]	–	–	4.2E-3
Image 5d (512×512)	2-3[d]	2[g]	2	0	3.2E-4
Image 5e (512×512)	3	2	2	1	8.5E-4

Table 1.1: How many linear features are recognized in each image, using the **S-4** scaling? The sub-images of Image 5c are horizontal strips of width 512 pixels, centered vertically on the main trip wire.

Scaling S-5:

Table 1.2 summarises the results for Image 5c using the scaling **S-5** with $\alpha_u = \max(M, N)$. The key thing to notice is that the $\max(\tilde{U})$ column is almost constant for the various dimensions of sub-image.

Image	Threshold $T =$			$\max(\tilde{U})$
	0.1	0.12	0.2	
Image 5c				
full 512×512 image	–	3	1	0.433
200×512 sub-image	2	1	1	0.492
150×512 sub-image	1	1	1	0.430
100×512 sub-image	1	1	1	0.427

Table 1.2: How many linear features are recognized in each image, using the **S-5** scaling? The sub-images of Image 5c are horizontal strips of width 512 pixels, centered vertically on the main trip wire, which should optimally show two linear features.

Here a threshold value T lying in the range 0.12 to 0.2 seems to reasonable (although it may be taken as large as 0.4 and still catch the main horizontal wire). While the second oblique feature in the three subimages does not get detected in this range of T, it is important to ask ourselves whether this is more important to detect *all* suspect linear features, or just to signal an alarm that there is *at least one* suspect wire.

One final observation: as the strip is reduced in height, the oblique feature no longer runs the length of the image and so is much harder to detect. This spells trouble for detection of trip wires where we have long, narrow images in which the wire does not run horizontally.

Idealised images.

To verify some of our previous conclusions we also generated a set of artificial images of wires of various types to check how the detection algorithm works on non-square images in an idealised situation. Four 300×300 images are pictured Figure 1.5.

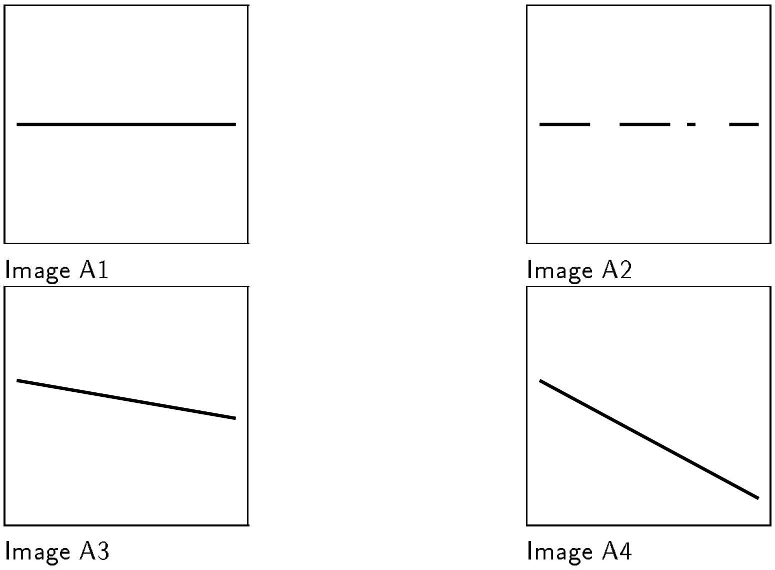


Figure 1.5: The four "idealised wire" images.

The wire detection algorithm is run on all four images with a threshold of $T = 0.2$ and scaling **S-5**, and the results are summarised in Table 1.3. Again, the scaling **S-5** with a threshold value of

Image	$\max(\tilde{U})$ for sub-image of size ...			
	300×300	150×300	75×300	38×300
A1	0.68	0.68	0.68	0.67
A2	0.47	0.47	0.47	0.45
A3	0.54	0.53	0.53	0.42
A4	0.62	0.42	0.26	0.16 (*)

Table 1.3: Maximum scaled Radon transform for the four artificial wire images, with scaling **S-5**. The "(*)" indicates that threshold with $T = 0.2$ would fail.

0.2 works well, except in the case of a wire at a large angle to the horizontal (Image A4), where it may fail if the image window is too narrow.

Difficult images.

Finally, we report on a second set of particularly "difficult" images obtained from ITRES near the conclusion of the Calgary Workshop. They comprised a sequence of 6 images, each taken in 5 filter bands (and named `file`m`_b`n`.tif`, with m corresponding to the image number, and n the band).

The results for this set of images were somewhat disappointing. The images taken in bands $n = 4$ and 5 were the clearest, and yet we were still unable to recognize the wire in any of these when the entire 900×461 image was processed. However, if we took the original image and grabbed a sub-image as small as 200×461 (pictured in Figure 1.6), then the wire can be detected, provided the edge detection method was changed. The default `sobel` edge detection option used in MATLAB does not work here, and we needed to use the `log` option instead (see Figure 1.7). After running

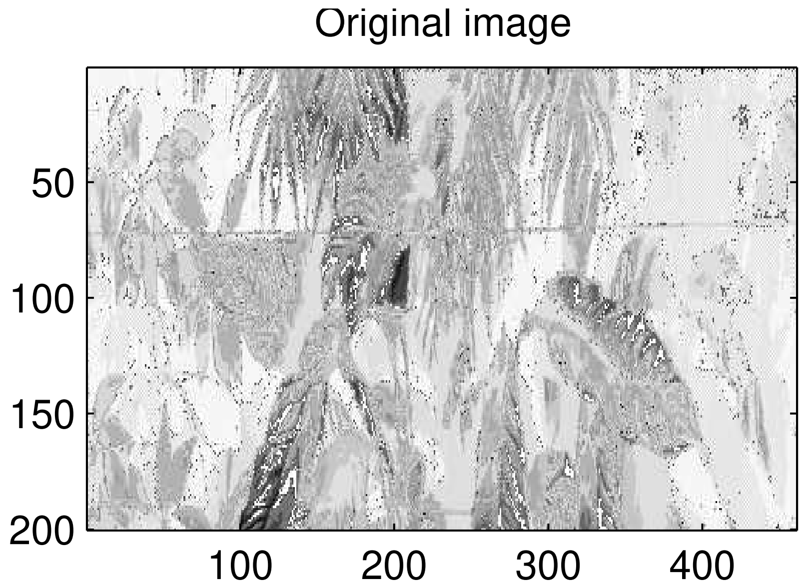


Figure 1.6: The top 200×461 portion of the image `file5_b4.tif` (original size: 900×461).

the inverse Radon transform on the `log` edge detected image, we obtain the plot in Figure 1.8. In constrast with the original set of images, the inversion here is very sensitive to the choice of threshold: $T = 0.25$ finds a large number of spurious linear features, while $T = 0.30$ fails to locate any wires. While this level of threshold is too high to detect *all* of the linear features from the set of "difficult" images, it is still sufficient to detect the major ones in Image 5c, for example.

We now summarise by considering the possible reasons for the failings of our algorithm with this last set of images:

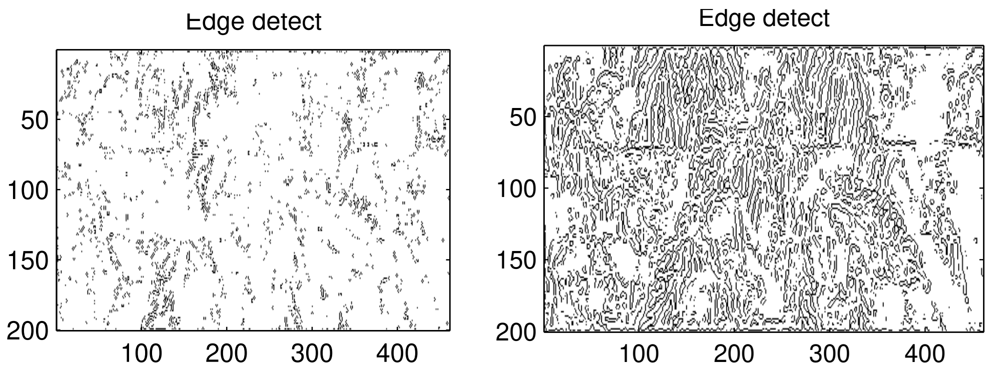


Figure 1.7: Edge detected image using options `sobel` (left) and `log` (right).

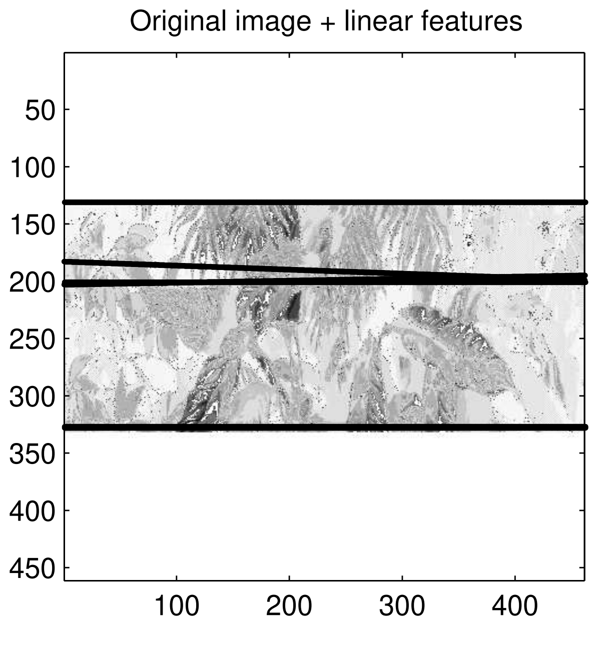


Figure 1.8: Inverse transform of the `log` edge detected image with $T = 0.27$ and scaling **S-5**.

- First of all, we may need to restrict the size of the image to a certain minimum, say $\max(M, N) < 500$. When using smaller subimages, the algorithm seems better able to detect individual linear features. It may be due to the fact that chance alignment of points in the edge-detected image is much more likely when the image is larger. Indeed, if we are to process CCD camera-generated images in real time, then we will likely be running on thin-strip type images anyways!

- second, these "difficult" images contain more noise than the others, since the wire appears to have a much more grainy texture. This seems to be causing a problem in the edge detection step (even with the `log` results from Figure 1.7, where the wire is not as well resolved by the edge detection). Perhaps there is another edge detection algorithm that will deal better with grainy lines by "smoothing" the edge detected image.

  Preliminary tests suggest that the choice of `log` edge detection does not significantly affect the results from the other sets of images considered provided that the threshold is increased to $T = 0.30$. Also, the results seem more sensitive to the choice of threshold than with `sobel` edge detection. Clearly, some more work is required here, to find an approrpriate edge detector and determine an image-independent threshold.

1.3.6 Speeding up the algorithm.

To get an idea of how fast the algorithm needs to be in order to process the images in real time, we can use the following information provided by ITRES:

- the CCD camera acquires one image line in 20 ms, and so the image is acquired at a rate of 50 rows per second;

- the truck on which the camera is mounted moves relative to the ground at 0.25 km/h or 7 cm/s;

- the boom length is on the order of 1 m;

- approx. 10 seconds to recognize a trip wire from image acquisition to danger zone

The algorithm as described was implemented directly in MATLAB using the default Laplacian filter, Sobel edge detector and Radon transform. For an image with N^2 pixels the MATLAB Radon transforms used order $\mathcal{O}(N^4)$ operations which is very inefficient and will make it very hard to use the method in real time. In contrast, methods based upon the FFT (as advertied in the MATLAB manual, but not implemented in the code) should take $\mathcal{O}(N^2 \log(N))$ operations. In an article by Brady [1] a direct algorithm (i.e. not one using the FFT) is proposed for the Radon transform which also takes $\mathcal{O}(N^2 \log(N))$ operations. Both of these methods are worth implementing.

It is possible that by exploiting special features of the image we can produce an algorithm which is $\mathcal{O}(N^2)$. To do this we observe that in detecting a trip wire we do not look at one image, but rather a *sequence* of images which are produced as the truck moves forward. Given an original reference image, the next image will be formed from this by adding one new row of N pixels and deleting a row of N pixels. Furthermore a straight line in the original image can be clearly extended to a straight line in the new image. Thus to find the projection of the new image onto such a line we can take the projection of the reference image onto this same line, delete the points from the pixels to be discarded and add on the points from the new line of pixels. For any given line we would only have to add on a bounded number of pixels and there are $\mathcal{O}(N^2)$ such lines. Thus the Radon transform of the new image can be obtained from that of the reference image with $\mathcal{O}(N^2)$ operations, which is *much* faster than taking the Radon transform of the new image without reference to the earlier image. This method has some messy book-keeping associated with it (for example the offset ρ will change when considering the same line in the reference and new images, although its angle θ will not) but should be not too difficult to implement. We did not attempt to do this during the course of the Workshop.

1.4 Other Methods.

We now return briefly to the two other methods for trip-wire detection mentioned earlier that relied on taking a series of one-dimensional slices of the image.

Suppose that we have an $M \times N$ image of M horizontal and N vertical pixels of an image $u(x, y)$. Each of the horizontal lines of M pixels at $N = n$ gives a one-dimensional sequence $v_n(x)$ which can be processed. A trip wire at an oblique angle should in principle give a localised maximum in the function $v_n(x)$ at a point $x(n)$ that depends upon n. A feature of the linearity of the wire is that $x(n)$ should be a *linear* function of n.

This hypothesis was tested by taking a series of horizontal slices of the test images described in the earlier sections. When applied to the original image the results were very inconclusive. Indeed, the wire did not show up very well as a localised maximum and was very easy to miss. To improve the resolution, the images were pre-processed as before by applying a Laplacian filter to emphasise the curvature. When horizontal slices were taken from the filtered images, the wire showed up much more clearly, and for certain examples under certain conditions the motion of the maximum from one horizontal slice to the next showed up quite clearly, but were still in general rather hard to detect. Attempts to automate this process (for example, by stringing together a series of one-dimensional slices and using a spectral analysis of the resulting string) were not very effective.

These procedures seem less certain and much harder to automate than the Radon transform methods.

1.5 Summary and Future Directions.

In summary, the pre-processed, Radon transform-based detection method seems the best one to use. There are some issues that arose while testing the method:

- On the set of "difficult" images, the wire detection was most problematic. The `log` method of edge detection seemed better than the default `sobel`, but it is still worth considering other edge detection algorithms that might deal more effectively with noisy or grainy images.

- The algorithm works best when the wire is horizontal and spans the length of the image. If we are constrained to thin-strip type images, then it may not be possible to detect oblique wires reliably. A simple solution is to run the algorithm twice, once with the image and a second time with the same image rotated through 45°.

More investigation is clearly needed on different approaches to edge detection and thresholding. If the image-dependency in these aspects of the algorithm can be eliminated, then a fast, FFT-based implementation will make real-time, automated detection of trip wires a possibility.

Acknowledgements

We would like to express our appreciationn to Avygdor Moise of ITRES Research Ltd. for posing this problem, and for his help throughout the Workshop.

Appendix A: Matlab Code

The Matlab code for reading an image and applying steps 1–3 of the trip wire detection algorithm is included below. The thresholding and inverse Radon transform are performed within the function `plotinvr.m`, which can be downloaded from the Web site

`http://www.jms.sfu.ca/radon.html`.[c]

All of the other code and test images used in generating the results in this report are also available at this site, along with a complete description.

1.5.1 Main program.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Main Program:
%%%   Perform the initial stages of the trip wire detection
%%%   algorithm using a combination of Laplacian filter,
%%%   edge detection, and Radon transform.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%   Read the image and normalise the intensities
%%%   to the range [0,1]:
```

[c] This site will only be activated once authorised by a representative from ITRES Research Ltd..

```
I    = double( imread( 'images/VEG5C.TIF' ));
Iraw = I;          % save original image
I    = (I - min(I(:))) / (max(I(:)) - min(I(:)));

%%%   Transpose the image so the number of columns is largest:
[nx, ny] = size(I);
if ny < nx
  I = I', [nx, ny] = size(I);
end;

%%%   Filter image using Laplacian:
h = [-1 -1 -1; -1 8 -1; -1 -1 -1];
Ilap = filter2( h, I );

%%%   Edge detection (options are log, prewitt,
%%%   roberts, zerocross and sobel (the default)):
edgealg = 'sobel';
Iedge = edge(Ilap, edgealg );

%%%   Compute the Radon transform:
th = 0:179;
[R, xp] = radon(Iedge, th);

%%%   Threshold and invert the Radon transform:
plotinvr( I, R, th, xp, 0.2, 5, 50 );
%%%   END.
```

Bibliography

[1] M. L. Brady, "A fast discrete approximation algorithm for the Radon transform," *SIAM Journal on Computing*, 27(1):107-119 (1998).

[2] A. C. Copeland, G. Ravichandran and M. M. Trivedi, "Localized Radon transform-based detection of ship wakes in SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, 33(1):35-45 (1995).

[3] R. O. Duda and P. E. Hart, "Use of the Hough tranformation to detect lines and curves in pictures," *Communications of the ACM*, 15(1):11-15 (1972).

[4] *The KUIM Image Processing Library*, Department of Electrical Engineering and Computer Science, Kansas University, 1998. `http://www.tisl.ukans.edu/ jgauch/kuim/kuim.html`.

[5] MATLAB *Image Processing Toolbox Manual, Version 2.0*, The MathWorks Inc., 1997.

[6] L. M. Murphy, "Linear feature detection and enhancement in noisy images via the Radon transform," *Pattern Recognition Letters*, 4:279-284 (1986).

[7] P. Toft, *The Radon Transform*, `http://eivind.imm.dtu.dk/staff/ptoft/Radon/Radon.html`.