A Model-Based Approach to Finding Tracks in SAR CCD Images

Tu-Thach Quach Rebecca Malinas Mark W. Koch
Sandia National Laboratories*
Albuquerque, NM 87185-1163
tong@sandia.gov, rmalina@sandia.gov, mwkoch@sandia.gov

Abstract

Combining multiple synthetic aperture radar (SAR) images taken at different times of the same scene produces coherent change detection (CCD) images that can detect small surface changes such as tire tracks. The resulting CCD images can be used in an automated approach to identify and label tracks. Existing techniques have limited success due to the noisy nature of these CCD images. In particular, existing techniques require some user cues and can only trace a single track. This paper presents an approach to automatically identify and label multiple tracks in CCD images. We use an explicit objective function that utilizes the Bayesian information criterion to find the simplest set of curves that explains the observed data. Experimental results show that it is capable of identifying tracks under various scenes and can correctly declare when no tracks are present.

1. Introduction

Multiple synthetic aperture radar (SAR) images taken at different times of the same scene can be combined to produce coherent change detection (CCD) images that can detect small surface changes such as tire tracks [4, 8]. The CCD images, however, are noisy due to SAR speckle, vegetation, shadows, and other weather related phenomena. These undesirable noise sources make it difficult to identify and label vehicle tracks.

Track finding can be viewed as fitting principal curves, an extension of principal components, through a set of points [5, 7, 12]. Unfortunately, the majority of the existing principal curve algorithms assume the data contains little to no noise and that there is only a single principal curve in the data. The noisy nature of SAR CCD images, along with the fact that there might be multiple or no tracks in a single image, limit the usability of existing principal curve algorithms.

The difficult nature of this problem is exemplified by the limited success of existing techniques [2, 3]. Given a search cue, e.g., starting location of a path, the technique in [2] finds the vehicle track by tracing parallel lines (tire tracks). The proposed method can find a single track provided that the user supplied the initial search cue. A related method uses cubic spline fitting to extract vehicle tracks [3]. It is limited to a single non-overlapping track.

In practice, a scene can have an arbitrary number of tracks, including no tracks. In order to automate the process of finding tracks, we must be able to account for these conditions. This work presents a novel approach for automatic track finding that can identify and label multiple tracks as well as declaring when no tracks are observed in the image. Unlike existing techniques, our approach defines an explicit objective function that quantifies how well our model fits the observed data. It further penalizes complex models and favors simple models that explain the data well. This makes it possible to determine when we have found all tracks in the image or whether there are no tracks at all.

Our approach is presented in Section 2. Experimental results demonstrating the effectiveness of our algorithm under various scenes are shown in Section 3. Concluding thoughts are provided in Section 4.

2. Finding Tracks

Given a CCD image, our goal is to identify and label the vehicle tracks. Our algorithm must also be able to correctly declare when no tracks are present. We assume that the input to our algorithm consists of a set of 2-dimensional points that indicate the locations on the image that are likely to be parts of a track. This can be obtained via different techniques such as

*Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

normalized cross-correlation with a template [9] and is not the focus of this paper.

Given a set of n spatial points, $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, we identify the tracks by modeling them as a set of m curves, $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$. The distance of a point to a curve is the distance from that point to its projection on the curve. We assume this distance is normally distributed with zero mean and a variance of σ. Since the input is likely to be noisy, we use a special curve, c_0, to capture the noise. Our model is a Gaussian mixture model with the following likelihood function:

$$L(\mathcal{X}|\mathcal{C}) = \prod_{i=1}^{n} \sum_{j=0}^{m} \pi_j L(x_i|c_j), \tag{1}$$

where π_j is the mixing coefficient of curve c_j. For real curves, e.g., $j \geq 1$,

$$L(x_i|c_j) = \frac{1}{||c_j||} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-||x_i - c_j||^2}{2\sigma^2} \right) \right), \tag{2}$$

where $||c_j||$ is the length of curve c_j and $||x_i - c_j||$ is the Euclidean distance from point x_i to its projection on curve c_j. For the noise curve, e.g., $j = 0$,

$$L(x_i|c_0) = l_{\text{noise}}, \tag{3}$$

where l_{noise} is a constant.

As with any model selection problem, we must be careful so that our model does not overfit the data. In other words, we must quantify the complexity of our model and penalize more complex models. We capture this using the Bayesian information criterion (BIC) [10]:

$$B(\mathcal{X}|\mathcal{C}) = -2\log(L(\mathcal{X}|\mathcal{C})) + k\log(n), \tag{4}$$

where k is the number of free parameters of the model. In our case, $k = \sum_{j=1}^{m} |c_j| + m$, where $|c_j|$ is the degree of freedom of curve c_j, and the term m comes from the mixing coefficients. As an example, if c_j is a line segment in a two-dimensional space represented by its endpoints, then $|c_j| = 4$.

Our task is simply to find a set of curves that minimizes (4). Our approach consists of the following steps:

1. Set $\mathcal{C} = \emptyset$, $B_{\text{min}} = B(\mathcal{X}|\mathcal{C})$
2. Find a line segment c through $\mathcal{X}$
3. If $B(\mathcal{X}|\mathcal{C} \cup c) < B_{\text{min}}$
   (a) Set $\mathcal{C} = \mathcal{C} \cup c$, $B_{\text{min}} = B(\mathcal{X}|\mathcal{C})$
   (b) Remove points assigned to $\mathcal{C}$ from $\mathcal{X}$
   (c) Repeat step 2.

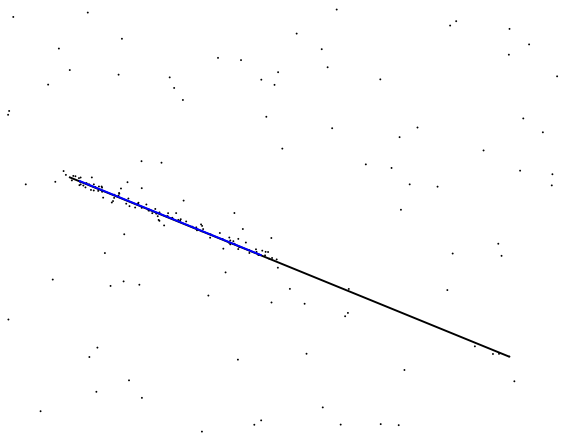


Figure 1: Forming line segments using the extreme ends of the projected points may result in poor segments due to noise (black line). Our approach finds segments that fit the density of the data (blue line). Best viewed in color.

4. Refine $\mathcal{C}$ and the model parameters iteratively until convergence
5. Merge segments in $\mathcal{C}$ to form curves

Steps 2 and 3 are responsible for finding initial line segments that fit the data well. Note that step 1 computes B_{min} on an empty set of curves. This enables the algorithm to declare that no tracks are found if the condition in step 3 is not satisfied. Step 4 is the standard expectation-maximization procedure that iteratively refines the model parameters. Prior to step 5, our curves are simply line segments. A vehicle track, however, may consist of several line segments. Step 5 merges line segments into curves. The following subsections describe steps 2 and 5 in details.

2.1. Finding Line Segments

Step 2 involves finding line segments that fit the data. We accomplish this by a two-step process. First, we find a line that fits the data, then we form a line segment from the obtained line. Any line-finding procedure must also account for the fact that the data can be noisy. We use RANSAC [6], a line finding algorithm that is robust to noise.

Once a line is found, we project the associated points onto it. We cannot simply form a line segment by using the extreme ends of the projected points. Doing so may result in long segments due to noise as illustrated in Figure 1. For better results, we must try to identify valid parts of the line to form our segment. For each projected point y, we perform the following steps:

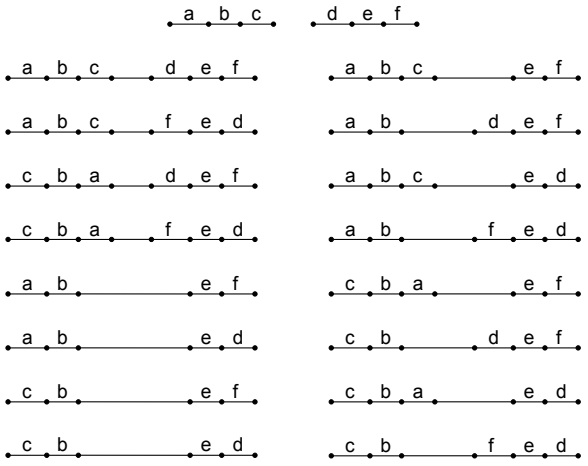


Figure 2: All 16 different ways to join two curves. The first curve is represented by line segments a, b, and c. The second curve is represented by line segments d, e, and f.

1. Compute μ, the mean value of all projected points within distance r from y
2. Set $y = \mu + r$
3. Repeat step 1 until μ changes by less than $\frac{r}{2}$

The segment corresponding to point y is defined by the first and last μ's. Among all obtained segments, we select the longest one. Intuitively, the algorithm walks along the line until it finds gaps in the projected points and is analogous to mean shift in one dimension. The search is one-dimensional along the line.

It is not necessary that we find the best line segment that captures all the points belonging to the actual segment. If we capture only half of the actual line segment (due to breaks), we can still discover the other half later in the main algorithm. What is more important is to prevent forming long segments that include noise points. This is accomplished by setting the threshold for convergence to a relatively large quantity, $\frac{r}{2}$. In all our experiments, the parameter r is derived from σ, $r = 10\sigma$.

2.2. Merging Curves

Step 5 involves joining curves that should be connected to each other. Initially, the curves are just line segments. As we continue the process, these segments turn into curves, each consists of piecewise continuous line segments. Given two curves, c_i and c_j, we can simply form a single curve by adding a new line segment that connects them. In addition, we can also remove an end segment from either or both curves and then join them. This enables two segments that are parts of a straight track to form into a single curve with fewer model parameters. There are a total of 16 ways to join two curves as illustrated in Figure 2.

For each pair of curves in $\mathcal{C}$, we form a new curve for each configuration as described above. This new curve replaces the previous two curves. We then compute the BIC of the new model. Among all pairs of curves, we keep the one with the smallest BIC. If this BIC is smaller than the original BIC, we update $\mathcal{C}$ by joining the corresponding curves and repeat the process.

3. Experimental Results

We first demonstrate the steps of our algorithm using simulated point data. We generate the data shown in Figure 3(a) using 3 line segments, each consists of 100 points. These points are displaced by a zero-mean Gaussian noise with $\sigma = 4$. In addition, we add 300 random noise points for a total of 600 points. We set $l_{\text{noise}} = \frac{1}{a}$, where a is the smallest area that encloses all points. Our algorithm initially finds 4 segments as shown in Figure 3(b). This is due to a break in one of the segments. After merging, our algorithm correctly produces 3 curves as shown in Figure 3(c). An example with more curvature is shown in Figure 3(d). The data consists of 200 points, 100 points for the curve plus 100 noise points. Due to the high curvature, no single line segment could capture the entire curve. The algorithm still finds the curve and represents it with several short segments.

We now demonstrate the effectiveness of our approach using real SAR CCD images from a publicly available data set [1]. As described earlier, given a CCD image, we identify pixel locations that are likely to be parts of a track and use these points as input to our algorithm. We use a simple approach (other techniques can be used as well), perhaps the simplest, based on normalized cross-correlation. Our template is a 5-by-21 image that resembles a short track (5 pixels long). We correlate the CCD image with the template rotated at 5-degree intervals and keep the maximum coefficients at each pixel. This results in a correlation image. We then threshold the correlation image and perform basic binary morphological processing to obtain our input points. We set $\sigma = 4$ and $l_{\text{noise}} = \frac{1}{h \times w}$, where h and w are the dimensions (height and width) of the input CCD image. The value for l_{noise} simply means that we assume the noise is uniformly distributed throughout the entire image. The results are shown in Figure 4.

Most tracks are represented by several segments due to their natural curvatures. The last image does not have a track and our algorithm correctly states that. There are a total of 3 tracks in Figure 4(d). Our al-

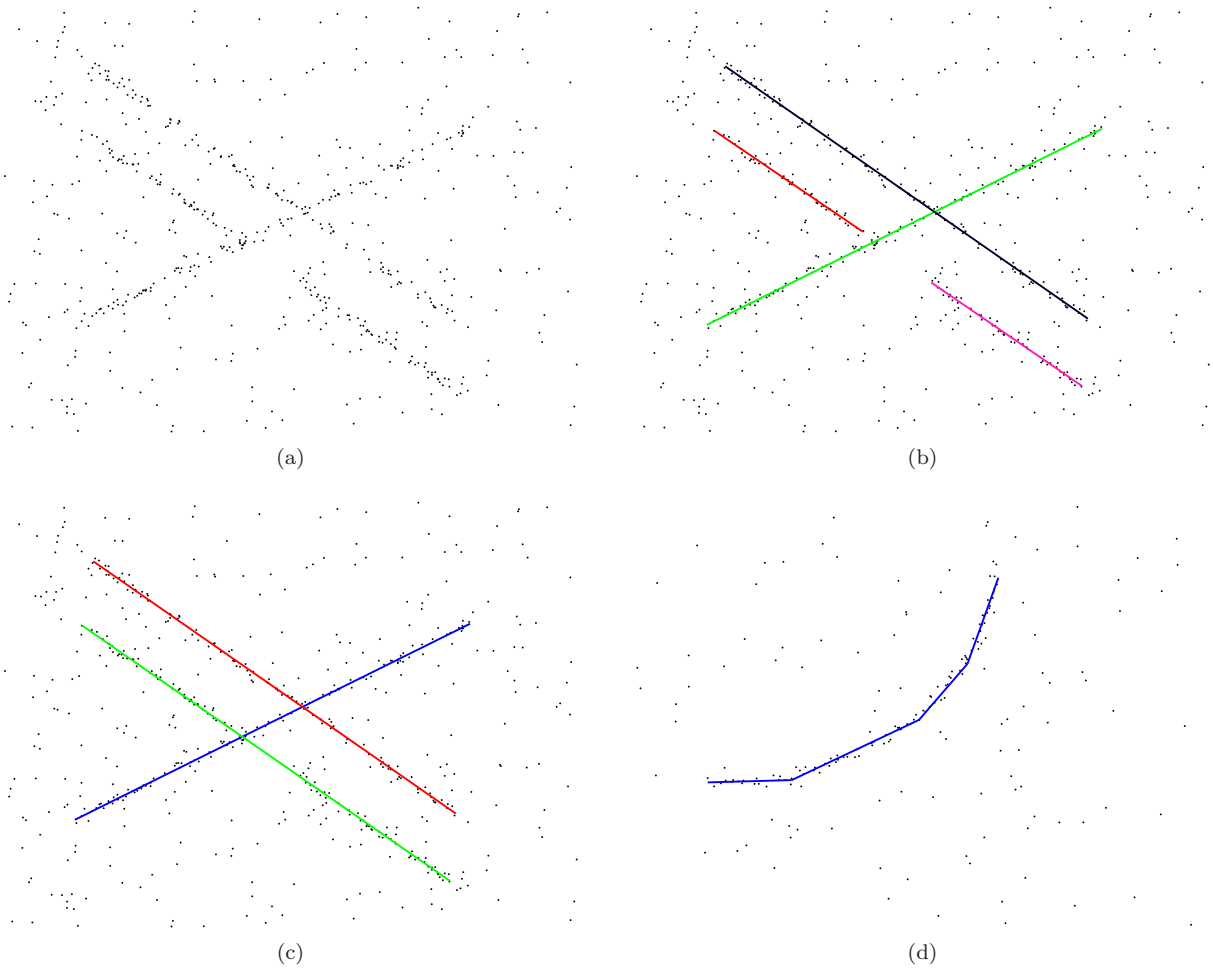


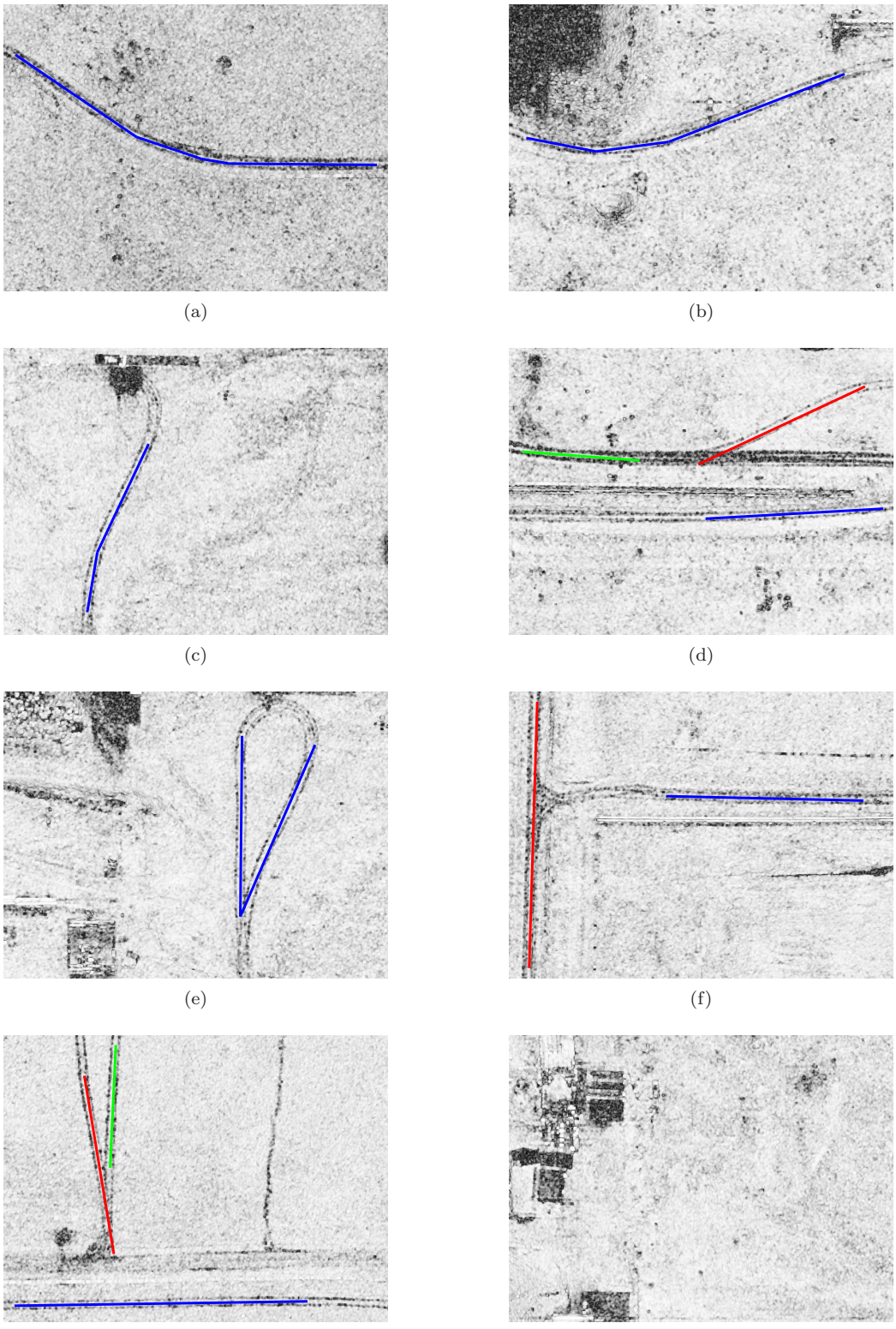
Figure 3: Algorithm illustration: (a) input data (3 segments plus noise), (b) obtained segments prior to merging, (c) final results, (d) another example with more curvature. Data points are displaced by a zero-mean Gaussian noise with $\sigma = 4$. Best viewed in color.

gorithm finds parts of all three tracks. The region between the green and red tracks does not match our template well. As a consequence, there are few points along that region.

To quantify the performance of our algorithm, we evaluate it on a set of 40 CCD images of size 800×600 containing simulated tire tracks of various curvatures. Each test image is generated from a real SAR image pair (from the same publicly available data set) and contains a single randomly-generated tire track. The tire track midline is taken to be ground truth. Given a track and a pair of SAR images of the same scene, the track is added to the output CCD image by adding random (Gaussian) phase shifts to pixels along the track trajectory in the non-reference image of the SAR image pair [11]. We generate three test sets of varying track thicknesses: *light*, *medium*, and *dark*. The same set of background images and tracks are used to generate all three test sets. This allows for a fair comparison among the three sets. The average track length is 705.18 pixels. Example images are shown in Figure 5.

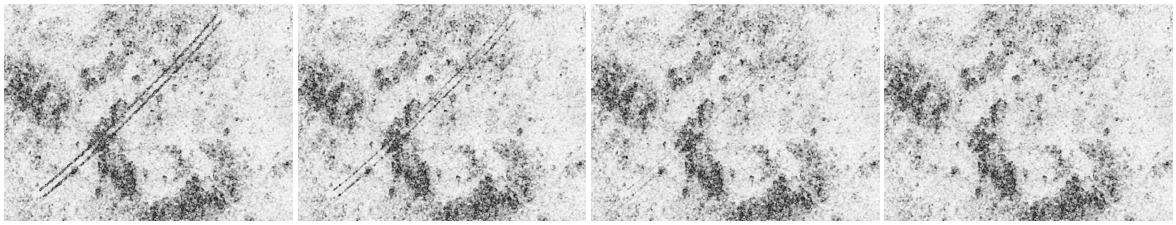
Detection is defined with respect to tire track midline (between the tracks) pixels. In the following, we define the function $D(p_1, p_2)$ as the Euclidean distance between pixels p_1 and p_2. Let m denote a midline pixel we wish to detect, d denote the candidate track pixel (*i.e.*, classified as the midline of a track by the algorithm) that is closest in Euclidean distance to m, and let t denote the tire track pixel that is closest in Euclidean distance to d. Then m is correctly detected if and only if d is contained within the area bounded by the outer edges of the tire tracks, *i.e.*,

$$D(m, d) \leq D(m, t). \tag{5}$$



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 4: Track finding results on real SAR CCD images. The last image does not have a track. Best viewed in color.



(a) (b) (c) (d)

Figure 5: Example test images: (a) *dark*, (b) *medium*, (c) *light*, and (d) original CCD image. Each set consists of 40 images of size 800×600 containing various track curvatures and background clutters. The average track length is 705.18 pixels.

To evaluate false alarms, for a given candidate track pixel d, we define m as the ground truth midline pixel closest to d, and t as the tire track pixel closest to d. Then d is a false alarm if and only if d lies outside the tire tracks, *i.e.*,

$$D(m, d) > D(m, t). \tag{6}$$

Note that a ground truth midline pixel is correctly detected as long as the closest (in Euclidean distance) candidate track pixel lies within the area bounded by the tire tracks (including the tracks themselves), and a candidate pixel is counted as a false alarm if and only if it lies outside the tire track area (candidate pixels are permitted to lie on the tire tracks).

Given the above definitions of correct detection and false alarms, we use the following accuracy metric to evaluate the performance of our algorithm:

$$\frac{TP}{GT + FA}, \tag{7}$$

where TP is the number of correctly detected midline pixels, GT is the total number of ground truth midline pixels, and FA is the number of non-track (outside tire track area) pixels incorrectly identified as track pixels. Our accuracy metric is a real number between 0 and 1. It is 1 if and only if all midline pixels are detected and there are no false alarms. As the number of false alarm pixels increases, accuracy decreases. Likewise, as the number if correctly detected pixels decreases, accuracy also decreases. We note that this metric only captures some aspects of the problem, namely detection and false alarms. It does not quantify the number of tracks detected and whether they are correctly connected. A more general metric that captures all aspects of this problem remains an open problem.

Table 1 shows the mean, median, and standard deviation of the accuracies computed for all three test sets. We use the same algorithm parameters and template to detect tracks across all three test sets. The results show that the algorithm performs well for *dark* and *medium* tracks and has difficulties detecting *light* tracks. The detection of light tracks may be improved by using a different template or noise parameter. To verify that our algorithm can determine when no tracks present, we also test it on the 40 original background CCD images that contain no tracks. Out of 40 images, the algorithm detects tracks in two images.

	Mean	Median	Std. Dev.
dark	0.9721	0.9872	0.0471
medium	0.8352	0.9108	0.1812
light	0.2631	0.2623	0.2495

Table 1: Mean, median, and standard deviation of accuracies for the three test sets.

4. Discussion

The ability to find tracks is an important tool in the areas of security and surveillance. While CCD images offer the possibility of seeing small surface changes, developing automatic algorithms to find these tracks have been limited. This work presented an approach that can accomplish this task. Our approach can find multiple tracks and declare when no tracks are visible.

The proposed approach is straightforward, requiring only a single user-supplied parameter, σ. It is straightforward to estimate this parameter from known data. The computation required by our algorithm consists mainly of computing the distance from a point to a line segment, which involves only simple matrix operations making the algorithm efficient. Due to the fact that we use line segments to represent curves, tracks with high curvatures might not be as smooth. This can be remedied by a final step in the algorithm that fits a cubic spline through each track.

We emphasize that our algorithm operates on a set of points. Furthermore, we are not limited to using normalized cross-correlation to extract points from the

image; any alternative method, such as those based on a classifier, can be used as a substitute as long as it produces a set of points as input to our algorithm. As a consequence, a high-quality point extraction algorithm may lead to better track finding results. This decoupling also makes it possible to use the same algorithm on different types of tracks, e.g., vehicles, foot. We defer investigating these problems to our future work.

Acknowledgment

This work was supported by PANTHER, a Laboratory Directed Research and Development (LDRD) Project at Sandia National Laboratories. For additional information about PANTHER, please contact Kristina Czuchlewski, Ph.D., krczuch@sandia.gov.

References

[1] W. J. Bow Jr. Sandia SAR data collect 2006. Technical Report SAND2006-2290P, Sandia National Laboratories, 2006.

[2] M. Cha and R. Phillips. Automatic track tracing in SAR CCD images using search cues. In *Signals, Systems and Computers (ASILOMAR)*, pages 1825–1829. IEEE, 2012.

[3] M. Cha, R. Phillips, and M. Yee. Finding curves in SAR CCD images. In *ICASSP*, pages 2024–2027. IEEE, 2011.

[4] D. G. Corr and A. Rodrigues. Coherent change detection of vehicle movements. In *Geoscience and Remote Sensing Symposium*, pages 2451–2453. IEEE, 1998.

[5] J. Einbeck, G. Tutz, and L. Evers. Local principal curves. *Statistics and Computing*, 15:301–313, 2005.

[6] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395, 1981.

[7] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[8] C. V. Jakowatz, D. E. Wahl, P. H. Eichel, D. C. Ghiglia, and P. A. Thompson. *Spotlight-mode synthetic aperture radar: a signal processing approach*. Kluwer Academic Publishers Norwell, MA, 1995.

[9] J. P. Lewis. Fast normalized cross-correlation. *Vision Interface*, 10(1):120–123, 1995.

[10] G. E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[11] E. Turner, R. D. Phillips, C. Chiang, and M. Cha. Inserting simulated tracks into SAR CCD imagery. In *Autumn Simulation Multi-Conference*. Society for Modeling & Simulation International, 2012.

[12] J. J. Verbeek, N. Vlassis, and B. Kröse. A k-segment algorithm for finding principal curves. *Pattern Recognition Letters*, 23:1009–1017, 2002.