Power Line Detection for Millimeter-Wave Radar Video

Qirong Ma

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2012

Reading Committee:

Ming-Ting Sun, Chair

Linda G. Shapiro

Zhengyou Zhang

Program Authorized to Offer Degree:

Department of Electrical Engineering

University of Washington

Abstract

Power Line Detection for Millimeter-Wave Radar Video

Qirong Ma

Chair of the Supervisory Committee:

Professor Ming-Ting Sun

Department of Electrical Engineering

To ensure the flight safety of helicopters, millimeter-wave radar imaging systems have been developed in both industry and academia. One application of such radar systems is to image the high-voltage power lines, and thus help the pilot to avoid power-line-strike accidents. In this dissertation, we present automatic power line detection algorithms for the radar video.

We investigate the defining characteristics of the power lines in the radar video. The power lines appear as parallel straight lines in the radar video, and Hough transform can be employed to detect them. The major challenge is that the radar videos are exceptionally noisy due to the ground return, and the noise points could fall on the same line which results in signal peaks after the Hough transform which are similar to the actual power lines.

To differentiate the power lines from the noise lines, in the first part of this dissertation we train a Support Vector Machine to perform the classification. We exploit the Bragg pattern, which is due to the diffraction of electromagnetic wave on the periodic surface of power lines. We propose a set of features to represent the Bragg pattern for the Support Vector Machine classifier. We also propose a slice-processing algorithm which supports parallel processing, and improves the detection of power lines in a cluttered background. An adaptive algorithm is also proposed to integrate the detection results from individual frames into a power line detection decision, in which temporal correlation of the power line pattern across frames is used to make the detection more robust. Simulation results confirm the effectiveness of the proposed algorithm.

In the second part of this dissertation, we further propose to utilize particle filter to more formally capture the temporal correlation of the power line objects, while the power-line-specific features are embedded into the conditional likelihood measurement process of the particle filter. Because of the fusion of multiple information sources, the detection of power line is more effective. We also propose a general framework of cascaded particle filters that takes advantage of the independence of different aspects of the object state, and decomposes the original tracking problem in a high-dimensional state space into several simpler cascaded tracking problems in low-dimensional sub-spaces. The reduced-dimensionality simplifies the problem and thus renders the solution more robust. Experimental simulations validate that the proposed approach can significantly improve the power line detection results.

TABLE OF CONTENTS

List of Figures

List of Tables

Acknowledgements

I want to express my deep gratitude to my advisor, Prof. Ming-Ting Sun. He is not only an excellent scholar and professor, but also a great teacher and mentor to his students including me. It is really fortunate of me to have him as my advisor. During the past few years he has led me forward on my academic journey, and he also taught me a ton of not-so-academic lessons that are equally important. His patience and spirit of hard-working will particularly inspire me. It is truly a pride of me for once being his student.

Thanks to Darren Goshi, Long Bui, and Yi-Chi Shih of Honeywell Corporation, for their support of my research project, their diligent work, and all the good discussions we had each week that made this dissertation possible.

I would like to thank the fellow students and lab mates, including Jian Lou, Ya-Ti Peng, Haowei Liu, Weiyao Lin, Di Guo, Yunbo Rao, Yuanyuan Huang, Qifei Wang, Zhongho Chen, Jun Xie and Zhouye Gu, for I have enjoyed working and spending time together with you. I am also grateful to several mentors of my internships, including Seho Oh, Donglok Kim, and James Lee who are with DRVision Technologies LLC, Zhan Ma and Felex Fernandez who are with Samsung Electronics.

And of course my friends, both in China and in the US, for your unfailing support. My parents and brother, for your unconditional love. My Lord Jesus Christ, for the life you have given me.

Chapter 1 Introduction

High-voltage power lines present hazardous operating conditions for the helicopters. The power lines are subtle objects for the pilot in the cockpit to see, not to mention the poor visibility in the case of at night or in rainy or foggy weather. Thus, power-line-strike accident has been a major threat for helicopter safety. As reported in [1], among the 934 registered helicopter accidents in the U. S. from 1996 through 2000, 50 of them are categorized as power-line-strike accidents. In these accidents, the helicopters were either destroyed or substantially damaged, and 15 of them even resulted in fatality. Most of these accidents happened at night; thus an automatic power-line detection and warning system for helicopters that can work anytime is highly desirable to ensure helicopter flight safety.

Radar is an object detection system that can especially work well at night. Unlike an RGB camera, radar system is largely independent of the environment lighting condition. Also, even though the power lines are usually subtle in the video captured by an RGB camera, the metal surface structures of the power lines can make them much more visible in the radar video. A few previous systems have been developed for power line detection with radar. In [2], a Passive Millimeter-Wave (PMMW) radar system is tested to image power lines from a vehicle. The power lines in the PMMW radar signal is reported to have higher contrast than in the RGB images, though still not very obvious. In [3], an active millimeter-wave radar is mounted on a rescue helicopter to detect power lines together with an infra-red camera and an RGB camera. However, no numeric performance evaluation of the system is reported, and the detection is performed in the raw radar Intermediate Frequency (IF) channel, which is not very robust

according to the example detection cases provided by the author. In [4], the Radar Cross-Section (RCS) model of power lines is developed, and the authors observe the so-called “Bragg-pattern” which is a distinguishing feature of the power lines in the radar signal due to their periodic surface structure. A polarimetric detection algorithm is further proposed in [5]- [6] by Sarabandi et al. for detecting power lines. However, this approach is primarily proposed for the Synthetic Aperture Radar (SAR), which is not especially suitable for helicopters that could fly at the same height as the towers and the power lines. We identify the problems with the previous works and the challenges for the power line detection problem as follows:

1. Power lines appear as multiple parallel straight lines in the radar video. Conventionally, people use the Hough transform [7] to detect the straight lines, and it was actually employed in [5]. However, in our situation, the millimeter-wave radar video is very noisy. Many noise points can fall on the same straight line, which results in peaks after the Hough transform in the line parameter domain. So, directly using Hough transform is not sufficient to differentiate the power lines and the lines formed by the noise points.
2. Most of the previous techniques for detecting power lines do not utilize important power line features, such as the Bragg pattern [5] in the radar signal, which is critical to differentiate actual power lines and noise lines, and the fact that power lines appear as multiple parallel lines. Also, in practical situations, since the helicopter moves relatively smoothly, there is significant temporal correlation between the positions of the power lines in successive video frames. It is interesting to investigate how to

incorporate these important features to help the differentiation of actual power lines and noise lines.

3. Since the power line detection and warning system is utilized to ensure the safety of the helicopters and pilots, its performance shall be well-tested and benchmarked. For such safety-critical applications, a detection accuracy of almost 100% is desired.
4. For practical applications, the power line detection and warning system needs to be able to operate in real-time (at least 10 fps (frames-per-second)). Thus, the algorithm cannot be too computationally intensive. Also, it is desirable that the algorithm can be implemented using parallel processing so that real-time detection can be guaranteed. However, previous approaches do not address this issue.

In [8] - [9], a power line imaging system based on a 94-GHz active millimeter-wave radar is reported. Unlike previous radar systems, it can synthesize the field-of-view scene containing the power lines from the Intermediate Frequency (IF) channel of the reception signal in real time at 10 fps. Based on the synthesized field-of-view, we would like to accomplish the automatic detection of power lines by applying image processing and machine learning approaches on the radar video to achieve significantly better results than those from the previous approaches. The power lines in the radar video have very distinguishing characteristics that can facilitate the robust detection of them. However, to achieve that, many technical issues remain to be addressed. The following are some key issues we are focusing on in this dissertation:

- Power line characteristics: straight lines with the Bragg pattern

Since the power lines are straight lines, we will review some of the major techniques for detecting straight lines from images, and discuss the use of a simple straight-line-detection

algorithm based on the Hough transform. However Hough transform is prone to noise lines, which in our case are incurred by the clutter background from the ground return of the radar signal when the pointing angle of the radar is low and the strong reflection of the ground is received. We propose to utilize the power-line-specific feature of the Bragg pattern to distinguish the true power lines from the noise lines. In order to achieve this, we propose to use a set of features to form a feature vector to represent the data on a line which can capture the essence of the Bragg pattern. We show that the design of this feature vector can lead to good overall performance of the detection algorithms. The feature vectors are utilized by a Support Vector Machine (SVM) to classify a candidate line into the category of power lines or noise lines.

- Temporal correlation and parallel lines

As mentioned, the positions of the power lines are temporally correlated, i.e., the same power line objects appear in proximity in the neighboring frames. Also, in practical situations, multiple power lines appear in parallel. We observe that the temporal correlation property of objects can be captured by using formal tracking methods such as particle filtering. Particle filtering offers a unified framework to represent and sequentially estimate the object state from the Bayesian probabilistic perspective. The object state probability distribution function (pdf) is represented by a group of weighted samples, or particles, and tracking is accomplished by a two-step process of prediction and update. The prediction step diffuses the particles from the last time instance by an object dynamic model, thus predicting the prior probability density of the object in the new time instance. The update step measures the likelihood of the diffused particles in the new time instance given the new observations. The

final tracking result is obtained from the posterior probability density combining the prior and the likelihood. The intrinsic characteristics of the parallel power lines can be naturally embedded into the update step, while the temporal correlation is naturally captured by the particle filter algorithm. The successful usage of these two types of information is the key to the accurate and robust detection of the power line objects. Unlike conventional particle filter tracking algorithms for video surveillance applications, we propose to use a cascaded structure of particle filters, with one particle filter for the tracking of the orientation of the entire group of parallel power lines, and another particle filter for the tracking of the range, or distance of each individual power line. As we will demonstrate, such a cascaded particle filter structure can better make use of the parallel property of the power lines, and can also simplify the original tracking problem into sub-problems in lower-dimensional spaces.

- Strong ground return noise

The strong ground return noise of the radar imager brings extra challenges to the robust detection of the power lines. When the radar points toward both the power lines and the ground, the return signal from the power lines are often mixed with the return signal from the ground, thus, “hiding” the power lines into the noise. To minimize the effect of the noise, we develop a preprocessing stage which uses a slice-wise thresholding which can suppress the noise while retaining the power line objects. The introduction of the slice structure also facilitates parallel processing, which is suitable for implementations for real-time performance. We also show that the successful tracking of the power lines and hence the good utilization of the temporal correlation can overcome the disturbance of the ground return noise in the particle filter framework.

The rest of this dissertation is organized as follows. In Chapter 2, we give an overview of the background information, including the Millimeter-Wave Radar system for power line imaging, the geometry of typical imaging scenarios and the interpretation of the imaging results. We also review some useful straight-line detection algorithms, classification algorithms, and tracking algorithms. In Chapter 3, we present a power line detection algorithm based on the Hough transform for straight-line detection, SVM classifier for differentiating the power lines and noise lines, and a heuristic adaptive post-processing algorithm to utilize the temporal correlation and the parallel lines property for good detection results. We propose the pre-processing step utilizing a slice-wise thresholding process which suppresses the noise while retaining the power line objects and facilitates parallel processing. We also show simulation results to verify the effectiveness of the proposed algorithm. Chapter 4 proposes to use a cascaded particle filter tracking algorithm for replacing the heuristic adaptive post-processing algorithm presented in Chapter 3. We show how the cascaded particle filter structure can naturally incorporate the parallel line property. We also show the use of the particle filter tracking can significantly improve the results. In Chapter 5, conclusions and future work directions are provided.

Chapter 2 Background

In this chapter, we give an overview of the background information related to the discussions in this dissertation. We also review different approaches for straight line detection, classification, and tracking, and comment on suitable technologies for achieving our goals.

2.1 The Millimeter-Wave Power Line Imager

2.1.1 Overview of the Active Millimeter-Wave Radar System

The active millimeter-wave (MMW) frequency is in the range from 30 to 300 GHz, corresponding to the wavelength from 1cm to 1mm [10], but the imaging systems need to operate at a frequency near an atmospheric window where the absorption due to water vapor is minimum [11]. These windows are centered near 35, 94, 140, and 220 GHz. MMW penetrates dielectric materials but is strongly reflected by metallic materials [12]. This property makes it very desirable in security and surveillance applications for detecting metallic objects such as vehicles. The active millimeter-wave imaging technology has also been successfully used in weapon detection [10] [13] [14]. When a target does not emit detectable MMW signals or when the passive MMW signal is weak, active illumination has to be used. Apparently, active illumination can have better imaging results than passive illumination, at the cost of the extra MMW signal generator and the coherent receiver.

The power line imaging system used in this work is an evolution of the legacy hardware that is described in [8]. A summary of the general technical specifications of this sensor is given in Table 2.1. A block diagram for the system is shown in Figure 2.1. The system hardware consists

of three basic modules: the millimeter-wave (MMW) front-end assembly, the back-end processing and data acquisition unit, and the embedded navigation solution. The front-end unit includes the MMW RF (Radio Frequency) component modules, a mechanically scanned interferometric antenna, a DDS (Direct Signal Synthesizer) signal generator, servo and control boards, and the IF (Intermediate Frequency) chain components. This front-end unit weighs less than 20 lbs., and is the only portion of the system that is mounted external to the aircraft. The back-end processing unit consists of a PC-based COTS (Commercial off the Shelf) architecture and a DSP board. Two networked PCs with an identical configuration are implemented for simultaneous power-line image display, data recording capability, and data processing, including the detection of the power lines. A signal hub is developed to distribute signals between the two PCs and interface with the front-end and the navigation unit. The navigation unit integrates a Honeywell HG9900 IMU (Inertial Measurement Unit) with a commercial GPS receiver. The back-end pallet is installed in the cabin of the helicopter. The front-end and back-end units are shown in Figure 2.2. Lastly, the radar display unit is a touch screen monitor that also supports the operation and control of the radar. The whole system consumes around 200 W of power.

Table 2.1 The Power-line Imaging System Parameters

Frequency	94 GHz
Sweep bandwidth	CW-800 MHz
Azimuth beam width	0.65°
Elevation beam width	4.0°

Range resolution	≥ 20 cm
Size	< 1 ft.^{2}
Weight	$<$ 20 lbs.
Field of view (FoV)	$+/-$ 15°
Frame rate	10 Hz

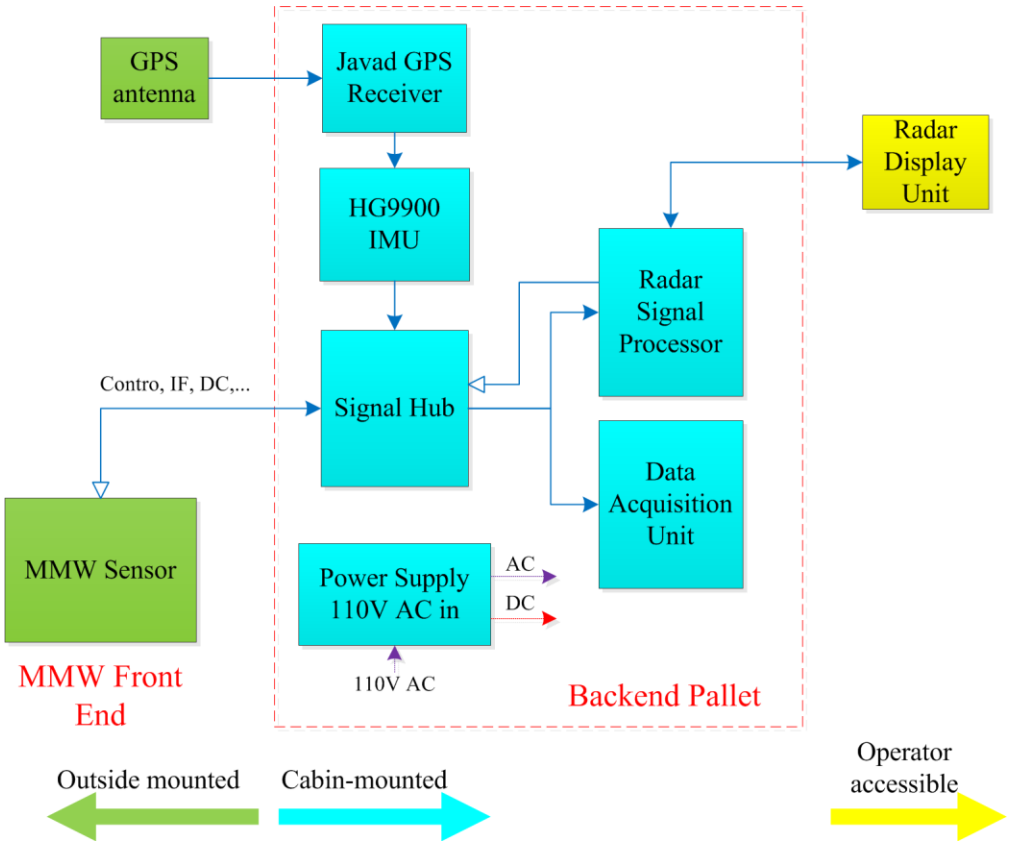


Figure 2.1 A block diagram for the active millimeter-wave radar system.

Figure 2.2 The front-end and back-end unit of the active millimeter-wave radar system.

2.1.2 Test Scenarios and Power-Line Imaging Results

A flight test was conducted in Snohomish, WA, to collect the radar video data used for developing and testing our power-line detection algorithms. Figure 2.3 offers a head-on view of the power-line scene from the cockpit at around 500 m to the power lines. The actual power lines are barely recognizable by our naked eyes even on a clear day. In this scene, there are three sets of power lines, as illustrated in Figure 2.4. The closest set to the helicopter includes three horizontal wires which are attached to a wooden pole structure of the type typically used in local power transmission. The other two sets contain two groups of three vertically stacked power lines, with a seventh line hanging above and in between, and are attached to a very tall metal tower structure typically used for long range power distribution.

Figure 2.5 shows a single frame of the power-line scene captured by the MMW radar displayed in a B-scope plot. The B-scope view is a frontal polar plot, or a range vs. angle mapping of the scene from the sensor's perspective. The vertical dimension of the data indicates the range, or the distance of the target to the radar, with its size usually being either 2048 or 4096. The top row of the data contains the furthest range from the radar sensor, and the bottom row the

closest. The horizontal dimension represents the sweeping angle of the radar sensor. As listed in Table 2.1, the sweeping angle is in the range of $+/-15°$, and the size of the horizontal dimension is usually less than 200. The aspect ratio of the radar video means that it is a long strip of data, and the image in Figure 2.5 is not in its original aspect ratio.

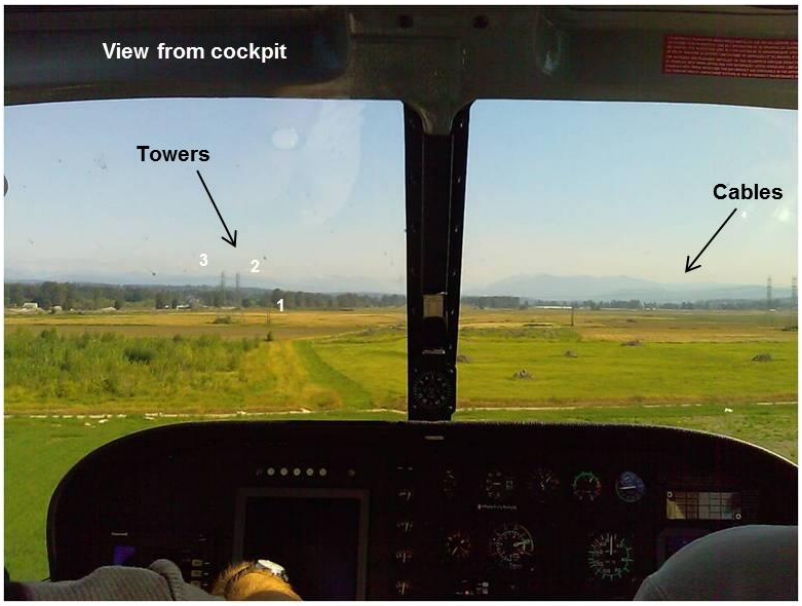


Figure 2.3 View of the power line scene from the cockpit at around 500 m to the power lines.

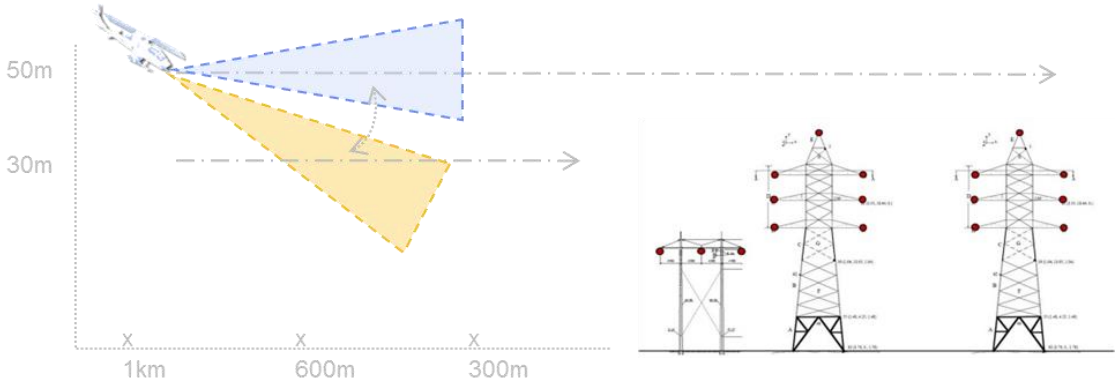


Figure 2.4 Illustration of power line target configuration and general flight paths.

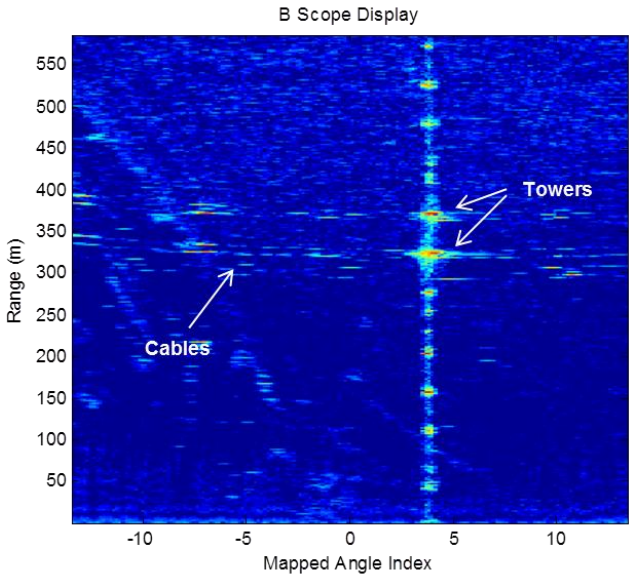


Figure 2.5 B-scope image display of the power-line scene.

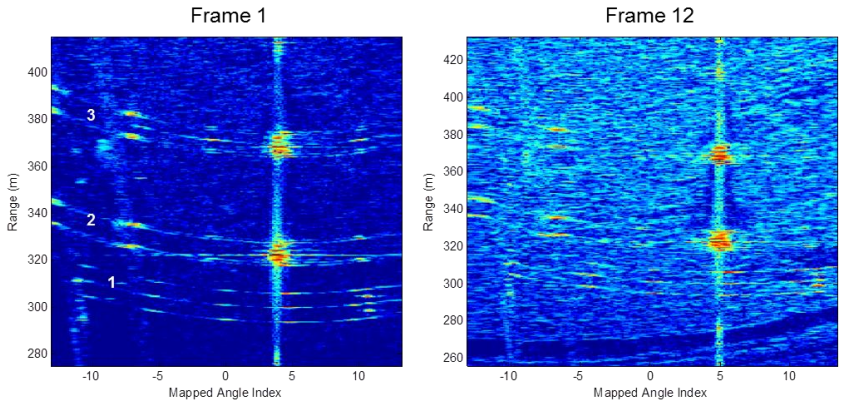


Figure 2.6 Zoom-in views showing the impact of ground return noise which depends on where the sensor is pointing.

The B-scope plot is generated directly from the raw radar IF data, and is an amplitude-only plot. In this work, we will only use the amplitude plot for power-line detection. Figure 2.6 offers a zoomed-in view of the power-line area in two frames. The three sets of power lines are evident

in them. The closest set especially has a clear view of the so-called “Bragg pattern”, i.e., the periodic pattern of the line segments. The Bragg pattern is generated due to the periodic surface structure of the power line. As shown in Figure 2.7, a typical power line is consisted of several metal wires twisting around each other, forming a periodic surface with period L, which is the distance between two braiding strands of wires. According to the Bragg’s Law of Diffraction, the peaks in the returned signal will appear in the following angles [15]:

$$\theta_n = \sin^{-1}\left(\frac{n\lambda}{2L}\right) \quad \textbf{(2.1)}$$

where $\boldsymbol{\lambda}$ is the wavelength of the millimeter-wave.

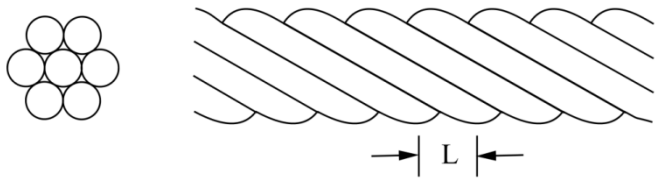


Figure 2.7 Physical structure of a typical power line.

In Figure 2.5 and Figure 2.6, the farther two sets of power lines have stronger peaks than the first set, since they are a stack of three power lines rather than one. We can also see the strong return of the towers associated with the two sets of power lines. In fact, the return from the metal tower structure is strong enough to cause saturation in the IF channel, and its effects are exhibited as harmonic ringing, or side-lobes, in the vertical range dimension of the images. This should not be confused with multiple returns at all these different ranges. Another point worthy of noticing is the much stronger background clutter noise in frame 12 of Figure 2.6, which is due

to the ground return, and is generated from a downward pitch angle directing the sensor to point at both the power lines and the ground simultaneously.

To further facilitate the interpretation of the radar images, in Figure 2.8 we show one example frame containing power lines and its corresponding scene geometry. In the scene geometry figure, the parallel lines in the gray triangle represent the wave-fronts of the millimeter-wave of the radar, and the objects on the same wave-front are shown in the radar image as objects at the same range level. For the first group of power lines, the wave-front has not met with the ground yet, so in the radar image they are clearly shown without any ground return noise. For the second group of power lines, they start to mix with the ground return noise. For the last group, they are completely hidden in the surrounding ground return noise since the same wave-fronts reach both the power lines and the ground.

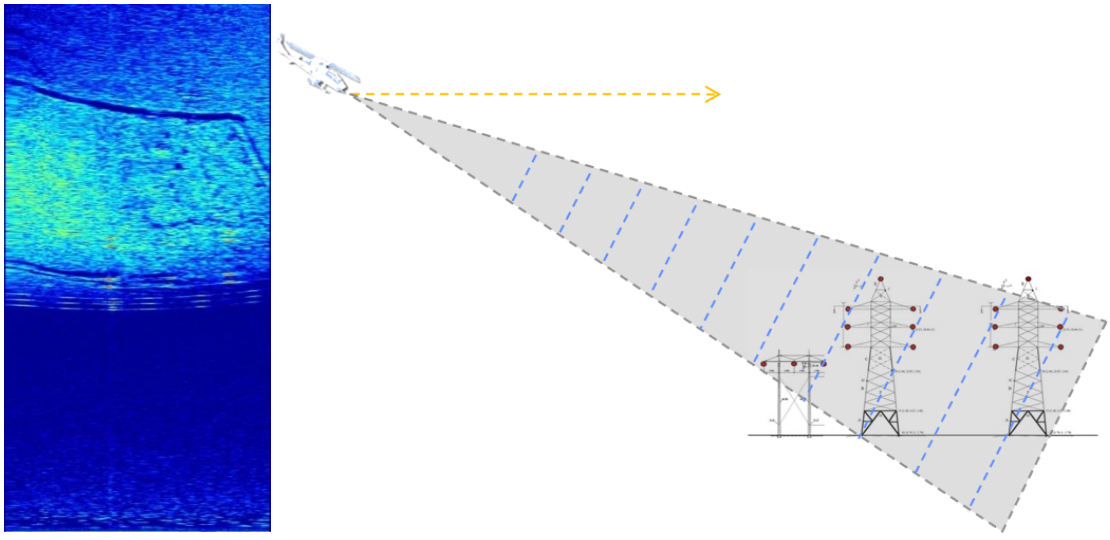


Figure 2.8 A frame and its scene geometry.

2.2 Review of Straight-Line Detection Algorithms

Straight lines are interesting features in images, and the detection of straight lines is an important operation in computer vision. With different representation of lines, the detection algorithms use different approaches, such as line detection with a particular line model, line detection with dynamic programming, and line detection with geometrical line characteristics. In this section we review some of the popular straight line detection algorithms.

2.2.1 Straight-Line Detection with the Hough Transform

After the Hough transform was invented back in the 1970s [16] [17], it has been widely adopted as a conventional method for detecting straight lines, and its generalized version is also able to detect circles, ellipses, and arbitrary-shaped curves [7]. For the simplest straight-line detection case, the Hough transform is a transformation from the image domain to the line parameter space. The original image is usually pre-processed to binary images containing the feature points (often by thresholding), so that the line pixels are the white pixels. In the image domain, each straight line can be represented by the following line equation:

$$x\cos\theta + y\sin\theta = \rho \qquad \textbf{(2.2)}$$

where θ denotes the line orientation, and ρ the distance of the line to the origin, as shown in Figure 2.9. In Figure 2.10, the basic idea of Hough transform is illustrated. For a specific point (x_0, y_0) in the binary image, all the possible straight-lines passing through it are characterized by a (θ, ρ) parameter curve with the equation $x_0\cos\theta + y_0\sin\theta = \rho$, which is a sinusoidal curve in the (θ, ρ) parameter space. On the other hand, a specific line in the image is characterized by a point in the parameter space. By transforming all the feature points after the preprocessing into

their corresponding sinusoidal curves in the parameter space, the line with the most points on it will be the "most intersected" parameter space point by the sinusoidal curves. In the example of Figure 2.10, the three points in the image are transformed into three sinusoidal curves in the parameter space, and the intersection point of the three curves represents the parameter of the line that the three points lie on. So, intuitively, the Hough transform is a voting process, in which each line feature point votes to the parameter space points that are "compatible" with it, and in the end, the parameter space points that get more votes correspond to the lines with more collinear points.

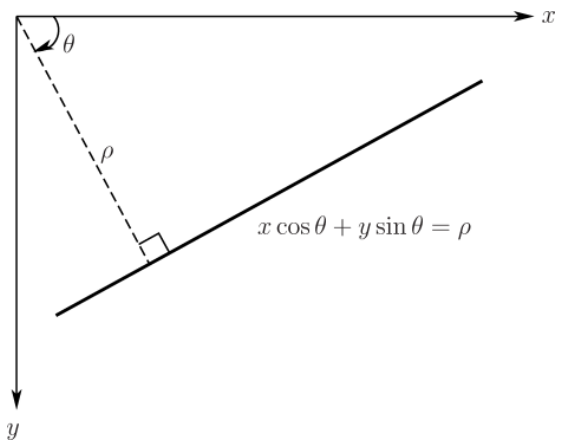


Figure 2.9 A straight line with its parameter equation.

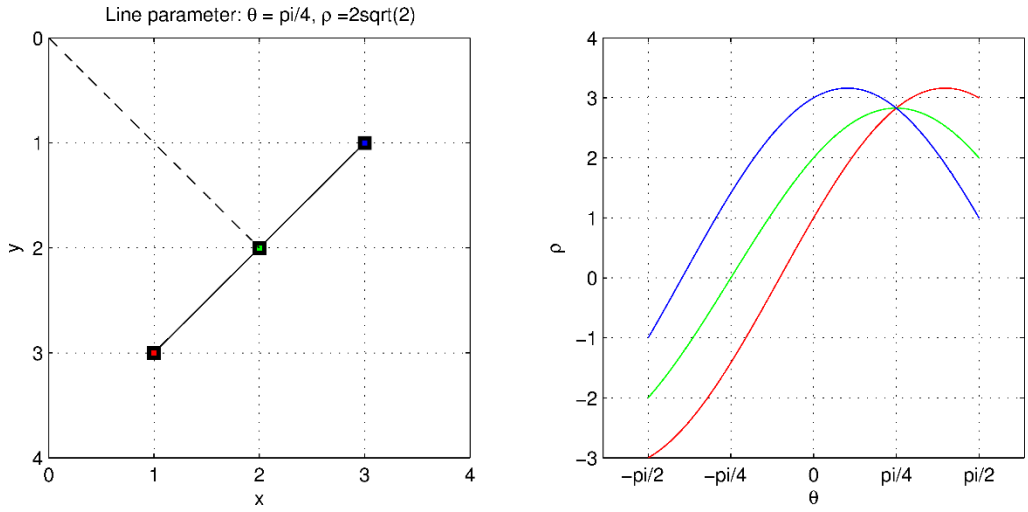


Figure 2.10 An example image with three points on a line and its Hough transform.

With the Hough transform, the straight-line detection for a binary image I is accomplished by first transforming I into the (θ, ρ) parameter space:

$$I(x, y) \rightarrow H(\theta, \rho) \tag{2.3}$$

Then, local maxima or peaks are identified in $H(\theta, \rho)$, corresponding to the straight lines in I. The selection of peaks in $H(\theta, \rho)$ is up to the application. Based on different peak selection processes and parameters, different line-detection results and performance can be controlled.

In addition to the Hough transform with circle and ellipse equations to detect circles [19] and ellipses [20], and the generalized Hough transform with any non-parametric form of curve to detect arbitrary shaped curves [21], many other variants of Hough transform have been developed. For example, [22] proposes the "randomized" Hough transform (RHT). For straight-line detection, instead of transforming an image point into a line in the parameter space as in the standard Hough transform (SHT), RHT picks up two randomly selected points in the image, and

accumulate the one point in the parameter space that is determined by the two image points. Similar idea is also developed in [23] as the probabilistic Hough transform. Compared to SHT, RHT is faster and requires less storage space for the parameter accumulators, though it is not a complete transform from the image domain to the parameter space, rather it is just a sampling of it. Other variants take advantage of certain line types in particular applications. For the work in [24] of detecting straight lines in sports video, it uses the property of sports video straight lines that they are usually long, such as marker lines in soccer and tennis fields. The so-called gridding Hough transform is proposed, in which only edge points on the grids (evenly draw horizontal and vertical straight lines on the given image) are considered for Hough transform. "linelets" (short straight line segments) in each grid block are detected first, and then connected into long straight lines. For more reviews on Hough transform and its variants, the reader can refer to [25] and [26].

Interestingly, the Hough transform is very similar, if not equivalent, to the Radon Transform widely used for tomographic reconstruction. Radon transform of a two dimensional function $f(x, y)$ is defined as its integral over straight lines:

$$Rf \equiv p(\theta, \rho) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\delta(x \cos\theta + y \sin\theta - \rho)dxdy \qquad \textbf{(2.4)}$$

In tomography, the Radon transform represents the scattering data as the output of the tomographic scan, and the inverse of the Radon transform is used for reconstructing the original density. In [27], it shows that the Discrete Radon Transform can be accomplished in $O(N^2 \log N)$ time for an $N \times N$ image, which is as fast as the Fast Fourier Transform (FFT).

2.2.2 Straight Line Detection with RANSAC

Random Sample Consensus, or RANSAC [28], is a general algorithm for parametric model fitting in the presence of noise and outliers. RANSAC first randomly selects a small number of data points and uses them to generate an instance of the model, and then checks the number of data points that are consistent with the fitted model. It iterates this process until a good enough consistent set is observed. The model is then re-estimated using all the points in the inlier set. If there are multiple structures in the data points, all the inliers of the first fitted model are removed from the data points, and the same procedures are repeated to find other structures in the remaining data points.

In [29], an algorithm for detecting quadratic curves from a binary image using RANSAC is proposed. The algorithm is shown in the procedure below.

Algorithm: Quadratic curve detection using RANSAC
Input: a binary image I, the number of quadratic curves to be detected k

while $k > 0$ **do**
  repeat K times:
    randomly select d pixels in I to determine a curve;
    label all pixels within a distance δ from the curve as inliers of the curve;
    record the current curve if it has more inliers than previous one;
  end repeat;
  accept the curve with most inlier pixels in the K hypotheses;
  remove the inlier pixels with it;
  decrease k by 1;
end while.

For straight-line detection, a hypothesis with two pixels is enough to determine the line equation, thus $d = 2$. If there are n pixels in the image, a deterministic search with $K = \binom{n}{d}$ is

guaranteed to find the straight line with the most number of inliers. However this requires $O(n^3)$ complexity because for each hypothesis it needs $O(n)$ time to test the number of inliers of the model. The work in [29] provides some probability analysis that gives a lower bound of K in order that at least one of the hypotheses contains pixels all from one true curve, thus the algorithm can successfully detect the true curve after iterating K hypotheses. It is given as $K \geq \frac{\log(1-\phi)}{\log(1-P)}$, where ϕ is the target success rate, and $P = \frac{k\binom{\mu n/k}{d}}{\binom{n}{d}}$, where μ is the "purity", or inlier percentage of the pixels. For detecting just one straight line from the image, i.e., $k = 1, d = 2$, and $\phi = 0.95$, we show the relationship between K and μ in Figure 2.11. It is clear that when μ is low, i.e., the outlier percentage is high, the number of hypotheses needed is substantially large. For example, when $\mu = 0.05$, K is in the order of 10^3 and the algorithm complexity is $10^3 O(n)$. Thus, the RANSAC algorithm has the disadvantage of being only able to handle a moderate percentage of outliers without blowing up the computational cost.

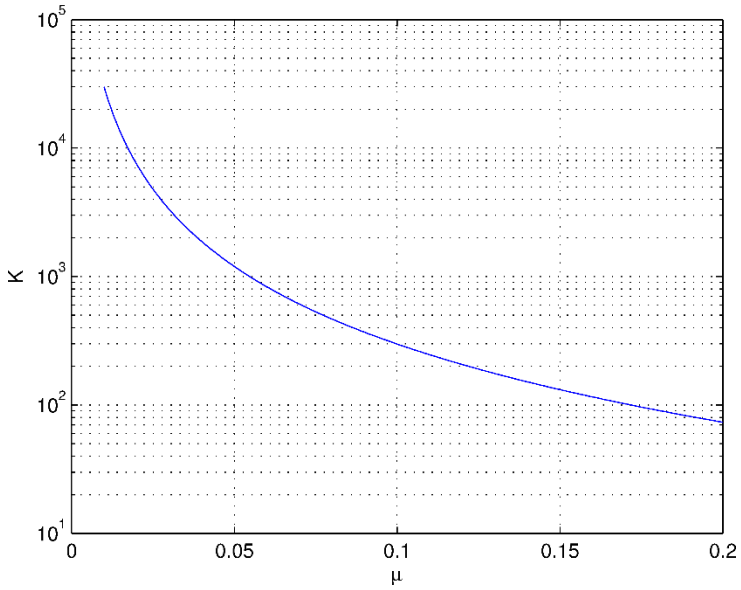


Figure 2.11 The relationship between the number of hypotheses and the percentage of inliers for RANSAC straight line detection algorithm.

2.2.3 Line Detection using Dynamic Programming

Dynamic Programming is a general methodology for optimization problems. It can solve many problems in $O(n^2)$ or $O(n^3)$ complexity, while a naïve approach would take exponential time. Dynamic Programming breaks a complex problem into simpler sub problems, and the sub problems typically overlap. Unlike recursion algorithms which have exponential times of recursive calls, Dynamic Programming algorithms save, or "memorize" results for each small sub problem, so that each sub problem has only one effective recursive call. From a bottom-up point of view, Dynamic Programming progressively computes the optimal solution for the problem from a small size to a large size (starting from the size of zero), while at each step the optimal solution depends on the optimal solution from the previously computed results. Classical examples of Dynamic Programming include the computation of Fibonacci sequence, the longest

common subsequence problem, the Knapsack problem, the matrix-chain multiplication problem, and the traveling salesman problem, etc.

Many computer vision problems including line detection can also be solved by Dynamic Programming. Fischler et al. present a technique of road detection from aerial images using Dynamic Programming in [30]. The work focuses on low-resolution aerial images, where the roads only have a width of three or fewer pixels. Firstly, road detectors are applied to generate local evidences for the presence of a road at every pixel. The road detectors include a type-I operator, which has a low false alarm rate but a high miss rate, the so-called Duda Road Operator. It is a heuristic operator that makes use of the road properties, including higher brightness than the surroundings, higher contrast with the surroundings, and lower contrast within the road. The road detectors also include several type-II operators, which have low miss rates but high false alarm rates, such as the Roberts cross gradient operator and the Sobel-type gradient operator. Then, the local evidences and constraints are combined into a cost value for each pixel, and lastly road detection is solved as to find the minimum cost path by Dynamic Programming. This algorithm is further improved in [31] by Merlot et al.

However, these methods based on Dynamic Programming may not be well suited for power line detection. For road detection, the problem is formulated as a minimum cost path finding problem by these methods, but such an approach heavily depends on the heuristic road detector that generates the cost map. Also, the straight line property is essential to the power lines, but it is not clear how to impose this property in the Dynamic Programming framework.

2.2.4 Other Line Detection Approaches

Steger proposes a line detection algorithm that uses an explicit model for lines and their surroundings in [32]. The purpose is the detection of the so-called “curvilinear structures” such as a road in an aerial image or a blood vessel in an angiogram image, rather than just straight lines. The algorithm uses a differential geometric approach, the basic idea of which is to locate the positions of ridges and ravines in the image function. Ridges are defined as points on a contour line of the image where the curvature is a maximum, while the ravines minimum. So the positions of the line edges are given by the maxima of its first derivative, or the zero crossings of the second derivative, both smoothed by a Gaussian kernel to suppress noise. For the road, it is most likely a ridge in the aerial image since its intensity is usually stronger than its surroundings, while for a blood vessel it is usually a ravine in the angiogram image. Besides the position of the line edge points, the direction of the line has also to be computed locally for each line edge point. By some geometric analysis, the author has shown that the line direction is just the direction that gives the maximum second directional derivative of the image function. This direction can be determined by calculating the eigenvalues and eigenvectors of the Hessian matrix of the image function. After the line points are detected, they are linked into continuous lines by a scheme similar to the hysteresis operation used in the Canny edge detector. The algorithm is also capable of determining the local width of a line. For a point on the line, the algorithm searches for points to the left and to the right of it where the absolute value of the gradient takes on its maximum value along the normal direction of the line, as the starting and ending point of the line, and compute their distance along the normal direction as the local line width. This algorithm has

been tested to be able to detect roads from aerial images and also to detect curve-linear structures from the MR images.

Contour extraction is a classical problem in computer vision. Conventional methods for contour extraction include energy minimization with an active contour model such as Snakes [33], and interactive segmentation such as Intelligent Scissors [34]. Lines can be viewed as an object type with a linear structure, and automatic extraction of the object contour can complete the task of line detection. This approach is often found useful in detecting roads from satellite or aerial images.

The work in [35] deals with the problem of extracting continuous contour structures from noisy or cluttered images. It specifically requires a starting point available, and grows a contour from this seed point. The contours are seen probabilistically as the paths of a stochastic process driven by both an inner stochastic dynamics and a statistic data model. In this setting, the approach takes the Monte Carlo technique based on the sequential importance sampling/resampling for tracking objects in video, where the “time” is not the real time but only an index variable associated with the progressive growing of the estimated contour in the image plane. Then the particle filter tracking algorithm is applied to progressively grow the contour from the initial seed point. The step length is fixed, and the dynamics model is a continuation of the previous step with a change in the angle, where the change of angle is drawn from either a normal distribution for regular points, or from a uniform distribution for corner points since the contour may have abrupt changes of directions on the corner points. The measurement weighting function is simply a function of the norm of the gradient. This method is applied to interactive segmentation applications, where the user input constitutes another form of the measurement

function. It is also applied to road extraction from aerial and satellite images, where the road width is also one part of the state in addition to the location.

Again, these approaches are more suitable to detect curve structures with some significant widths such as the roads. For power line detection, these algorithms do not have a clear way of utilizing the straight line characteristic. Also, due to the Bragg pattern, the power lines appear as a few segments of disconnected lines. These approaches are only able to detect a continuous line, so further steps of line linking and grouping have to be developed.

2.3 Review of Classification Algorithms

Classification is a task needed in many applications, and using computers to perform classification falls in the research area of machine learning. Usually, the machine is given some labeled samples of instances (the training data), and the classification algorithm is applied to exploit, or "learn", the relationship between the samples and their labels, so that it can predict the category of future sample instances. Our power-line detection algorithm is based on a general line-detection algorithm and a power-line specific classification algorithm, so in this section we review some existing classification algorithms. In the following description we use $(\boldsymbol{x}_i, y_i), i = 1, \dots, n$ to denote the training data set, where $\boldsymbol{x}_i$ is a feature (or attribute) vector, y_i is its associated label, and n is the total number of training samples. A feature vector $\boldsymbol{x} = (x_1, \dots, x_p)$ is a p-dimensional vector.

2.3.1 Decision Trees

Decision trees are trees that classify a sample instance by sorting it based on the attribute values. In a decision tree, each node represents a specific attribute value, and each branch of a node represents the different values or value ranges that this attribute can assume. Instances are classified starting at the root node, falling into a specific branch based on its attribute value, and finally reaching a leaf node which assigns a classification. For example, the decision tree in Figure 2.12 for diagnosing whether a patient has a cold is constructed using the training data in

Table 2.2. Decision trees are most useful when the instances are describable by attribute-value pairs and the target function is discretely valued. Some example applications include equipment or medical diagnosis, credit risk analysis, and modeling calendar scheduling preferences.

Table 2.2 Training Data for Classifying Whether There Is Cold Based on Symptoms

Symptoms				Cold
Cough	Fever	Tired	> 7 days	
No	No	No	No	No
No	No	No	Yes	No
No	No	Yes	No	No
No	No	Yes	Yes	No
No	Yes	No	No	No
No	Yes	No	Yes	No
No	Yes	Yes	No	No
No	Yes	Yes	Yes	No
Yes	No	No	No	No
Yes	No	No	Yes	No
Yes	No	Yes	No	Yes
Yes	No	Yes	Yes	No
Yes	Yes	No	No	No

Yes	Yes	No	Yes	No
Yes	Yes	Yes	No	Yes
Yes	Yes	Yes	Yes	Yes

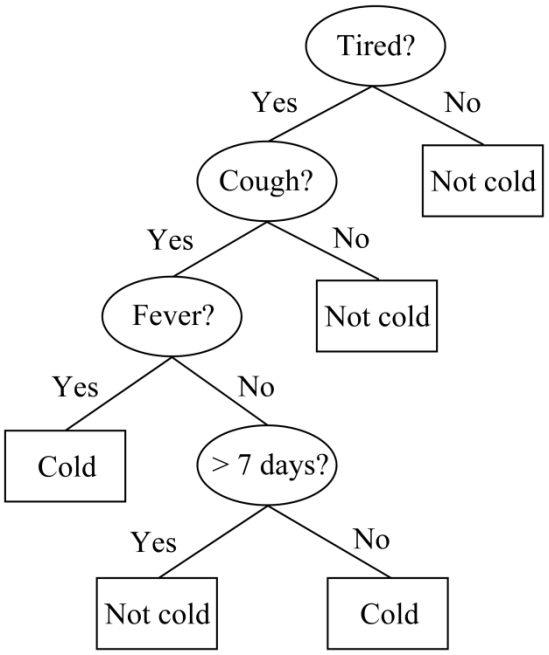


Figure 2.12 Decision tree for diagnosing cold based on the symptoms.

The problem of constructing optimal decision trees from the training samples is an NP-complete problem, thus researchers in this area have searched for efficient heuristics for constructing near-optimal decision trees. The root node of the decision tree is usually the attribute that can best divide the training samples. Methods for finding the attribute that best divides the training samples include information gain, gini index, myopic measures, etc. [36], while some study concludes that there is no single best method and the selection of the constructing algorithm depends on the particular application and dataset nature [37]. After the root node attribute is selected, the training samples are sorted according to this attribute value,

and the split subsets of the training samples are assigned to each branch of the root. The same procedure is iterated for each branch, creating sub-trees until reaching the leaf nodes when all the training samples in the branches have the same category class.

In decision tree learning, one problem often occurs is over fitting. A learned decision tree is said to be over-fitting the training sample when another decision tree exists that has a larger error on the training data but a smaller error on the testing data. To avoid over fitting the training data in decision tree learning algorithms, one approach is to stop the tree growing before it reaches a point at which the tree can perfectly fit the training data, i.e., to allow some training error. Other approaches prune the learned decision tree during the training process. A good survey for the methods of tree pruning and simplification is presented in [38]. For further review of the decision tree classification algorithms, the reader is referred to [37].

2.3.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are built upon the notion of perceptron [39]. The perceptron is basically a linear binary classifier. It computes the linear weighted sum of the feature vector, i.e., $\sum_i w_i x_i$, where $\boldsymbol{x} = (x_1, \dots, x_p)$ is the feature vector and $\boldsymbol{w} = (w_1, \dots, w_p)$ is the weight vector, and compares it with a threshold to output a binary decision. A simple learning algorithm for the weights and the threshold is to run the perceptron algorithm repeatedly over the training set until it finds a weight vector and threshold that gives correct prediction on the entire training set. However the brute-force algorithm has exponential growing complexity. More sophisticated learning algorithms iteratively update the weights according to the classification results of previous steps until convergence.

The perceptron learning algorithm cannot reach a termination if the training set is not linearly separable. In other words, perceptron can only classify linearly separable training sets, i.e., if there exists a straight line, a plane, or a hyper-plane that can completely separate the training sample instances into their correct categories. To solve this problem, Multi-layered Perceptron (MLP) has been created. As its name suggests, MLP contains a large number of perceptrons connected together in multiple layers in a directed graph, and each layer is fully connected to the next one, as shown in Figure 2.13. In MLP the activation function defining the output of a perceptron given the input is non-linear; otherwise it can be proven that any number of layers of perceptrons with linear activation function can be reduced to the single layer case. The activation function is usually a sigmoid function. During classification, the signal of the input feature vector propagates all the way through the network. Each perceptron computes an activation function value from the weighted sum of its input signals, and the output is passed along to the next layer as its input information. Because of the non-linear activation function at each perceptron, the MLP can fit non-linear target functions.

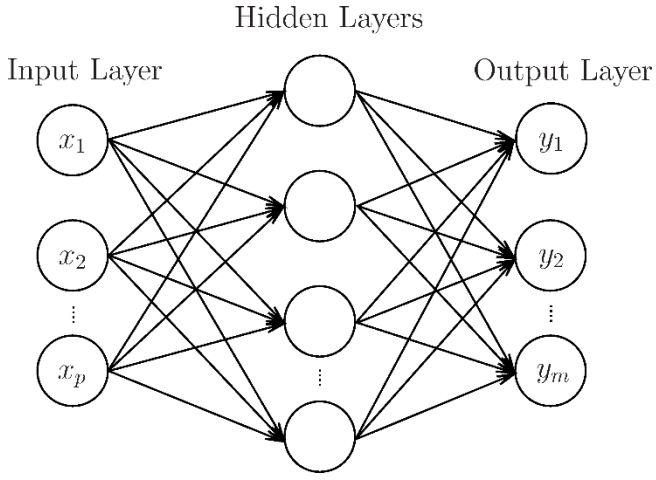


Figure 2.13 An example of MLP.

Given the fixed network architecture and the activation function of a perceptron, the behavior of the MLP depends on the weights of each perceptron. The most widely used learning algorithm to fit the weights is the Back Propagation (BP) algorithm. The weights are usually initialized with random numbers. With the training samples presented to the MLP, the algorithm first compares the output of the network with the desired output (i.e., the labels of the training sample instances), and the error is calculated in each output perceptron. Then this error is propagated back to the input of the output perceptron, adjusting the weights in the direction of lowering the output error. Then the error is further propagated back to the previous level, adjusting the weights accordingly. When the error has been propagated through the entire network and every weight has been updated, the process is repeated until convergence or some stop rule has been met. The BP algorithm is essentially a gradient descent algorithm, so it is prone to be trapped at local minimum. For more detailed review of ANN and other training algorithms, the reader is directed to [40].

2.3.3 Statistical Classification Algorithms

Linear Discriminant Analysis (LDA) is a dimensionality reduction as well as a classification technique – it seeks to reduce the dimensionality while preserving as much of the class discriminatory information as possible. For a set of p-dimensional training sample instances $\{x_1, x_2, \dots, x_n\}$ with two classes, LDA obtains the vector direction onto which the projections of $\{x_1, x_2, \dots, x_n\}$ have the maximum separability [41]. Fisher's suggestion for separability is based on the difference between the means of the two projected classes, normalized by a measure of

the within-class scatter. LDA looks for a projection where the training sample instances from the same class are projected very close to each other, and the projected means of the two classes are as far apart as possible. The maximum separability problem is solved as a generalized eigenvalue problem.

LDA is a parametric method that assumes unimodal Gaussian likelihoods. When the distributions of the data are non-Gaussian, the LDA projection may not preserve the complex structure in the data set that is needed for classification. Quadratic Discriminant Analysis (QDA) still assumes that the sample instances from each class are Gaussian distributed, but it does not further assume that the covariance of each of the classes is identical. It can be derived that in this case the decision boundary is no longer a line (or a plane, hyper-plane), but is a quadratic function. Generally speaking, QDA fits the data better than LDA, but it also has more parameters to estimate.

The Naïve Bayes classifier is another popular classification algorithm. It computes the posterior probability using the Bayes' Rule:

$$p(y|x) = p\left(y\middle|x_1, x_2, \dots, x_p\right) = \frac{p(y)p(x_1,x_2,\dots,x_p|y)}{p(x_1,x_2,\dots,x_p)} \quad \textbf{(2.5)}$$

where $p(x_1, x_2, \dots, x_p)$ is a normalization constant that does not affect y, so it can be omitted when determining the class y. The Naïve Bayes classifier assumes that the attributes $x_1, x_2, \dots, x_p$ are independent of each other, and the likelihood term can be simplified as

$$p\left(x_1, x_2, \dots, x_p\middle|y\right) = \prod_{i=1}^{p} p(x_i|y) \quad \textbf{(2.6)}$$

In the training, the likelihoods $p(x_i|y)$ can all be estimated from the relative frequencies in the training set, and these are the maximum likelihood estimates of the probabilities. The prior $p(y)$

can be similarly estimated from the relative frequencies, or can be assumed as equiprobable classes. When classifying a new sample, the output class is the one that maximizes $p(y)\prod_{i=1}^{p} p(x_i|y)$. The major advantage of the naïve Bayes classifier is its low computational complexity for both training and testing. Its major shortcoming is the strong independence assumptions of the features. From a graph point of view, the Naïve Bayes classifier has the form of the graph in Figure 2.14. The features $x_1, x_2, \ldots, x_p$ are as if independently generated from the underlying class label y. The Bayesian Networks try to reduce this strong independence assumption and take more general graphical models of the generative process. A Bayesian Network is a directed acyclic graph (DAG) in which a node represents a random variable and an edge represents a conditional dependency relationship between two nodes, in the sense that a node is conditionally independent with its non-descendants given its parents. A node can represent the random variable of the class label or a feature as shown in Figure 2.14, or it can be a hidden random variable (i.e., a latent variable). Probabilistic parameters are encoded into a set of conditional probability tables, one for each node. With such a generative model defined, the classification problem can be solved by maximizing the posterior probability of the class label conditioned on the observed data. With the network structure known (as can be specified by an expert) or unknown, the Bayesian Network learning algorithms

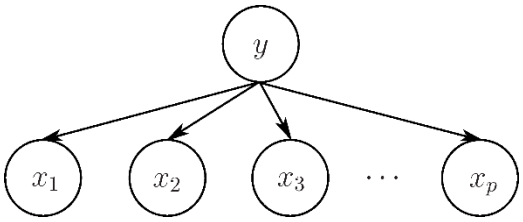


Figure 2.14 The graphical representation of the Naive Bayes classifier.

can be categorized into two classes. A review of the learning algorithms for Bayesian Networks can be found in [42].

Another category of statistical classification algorithms is the instance-based learning algorithms. They are the so-called "lazy-learning" algorithms [43], in which the induction process is delayed until the classification is performed. Compared to the "eager-learning" algorithms such as decision trees and Bayesian Networks, instance-based learning algorithms require less computation time in the training stage but more computation time in the classification stage. Nearest neighbor search is a basic instance-based learning algorithm, which simply finds one nearest neighbor (with a pre-defined distance metric) of the new input instance in the database of training samples with known labels, and assigns its label to be the classification output. More generally, k-Nearest Neighbor (kNN) searches for the k nearest neighbors instead of one, and uses the majority label of the neighbors to be the output. kNN is based on the principle that instances within the same class usually exist in close proximity to each other, or the manifold of a class in the feature space is smooth. The instance-based algorithms are intuitive and simple, while their main disadvantage is the large computational complexity for nearest neighbor search in a large database. Even with fast approximate nearest

neighbor search algorithms [44], the computation is still expensive, especially when the classification task is executed as a routine.

2.3.4 Support Vector Machines

The intuitive idea of the Support Vector Machine (SVM) is that among the many possible separation hyper-planes of two classes in the feature space, it is possible to find an optimal one that maximizes the margin between the two classes, as shown in Figure 2.15. Maximizing the margin has been proven to reduce the upper bound of the expected generalization error of the classifier [45]. For linearly separable training dataset $\{(\boldsymbol{x}_i, y_i), i = 1, \dots, n, y_i = \pm 1\}$, there exists hyper-planes specified by $(\boldsymbol{w}, b)$ such that $\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1$ for $y_i = 1$, and $\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1$ for $y_i = -1$. For the separation hyper-plane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$, the margin between the two classes is $\frac{2}{\|\boldsymbol{w}\|}$. The training points on the two margin hyper-planes $\boldsymbol{w} \cdot \boldsymbol{x} + b = \pm 1$ are called the "support vectors". Then, finding the optimal separation hyper-plane for the two linearly separable classes is formulated as the following optimization problem,

$$\begin{aligned} &\underset{(\boldsymbol{w},b)}{\operatorname{argmin}} \|\boldsymbol{w}\| \\ &\text{s.t. } y_i \cdot (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \end{aligned} \tag{2.7}$$

which can be solved by Quadratic Programming. When the original training dataset is not linearly separable due to mis-classification error, a soft margin can be used to accept some mis-classifications of the training instances

$$\begin{aligned} &\underset{(\boldsymbol{w},b)}{\operatorname{argmin}} \|\boldsymbol{w}\| \\ &\text{s.t. } y_i \cdot (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi, \xi \geq 0 \end{aligned} \tag{2.8}$$

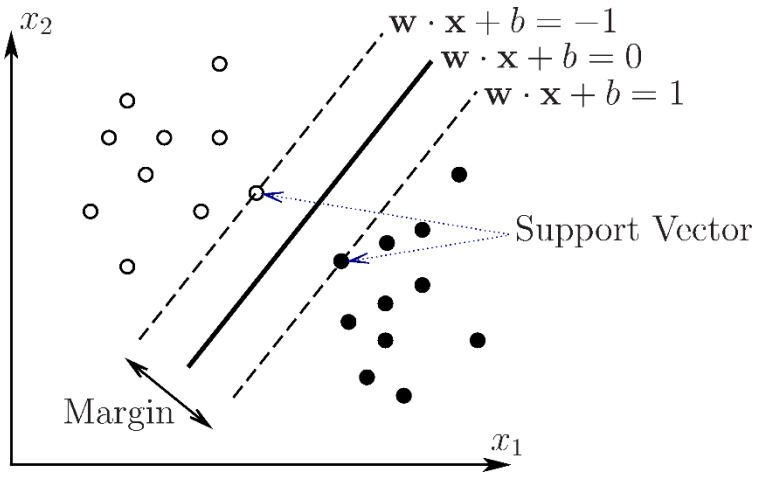


Figure 2.15 An intuitional illustration of the linear SVM.

In most real-world problems, the positive and negative training samples are non-separable due to the inherent non-linear data nature, but they can become separable once they are transformed into another feature space. A linear separation hyper-plane in the transformed feature space corresponds to a non-linear separation hyper-surface in the original feature space. If we denote such a transform as Φ, in the dual form of the training problem formulation the algorithm only depends on the inner product of the transformed data, i.e., $\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$. If there is a so-called "kernel function" K such that $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$, then only K needs to be evaluated for the training instances while not requiring to know the explicit form of Φ. Some popular kernel functions include polynomial kernel, Gaussian radial basis function kernel, and hyperbolic tangent (Sigmoid) kernel, etc.

2.3.5 Boosting

Boosting is a family of general meta-algorithms for supervised machine learning. Unlike the other algorithms described earlier in this section, boosting does not try to find sophisticated rules to classify the data, nor exploit complex patterns in the training data. Rather, it combines a set of weak classifiers to generate a strong classifier. One of the most popular boosting algorithms is the AdaBoost (Adaptive Boosting) algorithm developed by Freund and Schapire [46], which adaptively updates the weight for each training data samples based on the previous learned weak classifiers and focuses more on the "hard cases".

As described in [47], the AdaBoost algorithm is given in the following procedure.

Algorithm: AdaBoost from [47]
Input: training instances $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, where $y_i = \pm 1$

Initialize $D_1(i) = 1/n$
For $t = 1, \ldots, T$

- Train weak classifier using distribution D_t
- Get weak classifier $h_t(\boldsymbol{x}) = \pm 1$ with error

$$\epsilon_t = \Pr[h_t(\boldsymbol{x}_i) \neq y_i] = \sum_{i:h_t(\boldsymbol{x}_i)\neq y_i} D_t(i)$$

- Choose $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$
- Update weight:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ e^{+\alpha_t} & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$$

  where Z_t is a normalization factor such that D_{t+1} will be a distribution

Output the final classifier:

$$H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$$

In the training process, the initial weights for the training samples are set to uniform, and then the first weak classifier is trained from the weighted training samples. If the weak classifier learning algorithm does not support weighted samples, only a number of samples can be chosen according to the distribution specified by the weights, and only these samples can be used to train the weak classifier. ϵ_t is the weighted error for the weak classifier h_t, and $\epsilon_t = 0.5$ means that the classifier is equivalent to random guess. α_t measures the importance assigned to the weak classifier h_t based on its error. Note that $\alpha_t = 0$ if $\epsilon_t = 0.5$, i.e., a random guess classifier will not contribute to the final classifier; $\alpha_t > 0$ if $\epsilon_t < 0.5$, i.e., a better-than-random-guess classifier will have a positive contribution to the final classifier; and $\alpha_t < 0$ if $\epsilon_t > 0.5$, which means that a worse-than-random-guess classifier contributes negatively to the final classifier, while this is essentially the same as the $\alpha_t > 0$ case, because the negation of a worse-than-random-guess classifier is a better-than-random-guess classifier. Then, the weights for the training samples are either increased if h_t has a wrong prediction for a sample so that the next weak classifier can focus more on it, or decreased if the opposite is true. The algorithm iterates for T times and adaptively trains T weak classifiers, and the final classifier is a linear combination of the learned weak classifiers. It is noted in [47] that if each weak classifier is only slightly better than the random guess, the training error drops exponentially fast as T increases.

In this section we have reviewed some classification, or supervised machine learning algorithms, from a few basic and early approaches such as Decision Trees and Artificial Neural Networks to more recent approaches such as Support Vector Machine. In [48], an empirical comparison of supervised learning algorithms is provided. The algorithms with the best performance include SVM, Boosted Decision Trees, Bagged Decision Trees, and Random

Forests. Compared to the more complex approaches of Boosted Trees and Bagged Trees, SVM has the advantage that its design is simple (only involving the feature vector and the choice of an appropriate kernel function), and the implementation is relatively low-cost which is suitable for real-time operations.

2.4 Review of Tracking Algorithms

Object tracking is an important task within the research field of computer vision, especially for video analysis. Object tracking is often useful in the applications of motion-based recognition, video surveillance and suspicious activity detection, video indexing, human-computer interaction such as gesture recognition and eye gaze tracking, and traffic monitoring, etc. Tracking can be defined as finding the trajectory of a moving object in different frames of a video. The general tracking problem is challenging due to reasons such as complex object motions, complex object shapes and non-rigid objects, partial or full object occlusions, trajectory interference of different objects, scene illumination changes, etc. Based on different object representations, tracking algorithms can be roughly divided into three categories: point tracking, kernel tracking, and silhouette tracking, as noted in [49]. Point tracking uses a point representation of objects. Kernel refers to the object shape and appearance, such as a rectangular or an elliptical shape with certain color histograms. Silhouette tracking further tracks the change of the object shape either by shape matching or contour evolution. In this section we focus on point tracking, and review two popular tracking algorithms in these categories.

2.4.1 Kalman Filter

A Kalman filter is an optimal linear filter that recursively estimates the state of a dynamic system from a series of noisy measurements. It combines all the available measurements, plus any prior knowledge, and produces an estimate of the desired variables, while statistically minimizing the mean squared error. It assumes that the system is a linear dynamical system, and that any noise in the system is Gaussian. Since R.E. Kalman published his paper of a recursive solution to the discrete-data linear filtering problem in 1960 [50], the Kalman filter has been the subject of extensive research, finding successful applications particularly in navigation. Here we adopt the same notation as in [51]. Assume the underlying discrete-time controlled process is $x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$, where A relates the state at the previous time step to the state at the current step, B relates the optional control input u_{k-1} to the state x_k, and w_{k-1} represents the process noise, which is a Gaussian noise such that $p(w) \sim N(0, Q)$, i.e., the noise w is drawn from a zero-mean Gaussian distribution with covariance matrix Q, assuming the noise is time-stationary. The measurement process is described as $z_k = Hx_k + v_k$, in which H relates the state x_k to the measurement z_k. v_k is the measurement noise, which is also assumed to be Gaussian and $p(v) \sim N(0, R)$. The *a posteriori* state estimate $\hat{x}_k$ can be computed as the *a priori* estimate $\hat{x}_k^-$ plus an innovation term from the difference between the expected measurement $H\hat{x}_k^-$ and the actual measurement z_k:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \quad \textbf{(2.9)}$$

which is a weighted combination of the prior confidence about the state and the corrections from the new observation. By minimizing the *a posteriori* estimate error covariance $P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]$, the Kalman gain matrix K is given by

$$K = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad \textbf{(2.10)}$$

where $P_k^- = \mathrm{E}[(\mathrm{x_k} - \hat{x}_k^-)(\mathrm{x_k} - \hat{x}_k^-)^{\mathrm{T}}]$ is the *a priori* estimate error covariance. Kalman filter algorithm has two steps for each new time step, the time update (prediction) and the measurement update (correction). The time update projects the *a posterior* state estimate from the previous time step to the *a prior* state estimate for this time step, and the measurement update adds the influence from the new observation and updates the *a prior* state estimate to the *a posterior* estimate by Eq. (2-10). When the system dynamics or the measurement process is non-linear, they can be linearized by the Taylor series expansion, which is the extended Kalman filter (EKF). The Kalman filters have been successfully applied to various tracking problems such as point tracking in noisy images [52], people tracking [53], and object tracking with 3D trajectory [54].

2.4.2 Particle Filters

One major drawback of the Kalman filter is its strong assumption that the distributions of the state variables are Gaussian, thus it often has poor performance for non-Gaussian state variable cases. Particle filter does not rely on the Gaussian state variable assumption. More generally, tracking can be considered as a Bayesian estimation problem. Given the process model $x_k = f_k(x_{k-1}, w_{k-1})$, (w_{k-1} is the process noise, same as in the previous section) and the measurement model $z_k = h_k(x_k, v_k)$, (v_k is the measurement noise), tracking is to recursively calculate some degree of belief in the state variable x_k given the observations $z_{1:k}$. It can be done in two steps. The prediction step estimates the prior probability distribution function (pdf) of x_k:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|z_{1:k-1}) dx_{k-1} \quad \textbf{(2.11)}$$

The update step uses Bayes' rule to estimate the posterior pdf of x_k given the new measurement z_k:

$$p(x_k|z_{1:k}) = \frac{1}{Z}p(z_k|x_k)p(x_k|z_{1:k-1}) \quad \textbf{(2.12)}$$

Where Z is the normalization factor. For Kalman filter, all the probability functions in Eq. (2-11) and Eq. (2-12) have the form of a Gaussian function, so the closed-form solution can be derived. Other than the simple Gaussian case, Eq. (2-11) and Eq. (2-12) are usually intractable. What particle filter does is to sample the probability functions with a group of weighted samples, i.e. particles, and estimate the posterior probability density function based on these samples, making the otherwise intractable computation tractable.

At time step k, the posterior probability density function $p(x_{k-1}|z_{1:k-1})$ from the last time step is represented by a set of weighted samples (particles) $\boldsymbol{S}_{k-1} = \{(s_{k-1}^i, w_{k-1}^i): i = 1, \dots, N\}$, as shown in Figure 2.16. The weights define the sampling probability, or the observation frequency and the importance of a sample. The new samples at the time step k are then drawn from $\boldsymbol{S}_{k-1}$, and the most popular sampling method is the importance sampling [55]. First N random samples are selected from $\boldsymbol{S}_{k-1}$ by generating N random numbers in $[0, 1)$ and choosing the samples whose cumulative weights are closest to the random numbers. Then the N samples go through the "diffusion" process, which applies the system dynamic model f_k to the samples and serve as the prediction step. Lastly, the predicted samples $\{s_k^i: i = 1, \dots, N\}$ are weighted as measurement likelihoods given the new observation. The posterior probability density function $p(x_k|z_{1:k})$ is obtained in the form of $\boldsymbol{S}_k = \{(s_k^i, w_k^i): i = 1, \dots, N\}$, and any object property of interest, such as location, can be estimated from the posterior probability density function. For

other forms of particle filters such as auxiliary particle filter and regularized particle filter, the reader can find a good reference in [56].

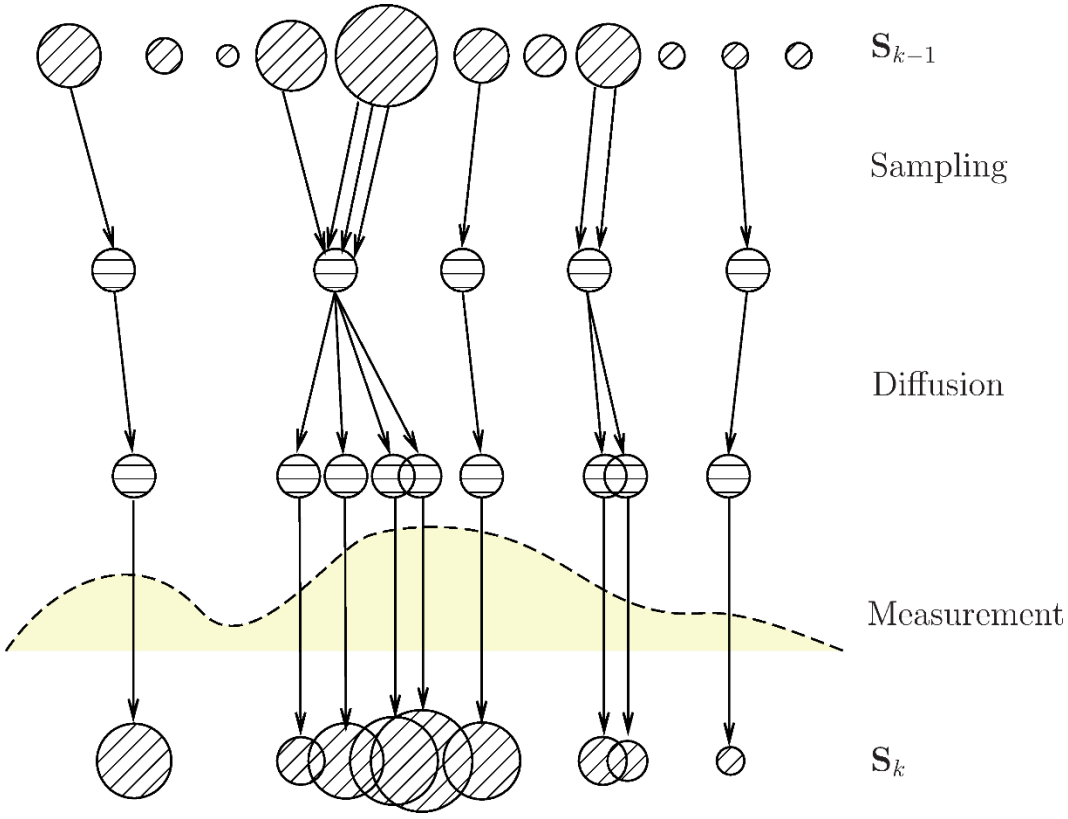


Figure 2.16 The process of particle filtering with importance sampling.

In [57], a particle filter tracking algorithm with the object represented by color distribution is presented. Color distribution has the good properties of being robust to partial occlusion, rotation and scale invariant, and computationally efficient. The algorithm adapts the object color distribution model throughout tracking. In [58], an appearance-adaptive model is developed for simultaneous particle filter tracking and object recognition. Yang et al. [59] proposes to use color and edge orientation histogram features as the object representation model, and the observation

likelihood is computed in a hierarchical coarse-to-fine manner. Mixture particle filter [60] deals with multi-target tracking as it assigns a mixture component to each target. Khan et al. [61] further incorporates a Markov Random Field (MRF) motion prior to model complex motion patterns for multiple interacting targets. The work in [62] places an object detector in the framework of particle filter tracking and achieves tracking-by-detection, since the object detector confidence output can serve as the measurement likelihood function.

Compared to the Kalman filter, particle filter has the advantage that it imposes less constraint on the problem, i.e., it does not assume Gaussian state variable and white Gaussian noise and can tackle practical problems with more general situations. Because of the sound theoretic ground and the relatively low computational cost, it has become one of the most successful object tracking approaches. The underlying principle of Monte Carlo sampling has also found many applications in various disciplines of modern science and engineering.

Chapter 3 An Automatic Power-Line Detection Algorithm

In this chapter, we describe our first developed power-line detection algorithm, which includes a pre-processing step, a straight-line detection algorithm, a classifier, and an adaptive post-processing algorithm. We also present the simulation results to show its performance.

3.1 Pre-processing

As mentioned earlier in Chapter 1, the power lines are straight-lines in the radar video. However, in Figure 2.5 the power lines are curves rather than straight lines. This is because the radar images are in B-scope display, i.e., range vs. angle, or polar plot. After a simple coordinate transformation from polar to Cartesian coordinate system, the power lines will appear as straight lines. Since pixels on the power lines typically have stronger intensities than their surrounding pixels, we first threshold the raw amplitude data into a cleaner image. The thresholding is done slice-wise, with the slice height being $1/16$ of the frame height, as an adaptive thresholding scheme. We keep a certain percentage of pixels in each slice, typically the top $2\% - 5\%$ pixels with the strongest intensities, and we also keep their original values for these surviving pixels. In simulations, we find that the system performance is not sensitive with respect to the threshold percentage. To save computations, we keep the top 2% of pixels in each slice. Then we process this sparse image by coordinate transformation to make power lines appear straight. In Figure 3.1, we show a frame with the results after the pre-processing step. We can see that the pre-processing can remove the noise while keeping most of the power line pixels, and also converts the power line curves into straight line-segments.

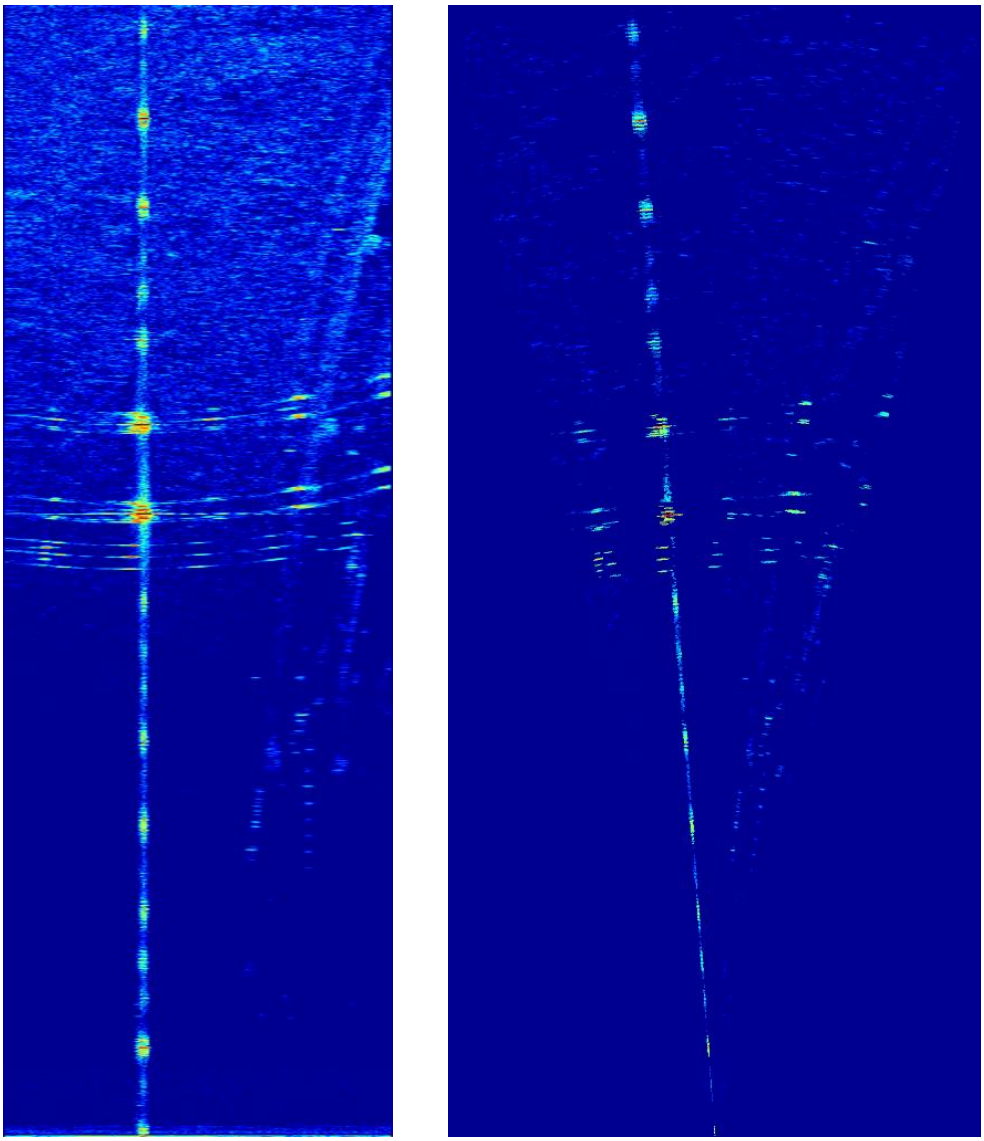


Figure 3.1 A frame and the result after the pre-processing step.

After the pre-processing step, we propose to divide each frame into a number of overlapped horizontal slices, and perform Hough transform and straight-line detection for each slice. The benefits of such a slice-processing algorithm include:

1. It gives the power-line detection algorithm additional robustness in the presence of noise. When the ground return causes clutter noise, different regions in a frame will have different data characteristics, as often the ground return noise will only appear in a certain region in the frame. Because of this inhomogeneous data nature, it is reasonable to consider different regions separately and process them separately. On the other hand, if we take the full frame and process all the data together, if the algorithm is designed to better accommodate for noisy regions, it may not work well in "clean" regions; and vice versa. In summary, the slice-processing scheme can be considered a locally adaptive algorithm so that it can work well in various conditions.
2. It makes parallel processing readily applicable for real-time operations. Processing speed is an important issue for this application because it is required to work in real-time. When we divide a big frame into smaller slices, multiple threads can process the slices simultaneously, which makes it suitable for parallel implementations.

The slice processing algorithm can only detect segments of a power line if the power-line crosses multiple slices. In our power-line imaging system, any power line off the horizontal direction by more than 30 degrees will diminish, thus, we do not need to take special care of the lines that are less horizontal. To detect the complete power lines that lie on the boundary of two slices instead of detecting the segments of the power line in each slice and trying to connect the segments, we introduce some overlapping between neighboring two slices. Figure 3.2 explains this idea. The entire frame is divided into overlapping slices as shown in the figure. Hough transform and line detection are performed within each individual slice. Without overlapping, power lines that lie on the boundary of two slices may be missed or be divided into two segments.

With overlapping introduced, this case can be taken care of. The overlapped slice processing will incur some extra computation, but the advantage by parallel processing can be more significant.

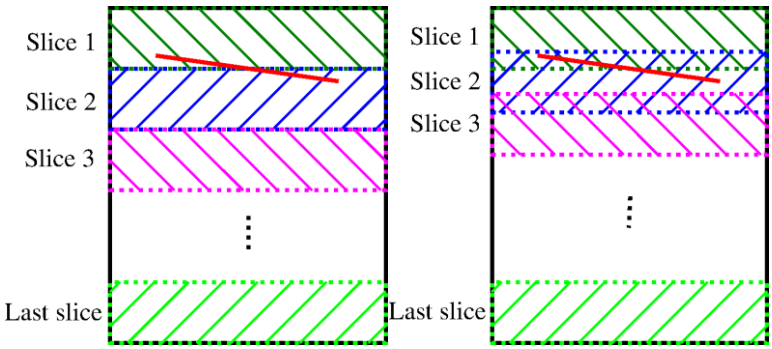


Figure 3.2 Slice processing algorithm. Left: without overlapping; right: with overlapping.

3.2 Using Hough Transform and SVM for Detecting Power Lines in a Noisy Environment

Since the power lines are straight lines in the radar video, using a straight-line detection algorithm to detect them would be a natural approach. Among the various straight-line detection approaches we have reviewed in Section 2.2, we use the Hough transform for detecting the power lines, for the following reasons:

1. Generality. The only line feature that Hough transform requires is that the lines are straight, not constraining any other additional line characteristics such as the width of the line, the intensity of the line, etc., which are often utilized in other heuristic curvilinear structure detection algorithms. The straight-line property always holds true for the power lines in the radar video, while their widths and intensities might vary under different conditions. Thus, Hough transform is particularly suitable for our application.

2. Simplicity. The intuitional idea of the Hough transform is straightforward, and its implementation is also simple compared to other algorithms. Its characteristics are well understood and its performance is well tested.
3. It can be extended to better suit for the power-line detection in our application. Standard Hough transform assumes a binary input image. In our case the power lines usually have stronger intensities than their surrounding pixels because of the radar wave's strong reflection from the metal power-line surface. It is clear that we can use a weighted Hough transform where the vote from each pixel is proportional to its intensity value, so that in the final Hough transform data, the power lines which have more pixels and stronger intensities could stand out better.

Although with these good properties, in our case simple Hough transform line detection will not work well. As can be seen in Figure 2.5 and especially the second frame of Figure 2.6, the power-line image is very noisy. The power lines are often "hidden" in the background noise, due to the strong clutter signal from the ground return. The signal magnitude of ground return is often comparable with that of the power lines. Many noise points can fall on the same straight line, which results in peaks after performing the Hough transform. On the other hand, if we identify fewer peaks after the Hough transform, certainly we could detect fewer noise lines – but we may also miss some real power lines. Thus, directly using Hough transform may not be sufficient to differentiate the power lines and the lines formed by the noise points.

To solve this problem, we propose to utilize a classifier in addition to the Hough transform for the power-line detection. The idea is to use a two-step process to solve two simpler problems. The first step is using the Hough transform to detect line "candidates" – which can be either true

power lines or noise lines. As we do not want to miss any true power lines, we use a strategy to detect as many candidate lines as possible in this step, and the candidate lines could also include some noise lines. In the second step, we use a pre-trained classifier to classify each candidate line into one of the two categories: a true power line, or a noise line. A noise line is discarded, while a true power line is kept.

Among the classifier algorithms we reviewed in Chapter 2, we use the Support Vector Machine (SVM) as the classifier to classify true power lines and noise lines. SVM has been used successfully in many different kinds of applications. It can handle linear and non-linear discriminating boundaries between the classes. It is robust and its good performance has been well documented. During the testing phase, it requires little computations, and thus is suitable for real-time applications. In fact we will see in Section 3.6.1 that SVM performs very well for the power-line classification problem, and it meets our real-time processing requirement.

3.3 Features to Represent the Bragg Pattern for the SVM Classifier

As noted by previous works, and also can be seen in our power-line imaging results in Figure 2.5 and Figure 2.6, the Bragg pattern is a characteristic pattern of the power lines which can be used to differentiate the power-lines from the noise lines. Specifically, the Bragg pattern is the quasi-periodic pattern of the power lines that have periodic peaks in the radar signal return. The Bragg pattern is formed for power lines, since they have periodic surface structure due to their braiding structure as shown in Figure 2.7. As the wavelength of the millimeter-wave radar is in the same order with the surface period L of power lines, according to Bragg's Law of Diffraction, peaks will be formed in the radar image at locations described by Eq. (2.1). In Figure 3.3, we

show some examples of the radar signal data of candidate lines. In the left column are the pixel values of the points on the true power lines, and in the right column noise lines. We can see that the power-line data have some clear quasi-periodic peaks due to the Bragg pattern, which are not present in the noise-line data. The data on the noise lines appear more random and lack a structure. The different characteristics of the data will appear both in the spatial domain and in the frequency domain. In the spatial domain, the data values of the power-lines are usually larger due to the relatively strong return of the radar signal from the power lines. In Figure 3.4, we show an example of the histogram of the data on a power line versus a noise line, and we can see that the power line data has a distribution that is more like a Gaussian distribution, while the noise line data is quite random. Also, as shown in Figure 3.5, the power-line data frequency spectrum has many low-frequency components, which is quite different from that of the noise line data.

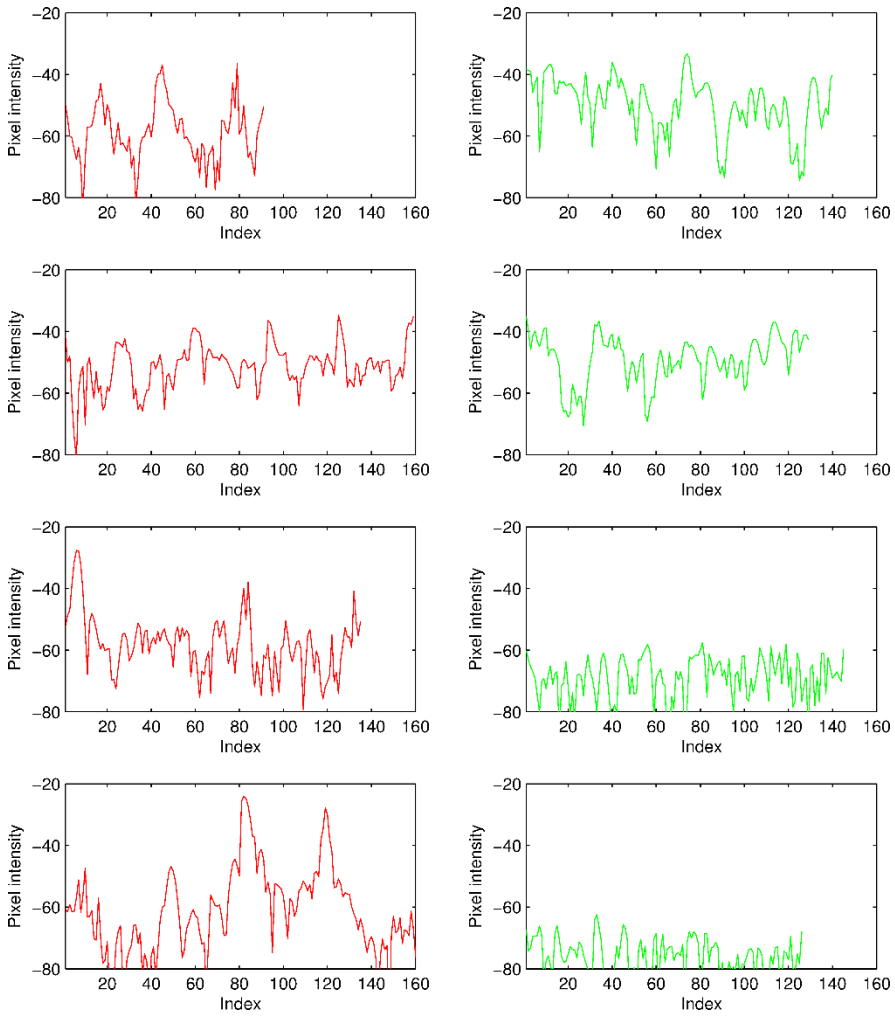


Figure 3.3 Some examples of power line data and noise line data.

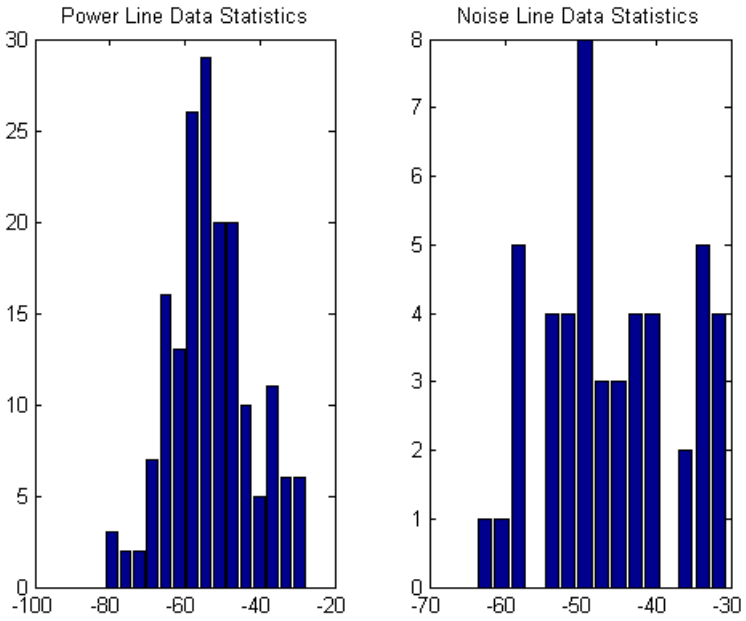


Figure 3.4 An example of power line data histogram vs. noise line data histogram.

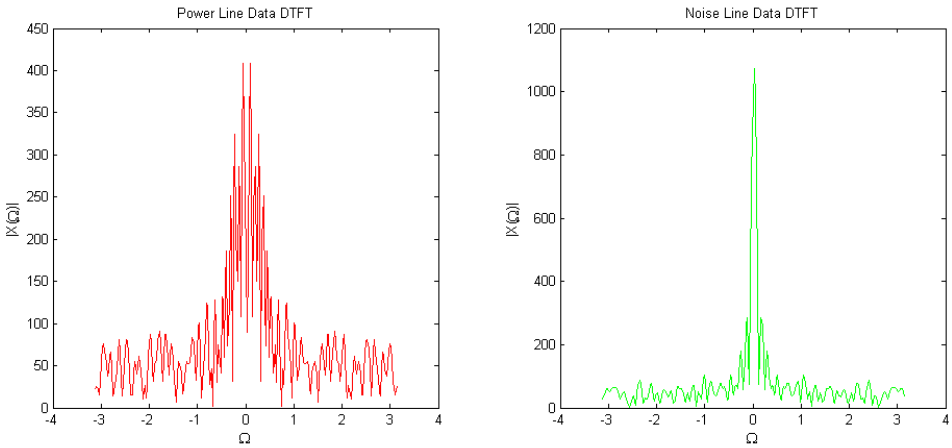


Figure 3.5 An example of power line data spectrum vs. noise line data spectrum.

Based on these observations, we propose a 14-dimensional feature vector to represent each candidate line data. This feature vector is devised in the effort of representing the periodic Bragg pattern in both the time domain and the frequency domain meaningfully. In the following description, $\boldsymbol{v} = (\boldsymbol{v}(1), \boldsymbol{v}(2), \dots, \boldsymbol{v}(n))$ represents the data on a candidate line, i.e., the array of pixel intensities of all the pixels on a particular line, and n is the number of pixels on that line. Let $\boldsymbol{x}$ denote the feature vector to be extracted from $\boldsymbol{v}$, and $\boldsymbol{x}$ is a 14-dimensional vector.

The first feature group is some basic statistics on the values of the line data:

$$\boldsymbol{x}(1) = \overline{\boldsymbol{v}} \quad \textbf{(3.1)}$$

$$\boldsymbol{x}(2) = \max(\boldsymbol{v}) \quad \textbf{(3.2)}$$

$$\boldsymbol{x}(3) = \text{percentile}(\boldsymbol{v}, 95) \quad \textbf{(3.3)}$$

$$\boldsymbol{x}(4) = \text{percentile}(\boldsymbol{v}, 68) \quad \textbf{(3.4)}$$

where $\overline{\boldsymbol{v}}$ represents the mean value of the data in $\boldsymbol{v}$, $\max(\boldsymbol{v})$ represents the maximum value in $\boldsymbol{v}$, $\text{percentile}(\boldsymbol{v}, 95)$ represents the 95 percentile value of the data in $\boldsymbol{v}$. We used the 68% and 95% percentile because in a Normal distribution, 0.68 and 0.95 are the probabilities that the random variable falls within one and two standard deviation from the mean value, thus, they can well characterize the general distribution of the pixel intensities on a line. The next feature group is a description of the periodic Bragg pattern in the spatial domain. According to Eq. (2.1), the peaks of the returned radar signal are evenly located on the line. As an approximate way to identify the peaks, we threshold the line data with $\boldsymbol{x}(3)$, keeping only the top 5% pixels on a line. We measure the number of connected segments after thresholding, n_s, as an approximation to the number of the signal peaks, to be feature $\boldsymbol{x}(5)$. We also compute the distance $d_1, \dots, d_{n_s-1}$

between the center points of these segments. The following two features are statistics of the distances between the peaks:

$$\boldsymbol{x}(6) = \bar{d} \tag{3.5}$$

$$\boldsymbol{x}(7) = \hat{\sigma}(d) \tag{3.6}$$

where $\hat{\sigma}$ represents the standard deviation. The two features depict how the signal peaks are spatially placed. For the power-line data, the distances between the signal peaks shall be roughly a constant as in Eq. (2-1), so $\boldsymbol{x}(6)$ shall be roughly a constant and $\boldsymbol{x}(7)$ shall be small. For noise lines these features are irregular. Since the signal peak identification by thresholding is only an approximation, for completeness, we repeat the feature group $\boldsymbol{x}(5) - \boldsymbol{x}(7)$, but with a different threshold $\boldsymbol{x}(4)$, to get feature $\boldsymbol{x}(8) - \boldsymbol{x}(10)$.

Next we have a group of features to represent the periodicity of the Bragg pattern in the frequency domain. The first two of them are:

$$\boldsymbol{x}(11) = \frac{\sum_{i=2}^{5} |\boldsymbol{V}|^2(i)}{|\boldsymbol{V}|^2(1)} \tag{3.7}$$

$$\boldsymbol{x}(12) = \frac{\sum_{i=6}^{1024} |\boldsymbol{V}|^2(i)}{|\boldsymbol{V}|^2(1)} \tag{3.8}$$

where $\boldsymbol{V}$ is the 2048-point DFT of $\boldsymbol{v}$. $\boldsymbol{x}(11)$ represents the ratio of the low-frequency harmonic power to the DC power, while $\boldsymbol{x}(12)$ represents the ratio of the remaining high-frequency harmonic power to the DC power. The quasi-periodic nature of the power line data will result in more low-frequency power, while the white-noise nature of noise line data will not.

The last two features, $\boldsymbol{x}(13)$ and $\boldsymbol{x}(14)$, are the median and mean value of the autocorrelation function of $\boldsymbol{v}$ which can also help distinguish the power lines and noise lines.

As will be shown in later sections, this set of features gives satisfactory performance. The feature vector $\boldsymbol{x}$ is fed into the SVM classifier. Let $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_n, y_n)$ be the training samples, where y_i is the label associated with a feature point $\boldsymbol{x}_i$, being $+/-1$, as $y_i = 1$ denotes that $\boldsymbol{x}_i$ is a feature vector from a power line, and $y_i = -1$ denotes that $\boldsymbol{x}_i$ is a feature vector from a noise line. The total number of training samples is n. The training problem can be formulated in the following dual-problem format

$$\min_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j) - \sum_{i=1}^{n} y_i\alpha_i \quad \text{s.t.} \begin{cases} \sum_i \alpha_i = 0 \\ 0 \le y_i\alpha_i \le C \end{cases} \tag{3.9}$$

where $D(\boldsymbol{\alpha})$ is the objective function we try to minimize, and C is a trade-off coefficient between maximizing the margin and minimizing the training error. $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j)$ is the kernel function, as a dot product of the two feature vectors mapped by Φ. We use the Gaussian radial basis function

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2) \tag{3.10}$$

which can give better performance. This optimization problem can be solved by Quadratic Programming.

After the training is completed and we have the coefficient vector $\boldsymbol{\alpha}$, the decision function of the classifier is

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{3.11}$$

where b is the offset term and it can be written as

$$b = \frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{n}\alpha_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) - y_i) \tag{3.12}$$

which is an average offset over all the training samples.

3.4 An Adaptive Algorithm for Generating Frame-wise Decision

In the development of the power line detection algorithm, a problem often encountered is that after a classifier is trained, new factors that could affect the performance are often identified. To take into consideration of these new factors, the classifier needs to be re-trained, and new training samples often need to be collected, which is rather tedious. Also, the parallel power line and temporal correlation properties have not been utilized in the algorithm. To address these issues, we propose an adaptive probability adjustment approach to take care of the factors that are not taken care of by the SVM classifier, where the probability denotes the confidence we have for a frame to contain power lines. With the adaptive probability adjustment approach, this frame-wise probability will be automatically adjusted instead of the tedious retraining of the SVM classifier. Also, the parallel power line and temporal correlation properties will be taken care of.

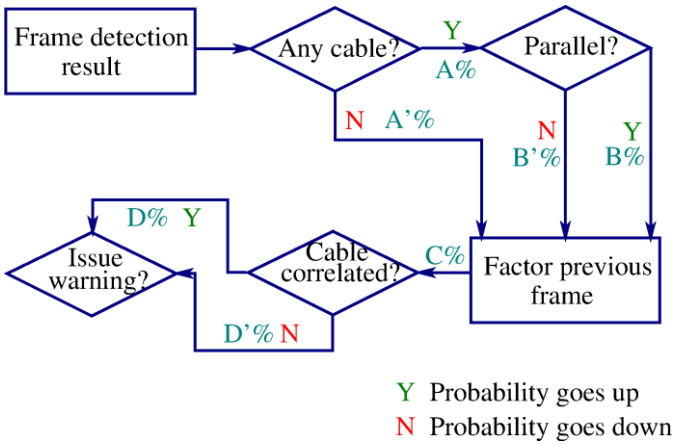


Figure 3.6 An adaptive algorithm to generate a frame-wise probability of containing power lines from power line detection results.

As shown in Figure 3.6 , the adaptive frame probability computing algorithm takes into consideration of power line parallelism, temporal smoothness, and temporal correlation between the power lines. $A\%$, $A'\%$, $B\%$, $B'\%$, $C\%$, $D\%$, and $D'\%$ are estimated probabilities for a frame to contain power lines. An initial probability (or score, we use the two terms interchangeably in this section) is set according to whether there are power lines detected or not in one individual frame. This initial probability can increase or decrease depending on the number of parallel power lines detected. Temporal smoothing is achieved by taking the average of current probability and the probability from previous frames. Lastly, the probability is further modified depending on the power-line correlation between the two frames – if we can find "correlated" power lines in the two frames, the frame probability will increase; otherwise, decrease. If the final probability is greater than a certain threshold such as 0.5, the system will generate a warning signal to the pilot and it will report the locations of the detected power lines and overlay them upon the radar display image.

The correlation between power lines from two neighboring frames is an important issue. The rationale is that in the radar video, considering that the helicopter is flying at a limited speed, the power line location in the radar video for the same physical power line shall not vary too much from one frame to the next. When the preprocessing algorithm and the classifier successfully detects a true power line in one frame, it is very likely that in the next frame the same power line will appear in some nearby region. When the classifier makes a mistake and generates a false alarm power line in one frame, it is less likely that in the next frame the same false alarm happens in a nearby place since noise lines are usually irregular. From this correlation, we can reduce the false alarm rate of the classifier. The SVM and the features do not consider the

information of power line correlation in neighboring frames; rather we encode this information in this adaptive algorithm. We propose a simple model to carry out this idea and define whether one power line is correlated with another in a neighboring frame. Specifically, in addition to the SVM, we also fit a linear model to give the "score" of each candidate line by Least Squares:

$$\boldsymbol{\beta} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} \quad \textbf{(3.13)}$$

where $\boldsymbol{X}$ is the features from all the training data concatenated into a matrix, $\boldsymbol{y}$ is the training data label vector, and $\boldsymbol{\beta}$ is the linear combination coefficient vector for the features to compute the score for a candidate line:

$$s = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x} \quad \textbf{(3.14)}$$

where $\boldsymbol{x}$ is the feature vector extracted from the data on a candidate line. This simple score gives an approximate probability for a candidate line being a power line. For a candidate line $l_i^{(k)}$ classified as the ith power line by SVM in frame k, we search for the closest power line $l_i^{(k-1)}$ from the detection result of the previous frame $k-1$:

$$l_i^{(k-1)} = \max_j \left\{ \exp\left(-\frac{\left(\rho_{l_i^{(k)}} - \rho_{l_j^{(k-1)}}\right)^2}{2\sigma_\rho^2} - \frac{\left(\theta_{l_i^{(k)}} - \theta_{l_j^{(k-1)}}\right)^2}{2\sigma_\theta^2} \right) \right\}, \text{s.t.} f\left(p_{l_j^{(k-1)}}\right) > 0 \quad \textbf{(3.15)}$$

where ρ and θ are the two parameters associated with a straight line, namely, its distance to the origin and its orientation. f is the decision function of the SVM, so we will only search for lines in the previous frame that have been classified as power lines. The "closeness" is depicted by a Gaussian function. Then we modify the score $s_{l_i^{(k)}}$ by the following equation:

$$s'_{l_i^{(k)}} = s_{l_i^{(k)}} * \exp\left(-\frac{(\rho_{l_i^{(k)}} - \rho_{l_i^{(k-1)}})^2}{2\sigma_\rho^2} - \frac{(\theta_{l_i^{(k)}} - \theta_{l_i^{(k-1)}})^2}{2\sigma_\theta^2}\right) \quad \textbf{(3.16)}$$

where $s'_{l_i^{(k)}}$ is the updated score for the power line $l_i^{(k)}$. Depending on the detected power line $l_i^{(k)}$ in frame k has a close neighbor power line in the previous frame or not, its score $s_{l_i^{(k)}}$ can increase or decrease. We will reject any power line with un updated score lower than 0.5.

The temporal correlation can not only boost the classifier performance, but also save computation. Since the true power lines cannot jump randomly from one frame to the next, when we have detected some power lines with high confidence, we have a prior knowledge about where the power lines are most likely going to appear in the next frame. We can then focus on just that area and save computation in the next frame. With the slice processing algorithm, this can be easily achieved.

As a summary, in the adaptive frame score computation algorithm shown in Figure 3.6 , we not only consider the power line detection results from one frame and whether there are parallel power lines, but also from previous frames' history, as whether there are correlated power line pairs between the two neighboring frames, and the frame score is adjusted accordingly. With this process, a more reliable and robust frame-wise decision (determining if a frame contains power lines or not) for issuing a warning signal can be obtained, especially in the face of strong clutter background noise.

3.5 The Overall Power Line Detection Algorithm

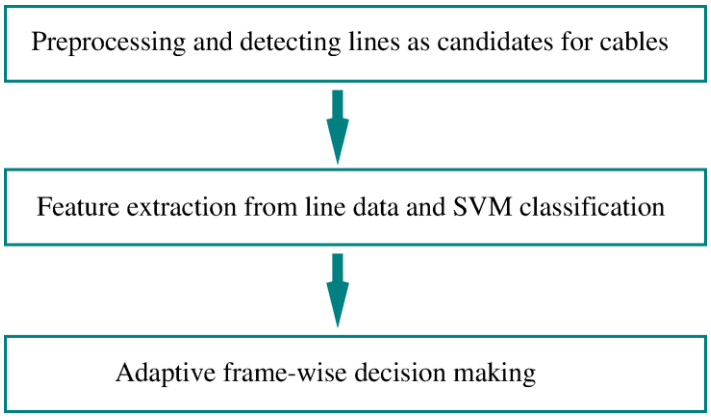


Figure 3.7 The overall power line detection system flowchart.

Figure 3.7 shows the general flow of our power-line detection algorithm. Before the power line classification takes place, we first identify possible locations of the power lines and retrieve data on the candidate lines. Hough transform is used for candidate line detection in each of the overlapping slices. Before the Hough transform, pre-processing of the radar video is applied, as the thresholding and coordinate transform procedures described in Section 3.1. For Hough transform, instead of using the black-and-white image where the vote of each pixel is the same, we use weighted Hough transform in which the vote of each pixel is its pixel value. The weighted Hough transform has better capability of detecting bright lines, in our case being the power lines. The overlapping between two slices is a quarter of the slice height. The resolution of the parameter space for the Hough transform is that

$$\begin{aligned} \Delta\rho &= 1 \text{ pixel} \\ \Delta\theta &= 1^\circ \end{aligned} \tag{3.17}$$

In order not to miss any power lines, we detect multiple lines in each slice. The slice height is chosen such that within one slice there usually would be no more than one group of power lines, e.g., three power lines in a close range. So we take the top 3 peaks in the Hough transform data

as the 3 candidate lines. The candidate lines will contain many noise lines, since in reality, not every slice contains power lines. It is the classifier's duty to filter out the noise lines. The feature vector described in Section 3.3 is computed using the data retrieved from a candidate line, and it is fed into the pre-trained SVM to classify the candidate line.

The detection results from one frame, in the form of the power line parameters such as ρ, θ, s, together with the results from the previous frame, are then fed into the adaptive algorithm shown in Figure 3.6 to compute a frame probability and make a final warning signal issuing decision. The initial probability is set according to whether there are power lines detected by the SVM in this frame:

$$\begin{cases} A = 50\%, \text{if there are power lines detected;} \\ A' = 0\%, \text{if there are no power lines detected.} \end{cases} \quad \textbf{(3.18)}$$

We set A only to 50% because there is some chance that the detected power lines are false alarm noise lines. We depend on other factors, such as parallel power line pairs and temporally correlated power line pairs to adaptively build up the confidence. If there are power lines detected, we further check whether there are parallel power lines. Parallelism is defined as having the same θ in the Hough transform parameter space. Let n_p be the number of parallel power lines, i.e., the number of power lines with the same θ. After straight line detection and SVM classification, we get a number of power lines for a frame. We group these power lines according to their θ value, and the number of power lines in the largest group would be n_p. Depending on the value of n_p, A will be further modified to give B or B':

$$B' = A \times 1, \text{if } n_p = 1,$$
$$B = A \times \begin{cases} 1.2, & \text{if } n_p = 2 \\ 1.3, & \text{if } n_p = 3 \\ 1.4, & \text{if } n_p = 4 \\ 1.5, & \text{if } n_p = 5 \text{ or more} \end{cases} \quad \textbf{(3.19)}$$

So the more parallel power lines we can find, the more confidence we have that they are true ones, and the more will the probability increase. Note that even if we do not find any parallel power lines, we will not penalize the probability, since in reality there are often cases when there is just one power line visible in one frame. Then we take the average of the current probability, either being A', B', or B, and the previous frame's final probability D_p:

$$C = (A', or\ B, or\ B' + D_p)/2 \quad \textbf{(3.20)}$$

The averaging serves as a temporal smoothing procedure. We also take care of the correlated power lines among this frame and the previous frame. We carry out Eq. (3.15) and Eq. (3.16) for each power line in the current frame, finding out the closest power line from the previous frame and alter the line score according to their closeness. We then find out the number of power lines that are correlated to some power lines from the previous frame, n_c. n_c is the number of power lines in the current frame that will have an increased score after evaluating Eq. (3.16). Depending on the value of n_c, we further modify the frame probability according to

$$D' = C \times 2/3, \text{if } n_c = 1,$$
$$D = c \times \begin{cases} 1.2, & \text{if } n_c = 2 \\ 1.3, & \text{if } n_c = 3 \\ 1.4, & \text{if } n_c = 4 \\ 1.5, & \text{if } n_c = 5 \text{ or more} \end{cases} \quad \textbf{(3.21)}$$

The idea is that the more correlated power line pairs we can find, the more confidence we have for them being real power lines. We choose these numbers since in simulations after testing with

different parameters these values generally give us a reasonably good performance. In the end the frame probability is clipped to 1 if it is greater than 1.

3.6 Simulation Results

3.6.1 Data Collection and Classifier Training

For this simulation, we have 6 datasets with different data nature as listed in Table 3.1. These datasets represent a broad range of different cases that might happen in reality. We manually label each frame as "1" – power line present, or "0" – power line absent.

Table 3.1 The 6 Datasets of Power Line Imaging Results for Algorithm Testing

Dataset #	Nature of data	# of frames	Frame size
1	Stationary hover at 350m to cables	49	2048×174
2	Flight toward cables at 450m	147	2048×174
3	Flight above cables with elevation scanning mode from 500m	148	2048×174
4	Flight toward cables with elevation scanning mode from 400m	147	2048×176
5	Flight toward cables at 600m	97	4096×175
6	Flight above cables with fixed pointing from 450m	149	4096×174

For training the SVM, we manually collected a ground truth set containing 515 power lines and 668 noise lines. We carried out cross-validation with it. In Figure 3.8, we plot the average training accuracy and the average testing accuracy vs. training set size. The training set size is

proportional to the total ground truth set size. The training accuracy is high for all sizes, indicating that the feature vector is descriptive enough to represent almost every data point in the training set correctly. However the testing accuracy varies because it might suffer from either under-fitting or over-fitting, and a good balance between the two has to be achieved by choosing a good training set size. The testing accuracy reaches its highest point when the training set size is around $\frac{2}{3}$ of the total ground truth dataset, and we have an average testing accuracy of 95.5%. Thus we select a training set with two thirds of the data in the ground truth set, and the SVM classifier is trained using this training set.

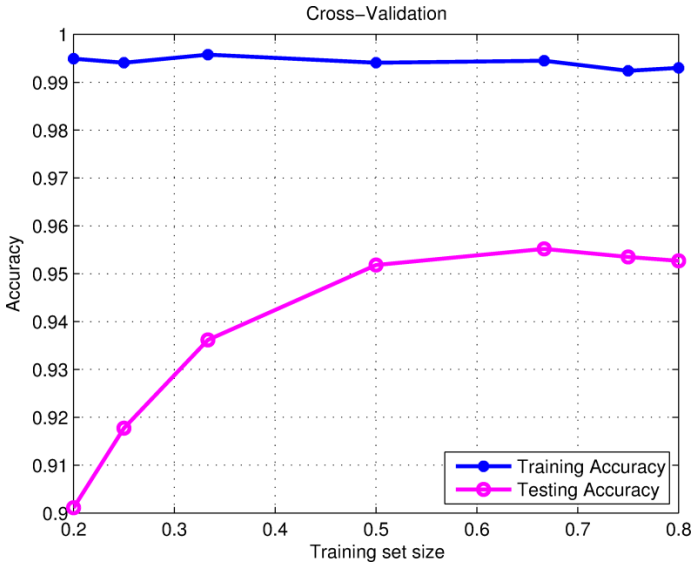


Figure 3.8 Cross-validation results.

3.6.2 Feature Selection

In section 3.3 we have proposed a 14-dimensional feature vector to represent the data on a candidate line. Though the dimensionality of the feature vector is not overwhelmingly high, it is desirable to select a more compact set from the feature vector for computational efficiency.

Moreover, selecting the characterizing subset from the feature can often improve classification accuracy, since the "noisy" features can be removed by feature selection. Furthermore, investigating the performance of the classifier using subsets of the feature vector can give us more insight of the meaningfulness of the different features, and thus helping designing more characterizing features if necessary.

To select the characterizing feature subset from the feature vector, we employ the feature selection method in [63]. The optimality criteria of minimal classification error usually requires the maximal statistical dependency of the target label on the data distribution in the subspace of the sub-features, which is often realized by maximal relevance – selecting the feature subset with the highest relevance to the target label. The method in [63] characterizes relevance by mutual information between a feature and the target label. However, as combinations of individually good features do not necessarily lead to overall good classification performance due to redundancy between features, it further uses a heuristic minimal-redundancy-maximal-relevance (mRMR) framework to select the characterizing feature subset. The mRMR algorithm is applied to the same ground truth dataset in section 3.6.1. We select feature subsets with feature number from 13 to 3, and use them to get cross-validation results as classifier performance measurement, which are shown from Figure 3.9 to Figure 3.12. From these figures we can see the consistent performance drop of the classifier when feature number is reduced. Even though the performance drop when 1 or 2 features being removed is not substantial, which could be an indication of some minor redundancy among the features, the full set of features still gets the best overall performance. Thus we keep the full set of 14 features in the SVM classifier.

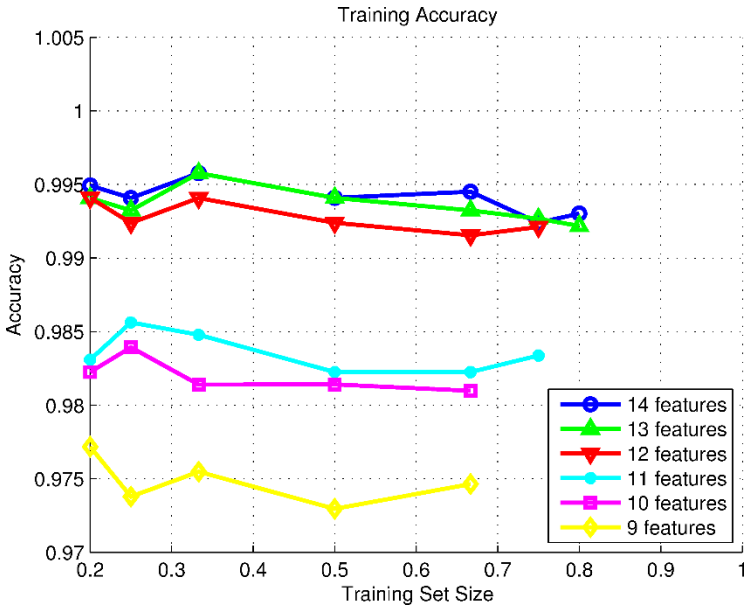


Figure 3.9 Cross-validation training accuracy for 14~9 features.[1]

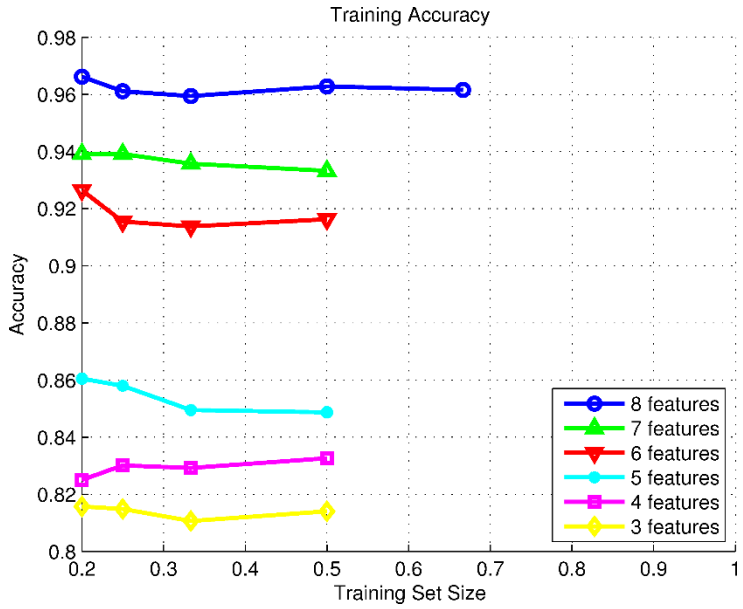


Figure 3.10 Cross-validation training accuracy for 8~3 features.

[1] The incomplete curves for 10 and 9 features are due to iteration numbers of SVM training exceeding the maximum allowed number, which indicates that the classifier has difficulty in finding a good separation surface for the training set because of reduced feature set. Same applies to Figure 3.10, Figure 3.11, and Figure 3.12.

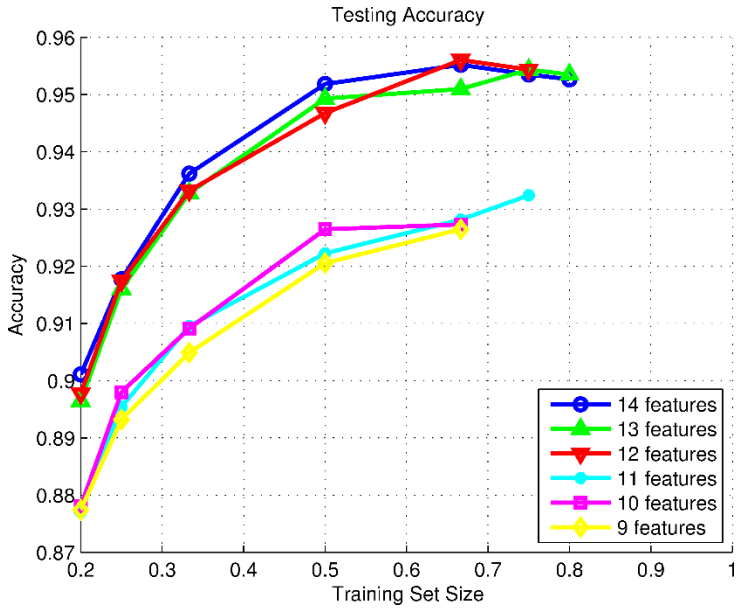


Figure 3.11 Cross-validation testing accuracy for 14~9 features.

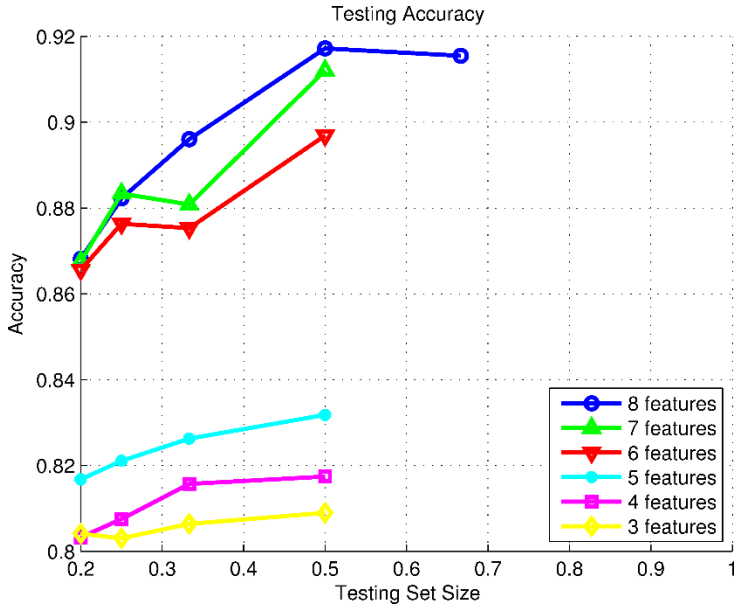


Figure 3.12 Cross-validation testing accuracy for 8~3 features.

In Table 3.2 we list the order of the feature vector in decreasing priority according to the mRMR feature selection algorithm, and we have some consistent observations from the order. At the low end are all the general statistical features, such as the mean data value, 95 and 68

percentile data value, and the mean value of autocorrelation function. This indicates that in average the power line data and the noise line data roughly fall in the same range in terms of the general statistics, as the noise lines often arise from some strong data points due to the ground return, and the average statistics of them are similar to that of power lines. Interestingly, at the other end, the single most effective feature is the maximum data value, which means that whenever we meet with some significantly large data value, it is very likely that this line is a power line, since the ground cannot generate as strong reflection as the metal power line. The other features in the high end are the specifically designed features to capture the Bragg periodic pattern, and we see that the 95 percentile is more effective than the 68 percentile as sharper peaks obtained by thresholding with the 95 percentile, which correlate better to the peaks of the true power lines.

Table 3.2 Priority order of the feature vector

Priority	Feature	Meaning of Feature
1	$\boldsymbol{x}(2)$	Maximum data value
2	$\boldsymbol{x}(12)$	Mid-high frequency component ratio
3	$\boldsymbol{x}(5)$	Number of peaks thresholded by 95%
4	$\boldsymbol{x}(7)$	Variance of peak distance thresholded by 95%
5	$\boldsymbol{x}(8)$	Number of peaks thresholded by 68%
6	$\boldsymbol{x}(13)$	Median value of autocorrelation function
7	$\boldsymbol{x}(10)$	Variance of peak distance thresholded by 68%
8	$\boldsymbol{x}(7)$	Mean value of peak distance thresholded by 95%
9	$\boldsymbol{x}(11)$	Low frequency component ratio
10	$\boldsymbol{x}(3)$	95 percentile of data value
11	$\boldsymbol{x}(9)$	Mean value of peak distance thresholded by 95%

12	$\boldsymbol{x}(4)$	68 percentile of data value
13	$\boldsymbol{x}(14)$	Mean value of autocorrelation function
14	$\boldsymbol{x}(1)$	Mean data value

3.6.3 Testing Results

The most direct yet effective way to examine the algorithm performance is to compare the frame-wise score, or probability, with the ground truth label, since the score is compared with a threshold of 0.5 to issue a warning signal to the pilot. In Figure 3.13 we show this comparison for dataset 2 and 3. The line with circles is the ground truth label, while the line with asterisks is the output of our algorithm. We can see that our score follows closely with the ground truth. In the power-line-absent frames, such as frame 25 – 45 of dataset 2, the computed scores have some spikes, which is due to false alarm line classifications. However, because of the adaptive frame probability computation algorithm, these false alarm lines do not lead to false alarm frames.

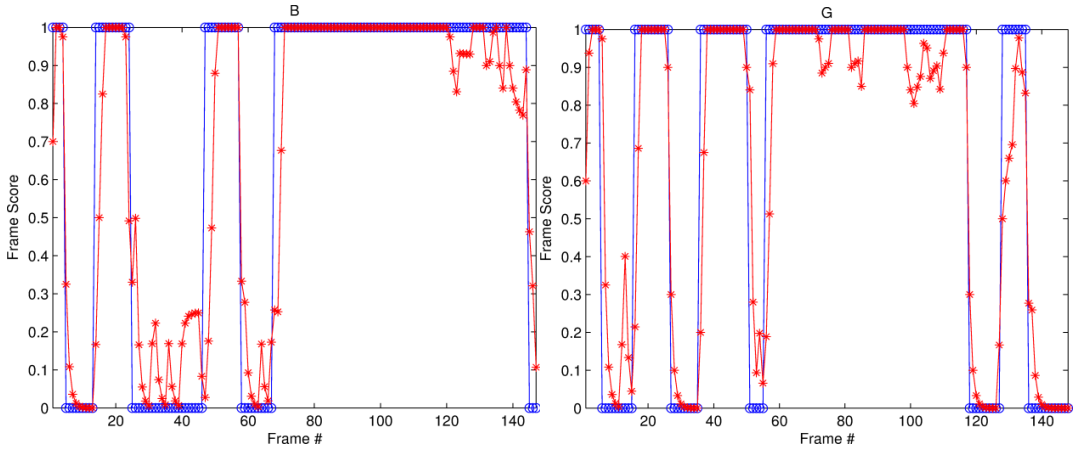


Figure 3.13 Frame-wise score results for datasets 2 and 3. Left: dataset2; right: dataset 3.

To further validate the effectiveness of the adaptive algorithm, we also include the results for the same two datasets without using the adaptive algorithm (while the frame probability is computed from the detection results in an individual frame directly) in Figure 3.14. From this, two points are worth noticing: (1) the single frame score cannot accumulate to a reasonable high value because lack of temporal correlation, as our frame-wise score computation algorithm depends on score accumulation through temporal correlation; (2) there are many false alarm frames. While (1) can be improved by adjusting algorithm parameters, (2) is inherent and cannot be easily overcome when the adaptive algorithm is disabled. With the adaptive algorithm in Figure 3.6 employed, a robust frame-wise detection result can be achieved.

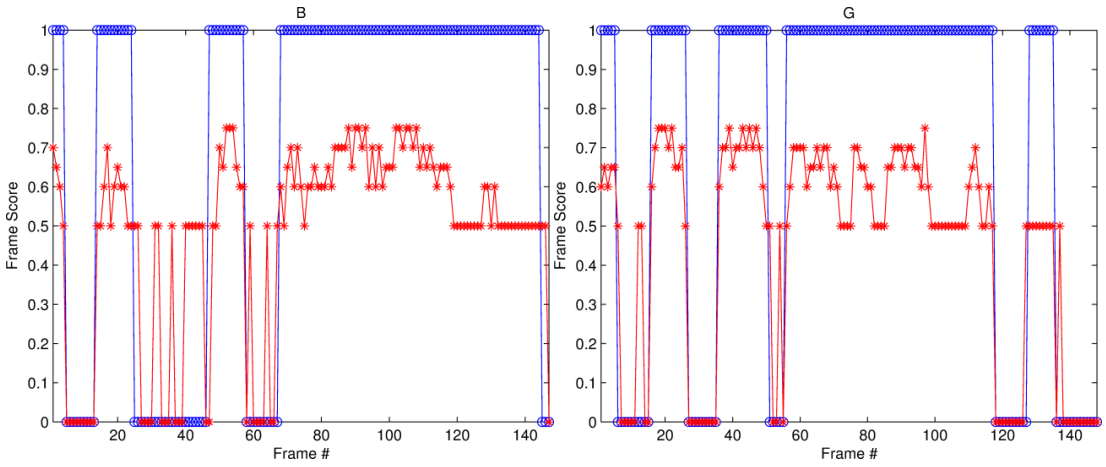


Figure 3.14 Frame-wise score results for datasets 2 and 3, with the adaptive algorithm disabled. Left: dataset 2; right: dataset 3.

We summarize the result comparisons for all the datasets in Table 3.3. We list the frame-wise detection rate, the false alarm rate, and the missing rate. The ground truth labels have sharp transitions from 0 to 1 and vice versa, but the frames in the transition stage from containing power lines to not containing power lines or vice versa are quite ambiguous, since the power

lines gradually “fade out” or “come into” the radar video. Thus, we exclude 2 frames on each side of the transition stage when calculating the rates in the table. Note that without the adaptive frame probability computing algorithm, there are a lot of false alarms. With the adaptive algorithm, the final frame-wise detection rate is raised from 96% to almost 100%.

Table 3.3 Performance Comparison With and Without the Adaptive Algorithm

Algorithm	Detection rate	False alarm rate	Missing rate
Full	99.84%	0.00%	0.16%
w/o adaptive algorithm	96.08%	3.92%	0.0%

To show that single Hough transform without the SVM will not work well, we list the result comparison in Table 3.4. Simple Hough transform basically fails, since it has a false alarm rate of more than 15%. This shows the effectiveness of the Hough transform line detection plus SVM classifier for the power line detection problem.

Table 3.4 Algorithm Performance Comparison With and Without the SVM

Algorithm	Detection rate	False alarm rate	Missing rate
Full	99.84%	0.00%	0.16%
w/o SVM	83.52%	16.48%	0.0%

In order to validate the effectiveness of the overlapped slice processing, in Figure 3.15 we show the results for the same two datasets without slice processing, i.e., the entire frame is thresholded once and the Hough transform is performed for it once. For both datasets the results

with slice processing is superior than that without, especially for dataset 2, where there are many frames containing power lines but with low probability using the algorithm without slice processing. To see how the performance varies with the slice size, in Table 3.5 we list the performance comparison with different slice heights. For each group the overlapping size is a quarter of the slice height. We can see that as the slice size gets smaller, the performance generally gets improved. However, too small slice size might prevent lines away from horizontal direction to be correctly detected, so we still use a slice height that is $1/16$ of the frame height.

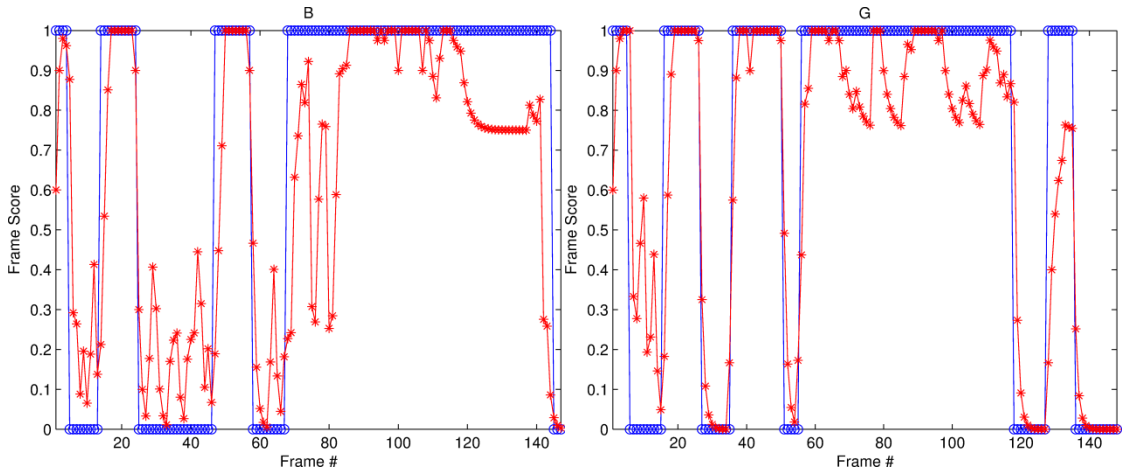


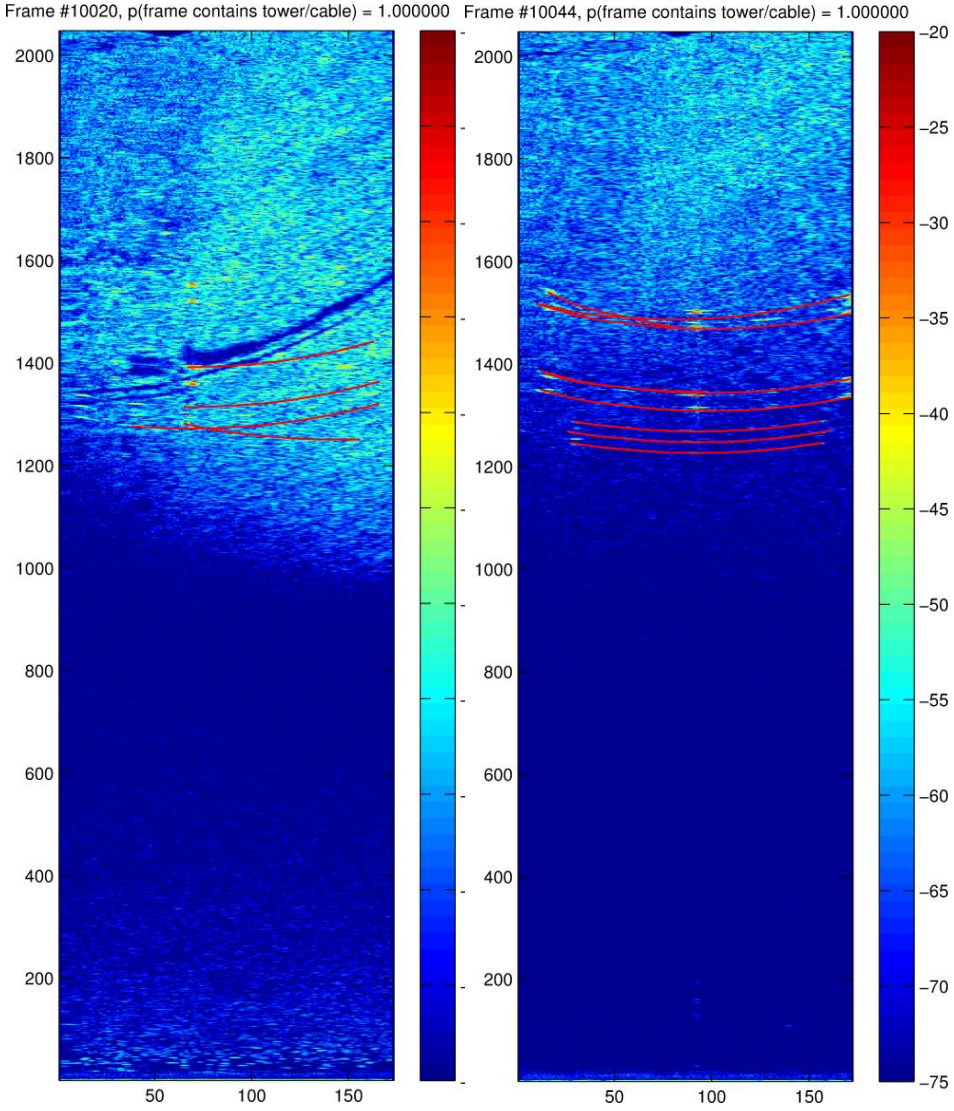
Figure 3.15 Frame-wise score results for datasets 2 and 3, without slice-processing. Left: dataset 2; right: dataset 3.

Table 3.5 Algorithm Performance Comparison With Respect to Slice Size

Slice height (w.r.t. frame height)	Detection rate	False alarm rate	Missing rate
1	98.12%	1.41%	0.47%
1/2	98.90%	0.78%	0.31%
1/4	99.22%	0.63%	0.16%

1/8	98.59%	1.41%	0.00%
1/16	99.84%	0.00%	0.16%
1/32	100.0%	0.00%	0.00%

In Figure 3.16 we show some examples of frame detection results, with the detected power lines overlaying on top of the radar images. We can see that the algorithm is capable of detecting most of the power lines while having few instances of false alarms.



(a) (b)

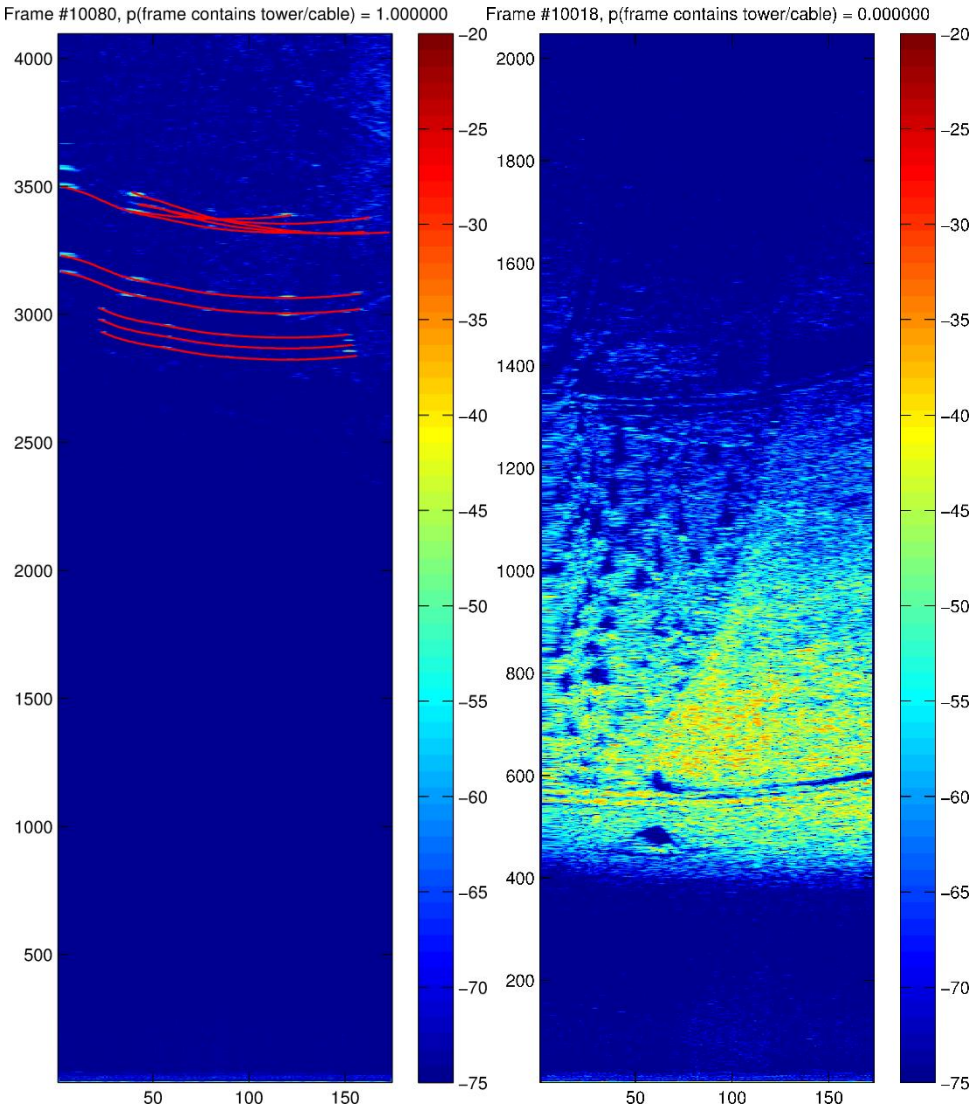


Figure 3.16 Some example frames with power line detection results. The last one does not contain any power line.

Lastly, the power-line detection algorithm is designed to run in real-time. We tested the algorithm on a PC with a 2.50 GHz dual-core CPU and 2GB memory, in a MATLAB and C hybrid implementation. In Table 3.6, we list the simulation results in terms of the average

processing time per frame for some of the datasets. We see that even in the un-optimized single-thread prototype implementation, the algorithm can run in real-time or near real-time. With code optimization and parallel implementation, the processing speed will be even faster.

Table 3.6 Average Processing Time Per Frame

Dataset	Frame size	# Frames	Time (s/frame)
1	2048×174	49	0.104
2	2048×174	147	0.106
3	2048×174	148	0.085
4	2048×176	147	0.083

3.7 Summary

In this chapter we present an algorithm for automatically detecting power line obstacles from the active millimeter-wave radar video. The algorithm is built upon several proposed new techniques such as pre-processing to reduce the noise while preserve the characteristics of the power lines in the radar video, Hough transform plus Support Vector Machine for automatic power line detection, a suitable set of features to represent the Bragg pattern of power lines so that a classifier can differentiate the power lines from the noise lines based on the feature set, overlapped slice-processing algorithm for robust performance in noisy situation and supporting parallel processing, and an adaptive frame probability computing algorithm utilizing the results from the classifier and the temporal correlation and parallel power line properties for reliable warning signal generation. Our testing results validate the performance of the algorithm; the frame-wise detection accuracy (i.e., correctly identify if a frame contains power lines) is higher

than 99%. The whole system, including radar data processing, display, and power line detection, can run in real time at the radar operating frame rate of 10 Hz.

Chapter 4 Robust Power Line Detection with Particle Filter Tracking

In the last chapter we proposed a power line detection algorithm based on Hough transform, Support Vector Machine, and an adaptive frame probability generating algorithm that takes care of other factors such as parallel power lines and temporal correlation. As stated in the introduction chapter, the idea is to identify the characteristics of the power lines, and use a specific tool to utilize one specific feature of the power lines. For the straight-line feature, Hough transform is applied; for the Bragg pattern, SVM is utilized. Yet we notice that for the important temporal correlation feature, the adaptive algorithm is rather heuristic. In this chapter, we propose to use a more formal tracking approach to better utilize the temporal correlation of the power lines. Specifically, we will use particle filter for tracking the power lines, for the reason that particle filter tracking does not have strong assumptions about the object dynamics nor noise distribution compared to the Kalman filter. As we are not sure about whether specific assumptions such as linear model and Gaussian noise hold true for the radar video, taking an approach that does not rely on these assumptions is better.

The organization of this chapter is as follows. In Section 4.1 we provide a more detailed discussion on particle filters for object tracking, followed by our proposed cascaded particle filter in Section 4.2. Then, in Section 4.3 we describe our observation models which link the power line detection problem with the framework of particle filter tracking. The overall power line detection with the particle filter tracking algorithm is presented in Section 4.4. Finally, we conclude this chapter with the simulation results in Section 4.5.

4.1 Object Tracking with Particle Filtering

The problem of object tracking can be more generally modeled as the estimation of the hidden state of a system that changes over time using a sequence of noisy measurements that are made on the system. The system state includes the information about the object that is of interest, such as the position, velocity, and size of the object. The measurement is carried out in the image frames of the video. Mathematically, consider the evolution of the state sequence $x_k,\ k \in \mathbb{N}$ of a target object given by

$$x_k = f_k(x_{k-1}, w_{k-1}) \tag{4.1}$$

where x_{k-1} is the state in the previous frame, f_k describes the system dynamics model which is a first-order Markov Chain, and $\{w_{k-1},\ k \in \mathbb{N}\}$ is an i.i.d. process noise sequence. The measurement sequence z_k is generated from the state sequence x_k by the measurement process

$$z_k = h_k(x_k, v_k) \tag{4.2}$$

and $\{n_k, k \in \mathbb{N}\}$ is an i.i.d. measurement noise sequence.

With the model and symbols defined, the tracking problem is to estimate $p(x_k|z_{1:k})$, the posterior pdf of the state x_k given all the measurements $z_{1:k}$ up to frame k from the Bayesian perspective. $p(x_k|z_{1:k})$ may be obtained recursively in two steps: prediction and update. The prediction step is to obtain the prior pdf of x_k, $p(x_k|z_{1:k-1})$, from the posterior $p(x_{k-1}|z_{1:k-1})$ in the previous frame and the system model in Eq. (4-1) using the Chapman-Kolmogorov equation [56]

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{4.3}$$

In frame k, when a new measurement is available, the posterior pdf is obtained via Bayes' rule

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{\int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k} \quad \textbf{(4.4)}$$

where $p(z_k|x_k)$ is the likelihood of the new measurement z_k given the predicted state x_k.

Particle filter is a sequential importance sampling (SIS) technique to approximate the posterior pdf $p(x_k|z_{1:k})$ using a finite set of N weighted samples $\{x_k^i, w_k^i\}_{i=1,\dots,N}$ by Monte Carlo simulation. When the number of samples N is large enough, the approximated posterior pdf becomes close to the true probability density and the approximate solution approaches the optimal Bayesian solution. Then candidate particles $\tilde{x}_k^i$ are sampled from an appropriate importance distribution $q(x_k|x_{1:k-1}, z_{1:k})$, and the weights of the samples are given by [64]:

$$w_k^i = w_{k-1}^i \frac{p\left(z_k|\tilde{x}_k^i\right)p(\tilde{x}_k^i|x_{k-1}^i)}{q(x_k|x_{1:k-1},z_{1:k})} \quad \textbf{(4.5)}$$

In the case of bootstrap filter [65], the importance distribution $q(x_k|x_{1:k-1}, z_{1:k})$ is the same as the state transition density $p(x_k|x_{k-1})$, and the weight w_k^i for each particle i in frame k is then simplified as

$$w_k^i = w_{k-1}^i \cdot p\left(z_k|\tilde{x}_k^i\right) \quad \textbf{(4.6)}$$

Because a large number of these particles will have negligible weights, the particles are re-sampled in each frame to avoid the degeneracy problem. For a fixed number of particles, $w_{k-1}^i = \frac{1}{N}$ is a constant and can be ignored. In the end, the importance weight in Eq. 4-6 is reduced to $p\left(z_k|\tilde{x}_k^i\right)$, the conditional likelihood of a new observation z_k given the particle $\tilde{x}_k^i$.

4.2 Cascaded Particle Filters

The reason for the success of particle filter is two-fold. First, the theoretic framework of Bayesian estimation is a general and well-established model. The sequential Bayesian estimation model in Eq. 4-3 and Eq. 4-4 captures the nature of object tracking. Secondly, even though Eq. 4-3 and Eq. 4-4 are usually intractable except for a few special cases such as the linear dynamic model with Gaussian noise, Monte Carlo simulation can deal with any general distribution in a non-parametric way as long as the number of samples N is large enough.

However, when the dimensionality of the state space increases, the number of samples needed to effectively represent the probability density also increases, in an exponential rate – well known as the "curse of dimensionality". Although the computational cost for evaluating the likelihood function $p(z_k|\tilde{x}_k^i)$ for one sample is low, when the number of samples increases exponentially, the cost becomes significant. One could sacrifice the number of samples N for speed, but this will cause incomplete and problematic representation and estimation of the true probability density.

We thus propose to use cascaded particle filters to alleviate the curse of dimensionality. We observe that when the state vector incorporates more information and the dimensionality of the state space increases, the state vector can often be decomposed into a few un-correlated sub-states, and the state space can be decomposed into a few orthogonal sub-spaces. For example, when a soccer player is tracked and the state vector consists of the location and the speed of the player, usually the location is independent of the speed, or at least the dependence is weak. Let $x_k = (u_k, v_k)$, if we have the independence conditions for u_k and v_k satisfied such that $p(x_k) =$

$p(u_k, v_k) = p(u_k) \cdot p(v_k)$, $p(x_k, x_{k-1}) = p(u_k, u_{k-1}) \cdot p(v_k, v_{k-1})$, and $p(x_k, z_{1:k}) = p(u_k, z_{1:k}) \cdot p(v_k, z_{1:k})$, it can be easily shown that

$$\begin{array}{ccc} p(x_k|x_{k-1}) & = & p(u_k|u_{k-1}) \cdot p(v_k|v_{k-1}) \\ p(x_{k-1}|z_{1:k-1}) & = & p(u_{k-1}|z_{1:k-1}) \cdot p(v_{k-1}|z_{1:k-1}) \\ p(z_k|x_k) & = & p(z_k|u_k) \cdot p(z_k|v_k) \end{array} \quad \textbf{(4.7)}$$

So the prediction step in Eq. (4.3) becomes

$$p(x_k|z_{1:k-1}) = p(u_k|z_{1:k-1}) \cdot p(v_k|z_{1:k-1}) \quad \textbf{(4.8)}$$

And the update step in Eq. (4.4) becomes

$$p(x_k|z_{1:k}) \propto p(u_k|z_{1:k}) \cdot p(v_k|z_{1:k}) \quad \textbf{(4.9)}$$

It is clear from Eq. (4.8) and (4.9) that both the prediction and the update steps can be factored into the prediction and update of u_k and v_k separately, so the estimation of u_k and v_k are independent of each other. Thus, the original tracking problem in a high-dimensional state space can be solved by some cascaded tracking in low-dimensional sub-spaces, given that the state vector can be decomposed into some independent sub-space state vectors. The reduced dimensionality simplifies the problem, requires fewer samples to represent and estimate the probability density, and has a higher chance of success. The dynamic model to propagate the particles can be defined separately in the sub-spaces, and the measurement likelihood $p(z_k|\tilde{x}_k^i)$ is also decomposed into individual measurement likelihood in the sub-spaces, i.e., $p(z_k|\tilde{u}_k^i)$ and $p(z_k|\tilde{v}_k^i)$. Similar idea of factorization has been successfully applied to face detection in [66].

4.3 Observation Models

In this section we define the observation models that embed the previous power line detection algorithm into the particle filter tracking framework. The power line is represented by two parameters, θ and ρ in the Hough transform domain, and they are very much independent, i.e., the distance between the radar sensor and the power line is independent from the approaching angle of the helicopter. Another reason for separating θ and ρ is because in reality we find that all the power lines in the field-of-view captured by the radar are parallel, thus the θ value for all the power lines are the same. θ can be estimated first, then individual ρ value for individual power line can be further estimated by individual ρ tracker along the estimated θ direction. Thus, according to the cascaded particle filters algorithm developed in the previous section, we consider two separate likelihood measurement functions, $p(z_k|\tilde{\theta}_k^i)$ and $p(z_k|\tilde{\rho}_k^i)$.

4.3.1 Observation Model for θ

The purpose of θ tracking is to estimate the orientation of all the power lines in each frame. To compute the conditional likelihood of a particular θ sample $\tilde{\theta}_k^i$, we combine different sources of information, namely, a concentration measure based on the Hough transform data, the strength of lines, and temporal smoothness:

$$p(z_k|\tilde{\theta}_k^i) = \underbrace{c(z_k|\tilde{\theta}_k^i)}_{\text{concentration}} \cdot \underbrace{s(z_k|\tilde{\theta}_k^i)}_{\text{line strength}} \cdot \underbrace{g_\theta(\tilde{\theta}_k^i, \hat{\theta}_{k-1})}_{\text{temporal smoothness}} \quad \textbf{(4.10)}$$

After the preprocessing algorithm consisting of thresholding and coordinate transformation, as described in the previous chapter, Hough transform converts an entire frame z_k to Hough transform domain data $H_k(\theta, \rho)$, where $H_k(\theta_i, \rho_j)$ represents the number of pixels (or line

strength) for a particular straight line parameter combination (θ_i, ρ_j). From the definition of Hough transform, it can be shown that the sum of all the Hough transform domain data for any particular θ is constant, i.e., $\sum_\rho H_k(\theta_1, \rho) = \sum_\rho H_k(\theta_2, \rho)$, yet $H_k(\theta_1, \rho)$ and $H_k(\theta_2, \rho)$ have different distributions over ρ. For the true power line orientation, $H_k(\theta_{\text{true}}, \rho)$ is more concentrated because a few power lines with large number of pixels will dominate, as shown in

Figure 4.1. Taking the idea from information theory that the more concentrated a distribution is, the lower its uncertainty and its entropy, we define the "concentration" measure similar to the entropy:

$$c\left(z_k \middle| \tilde{\theta}_k^i\right) = \sum_\rho H_k\left(\tilde{\theta}_k^i, \rho\right) \log H_k\left(\tilde{\theta}_k^i, \rho\right)\Big|_{H_k\left(\tilde{\theta}_k^i, \rho\right) > 0} \quad \textbf{(4.11)}$$

For the true power line orientation, there will be a few lines with significant strength (the power lines), and having significant number of pixels on these lines. The line strength measure $s(z_k|\tilde{\theta}_k^i)$ takes the sum of the top T values (in our simulation we use $T = 3$) in $H_k\left(\tilde{\theta}_k^i, \rho\right)$. Lastly, the temporal smoothing term for θ is defined as:

$$g_\theta\left(\tilde{\theta}_k^i, \hat{\theta}_{k-1}\right) = \exp\left(-\left(\tilde{\theta}_k^i - \hat{\theta}_{k-1}\right)^2 / 2\sigma_\theta^2\right) \quad \textbf{(4.12)}$$

where $\hat{\theta}_{k-1}$ is the tracked θ in the previous frame and σ_θ is the standard deviation parameter of the Gaussian function.

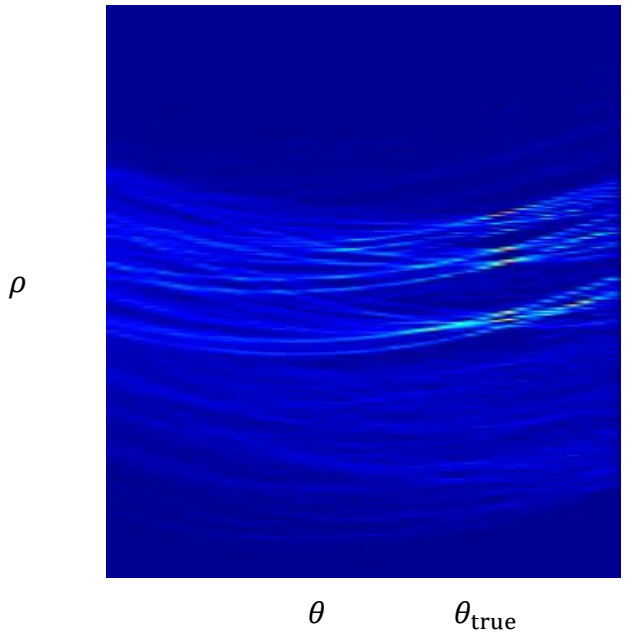


Figure 4.1 Hough transform (part) of a frame. The image is displayed in pseudo-color.

4.3.2 Observation Model for ρ

The ρ tracker is cascaded after the θ tracker and tracks for the ρ value of each individual power line along the orientation $\hat{\theta}_k$ tracked by the θ tracker. Similarly, different sources of information are combined to define the conditional likelihood of a particular ρ sample $\tilde{\rho}_k^i$:

$$p\left(z_k\middle|\tilde{\rho}_k^i\right) = \underbrace{f\left(z_k\middle|\tilde{\rho}_k^i\right)}_{\substack{\text{classifier}\\ \text{confidence}}} \cdot \underbrace{a\left(z_k\middle|\tilde{\rho}_k^i\right)}_{\substack{\text{association}\\ \text{function}}} \cdot \underbrace{g_\rho(\tilde{\rho}_k^i, \hat{\rho}_{k-1})}_{\text{temporal smoothness}} \quad \textbf{(4.13)}$$

The classifier confidence is directly inherited from the SVM as the SVM decision function value for a particular candidate line specified by $(\hat{\theta}_k, \tilde{\rho}_k^i)$. Here we see the previous power line detection algorithm can fit nicely into the tracking framework via $f\left(z_k\middle|\tilde{\rho}_k^i\right)$. The association function measures the similarity of the Hough transform domain data in a local neighborhood

between this sample $\tilde{\rho}_k^i$ and the tracked $\hat{\rho}_{k-1}$ in the previous frame, based on the intuition that for the same power line, the Hough transform domain data should be similar in local neighborhood for neighboring frames. It is defined as the normalized correlation:

$$a(z_k|\tilde{\rho}_k^i) = \frac{\sum_{l=-r}^{r} H_k(\hat{\theta}_k,\tilde{\rho}_k^i+l) H_{k-1}(\hat{\theta}_{k-1},\hat{\rho}_{k-1}+l)}{\sqrt{\sum_{l=-r}^{r} H_k(\hat{\theta}_k,\tilde{\rho}_k^i+l)^2} \cdot \sqrt{\sum_{l=-r}^{r} H_{k-1}(\hat{\theta}_{k-1},\hat{\rho}_{k-1}+l)^2}} \quad \textbf{(4.14)}$$

where r is a parameter specifying the size of the local neighborhood. Lastly, the temporal smoothing term for ρ is defined in the similar way as Eq. 4-12:

$$g_\rho(\tilde{\rho}_k^i, \hat{\rho}_{k-1}) = \exp(-(\tilde{\rho}_k^i - \hat{\rho}_{k-1})^2 / 2\sigma_\rho^2) \quad \textbf{(4.15)}$$

4.4 The Overall Power Line Detection with Particle Filter Tracking Algorithm

To complete the particle filter tracking algorithm, we need to define the motion dynamic models that propagate the particles. Considering that the movement of the helicopter is smooth, we use a simple drifting model for both θ and ρ:

$$\theta_k = \theta_{k-1} + \varepsilon_\theta \quad \textbf{(4.16)}$$

$$\rho_k = \rho_{k-1} + \varepsilon_\rho \quad \textbf{(4.17)}$$

The process noise ε_θ and ε_ρ are drawn from zero-mean Gaussian distributions with standard deviations of σ_θ and σ_ρ, respectively.

With all the building blocks explained, now we present the complete power-line detection with tracking algorithm. For readability, we first present the θ-tracking algorithm, the purpose of which is to estimate the orientation of all the parallel power lines in a frame given the orientation of the power lines in the previous frame:

Algorithm 1 The θ-tracking algorithm
Input: A new radar video frame z_k, and its Hough Transform $H_k(\theta, \rho)$

If $k = 0$, i.e., the first frame **then**

- $\hat{\theta}_0 = \underset{\theta}{\operatorname{argmax}}\, c(z_0|\theta) \cdot s(z_0|\theta)$
- **Initialize** N_θ θ-particles $\tilde{\theta}_0^i = \hat{\theta}_0$, $w_{\theta_0}^i = \frac{1}{N_\theta}$

Else

- **Propagate** θ-particles according to Eq. 4-16
- **Measure** weight according to Eq. 4-10, $w_{\theta_k}^i = p(z_k|\tilde{\theta}_k^i)$
- **Output** $\hat{\theta}_k = \underset{i}{\operatorname{argmax}}\, w_{\theta_k}^i$
- **Re-sample** N_θ un-weighted particles from $p(\theta_k|z_{1:k})$ approximated by $\{\tilde{\theta}_k^i, w_{\theta_k}^i\}$

End if

Output: $\hat{\theta}_k$

The re-sampling step is implemented in the same way as [55]. The algorithm for processing a ρ-tracker is presented in the following Algorithm 2.

Algorithm 2 The ρ-tracker processing algorithm
Input: A new radar video frame z_k, its Hough transform $H_k(\theta, \rho)$, the output $\hat{\theta}_k$ from the θ-tracker, and the particles of the ρ-tracker $\{\tilde{\rho}_{k-1}^i, w_{\rho_{k-1}}^i\}$ from the previous frame

Propagate ρ-particles according to Eq. 4-17
Measure weight according to Eq. 4-13, $w_{\rho_k}^i = p(z_k|\tilde{\rho}_k^i)$
Output $(\hat{\rho}_k, w_{\hat{\rho}_k}) = \underset{i}{\operatorname{argmax}}\, w_{\rho_k}^i$
Re-sample N_ρ un-weighted particles from $p(\rho_k|z_{1:k})$ approximated by $\{\tilde{\rho}_k^i, w_{\rho_k}^i\}$

Output: $\hat{\rho}_k, w_{\hat{\rho}_k}$

And the complete power line detection with tracking is presented in Algorithm 3, and also depicted in Figure 4.2. In this algorithm, T_ρ is a parameter that controls the association threshold for the ρ-tracker, and M_ρ defines the maximum number of ρ-trackers allowed in each frame. In the first frame the ρ-trackers are initialized by searching for local maxima in the Hough transform data, which is the same way for detecting power lines in the previous algorithm. If a line candidate (corresponding to a local maximum in the Hough transform data) is classified by the SVM as a power line, a ρ-tracker is initialized and it continues to track the positions of this power line in future frames; otherwise it is ignored and no ρ-tracker is initialized. In the case of a false alarm power line, the tracker will most likely not be able to find any good association in the next frame and this false alarm ρ-tracker will be terminated. We also deal with the situation of power line occlusion by ground return noise. When a power line is occluded by noise, the tracker could lose track of it. To re-capture it when the power line appears again, in each fame we also search for candidate power lines in the region that is not covered by any ρ-tracker and initialize new ρ-trackers. We immediately terminate the "lost-track" trackers rather than keeping them running and predicting because the purpose is to detect the power lines rather than to have an exact track of each single power line, and in simulation, we find that such a strategy of immediate termination and re-initialization is more effective for detecting the power lines.

Algorithm 3 The overall power line detection with tracking algorithm
Input: A new radar video frame z_k

Step 1. Pre-processing: thresholding and coordinate transformation
Step 2. Hough transform: $z_k \rightarrow H_k(\theta, \rho)$
Step 3. θ-tracking: get $\hat{\theta}_k$ by Algorithm 1
Step 4: ρ-tracking
If $k = 0$, i.e., the first frame **then**
    Initialize M_ρ ρ-trackers by searching for M_ρ local maxima $\{\rho_{0,j}, j = 1, \dots, M_\rho\}$ in $H_0(\hat{\theta}_0, \rho)$
    For all $\rho_{0,j}$ **do**
        If $f\left(z_k \middle| \rho_{0,j}\right) > 0$, i.e., passes SVM **then**
            initialize N_ρ particles $\tilde{\rho}_{0,j}^i = \rho_{0,j}$
        Else
            Terminate this ρ-tracker $\rho_{0,j}$
        End if
    End for
Else
    Process each ρ-tracker $\rho_{k,j}$ by Algorithm 2, get the ρ-tracker output $\hat{\rho}_{k,j}, w_{\hat{\rho}_{k,j}}$
    If $w_{\hat{\rho}_{k,j}} > T_\rho$ **then**
        Keep this ρ-tracker
    Else
        Terminate this ρ-tracker
    End if
    Add new ρ-trackers by searching for local maxima not covered by any ρ-tracker, similar to the $k = 0$ initialization case, and allow up to M_ρ ρ-trackers
End if

Output: $(\hat{\theta}_k, \hat{\rho}_{k,j})$ in each existing ρ-tracker as detected power lines in z_k

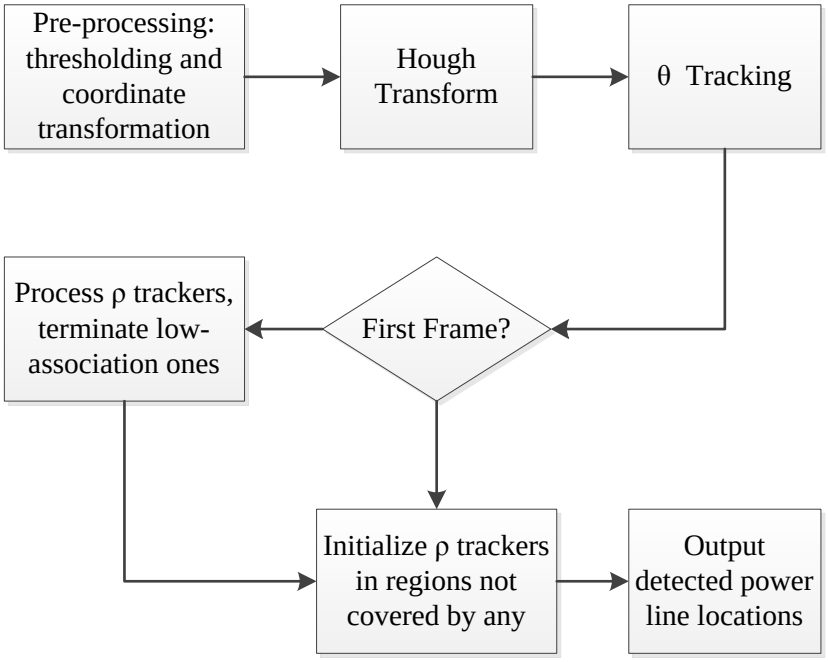


Figure 4.2 The flowchart of the overall power line detection by tracking algorithm.

4.5 Simulation Results

In Table 3.1 we listed the 6 datasets that we use to evaluate the previous power-line detection algorithm in the previous chapter. In addition to that, we collected two more datasets that represent the more difficult "pedal turn" cases as described in Table 4.1.

Table 4.1 Two Additional Datasets

Dataset #	Nature of data	# of frames	Frame size
7	Pedal turn at 150m	118	4096×343
8	Advanced pedal turn at 400m	119	4096×174

To show the effectiveness of the new power line detection algorithm, we compare the line-level detection results in Table 4.2.

Table 4.2 Power-line-level Recall and Precision Comparison With the Previous Algorithm

Dataset #	Previous		New	
	Recall	Precision	Recall	Precision
1	75.87%	58.13%	97.73%	91.51%
2	57.29%	77.59%	79.57%	94.29%
3	77.72%	92.55%	80.69%	91.01%
4	50.26%	63.23%	89.65%	84.81%
5	79.06%	76.68%	97.91%	97.84%
6	73.78%	81.17%	87.45%	99.85%
7	57.37%	50.13%	94.78%	91.75%
8	46.15%	45.08%	100.0%	90.64%
Overall	**68.36%**	**68.94%**	**92.03%**	**92.83%**

We manually inspect the result for each frame, and compute the line-level recall and precision for each dataset. The line-level results is the performance of detecting each individual power line instead of just detecting if a frame contains power lines or not as what the frame-level result indicates. We can see that with the cascaded particle filter tracking algorithm, both the line-level recall and the precision are significantly improved, thus boosting the robustness of the power line detection algorithm. The frame-level result is for the power line detection with tracking algorithm is as good as the previous algorithm.

To validate the necessity for cascaded θ-tracking and ρ-tracking, in Table 4.3, we compare the power-line-level recall and precision with θ-only tracking algorithm.

Table 4.3 Power-line-level Recall and Precision Comparison With θ-only Tracking

Dataset #	θ-only		$\theta + \rho$	
	Recall	Precision	Recall	Precision
1	84.64%	91.59%	97.73%	91.51%
2	61.88%	92.70%	79.57%	94.29%
3	77.92%	85.90%	80.69%	91.01%
4	64.68%	75.31%	89.65%	84.81%
5	92.56%	100.0%	97.91%	97.84%
6	84.88%	99.83%	87.45%	99.85%
7	90.69%	88.31%	94.78%	91.75%
8	93.86%	94.31%	100.0%	90.64%
Overall	**79.86%**	**90.47%**	**92.03%**	**92.83%**

For θ-only tracking, Algorithm 1 is activated, but not Algorithm 2. In each frame, the overall power line orientation is tracked, and the top M_ρ lines along that direction are classified by the SVM classifier as power lines or noise lines. We can see the performance with full $\theta + \rho$ tracking algorithm is superior to that of θ-only. Note that the involvement of ρ-trackers particularly improves the recall, which means more true power lines can be detected. The reason is that without the ρ-trackers, power lines that are half-occluded by the ground return noise may not be correctly classified by the SVM, thus they are missed by the θ-only algorithm. But with the ρ-tracking algorithm, the strong association of these partially occluded power lines between neighboring frames can still be greater than T_ρ, thus the effective utilization of temporal

correlation complements the "blind spots" of the SVM classifier. On the other hand, when the power lines are tracked through regular particle filter tracking algorithm in the original (θ, ρ) state space instead of the cascaded particle filter tracking, in simulation, we find that the performance is not as good. The main reason is that the data on a line off the real orientation but crossing multiple power lines looks just like a true power line, having multiple peaks corresponding to its crosses with the true power lines, thus bringing confusions. However, with the cascaded particle filter, since the overall true orientation of the power lines can often be correctly estimated, the confusions will be avoided.

We show the visual results comparison in Figure 4.3, where we list the original frames, the ground truth, the detection results with the previous heuristic adaptive algorithm, and the results with the new algorithm based on cascaded particle filter tracking. It should be noted that these are all zoom-in views showing the power line regions, not the entire field-of-view of the radar. The power lines are overlaid as red lines in the detection results. We can clearly see the superiority of the detection results with particle filter tracking algorithm. Even when the ground return noise is strong and the power lines are occluded, the detection with tracking algorithm can still correctly detect most of them. The previous heuristic adaptive algorithm suffers from some false alarm lines, while the new algorithm based on particle filter tracking has a much cleaner result.

In Table 4.4 we also present the processing speed performance comparison of the new algorithm based on particle filter tracking with the previous heuristic adaptive algorithm. The implementations of both the algorithms are un-optimized single-thread prototype implementations in C. The test is performed on a desktop PC with a 3.40 GHz CPU. We can see

that the speed of the new algorithm based on particle filter tracking is at about the same level as the previous heuristic adaptive detection algorithm, while for some datasets the new algorithm with particle filter tracking actually performs even better. For a radar video with a frame size of 2048 by 175 pixels, the algorithm can run in real-time of about 10 frames/second. For a 4096 line video, the algorithm can run roughly in half real-time.

Table 4.4 Speed Performance Comparison in Terms of Average Processing Time Per Frame

Dataset #	Size	Frames	Time per frame (s)	
			Previous	New
1	2048×174	49	0.079	0.093
2	2048×174	147	0.071	0.076
3	2048×174	148	0.068	0.065
4	2048×176	147	0.065	0.073
5	4096×175	97	0.198	0.181
6	4096×174	149	0.212	0.188
7	4096×343	118	0.304	0.276
8	4096×174	119	0.204	0.184

The implementation of the tracking algorithm requires few user-specified parameters. $\varepsilon_\theta, \varepsilon_\rho$ and T_ρ are inherent to the nature of power line object dynamics and association, which can be optimally estimated from some training data. The training requires manually labeling the track of each individual power line which is rather tedious. M_ρ specifies the maximum number of power lines in each frame, and we set it to 8 since in the datasets we have, that is the maximum number

of power lines to appear in one frame. For the number of particles of the trackers, we set $N_\theta = 80$, and $N_\rho = 20$.

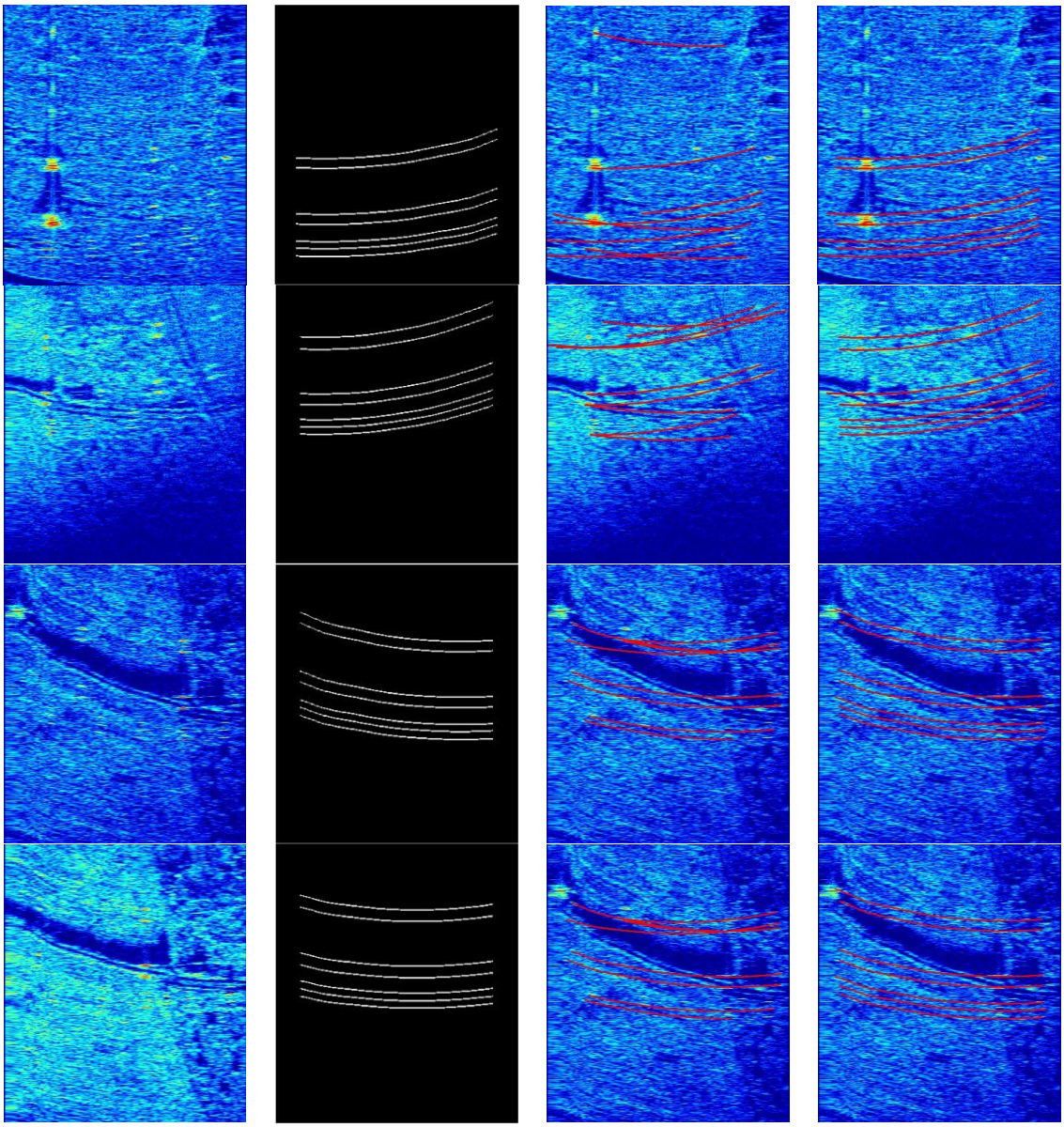


Figure 4.3 Some example frames with power line detection results comparison. From left to right: original radar images, ground truth, results of the previous algorithm, and results of the new algorithm.

4.6 Summary

In this chapter we propose an improved power line detection algorithm for the millimeter-wave radar video based on Hough transform, Support Vector Machine, and particle filter tracking. Instead of using the heuristic adaptive frame-probability approach that we use in the previous algorithm described in the previous chapter, the particle filter tracking can better utilize the temporal correlation property of the power lines. The particle filter framework naturally captures the temporal correlation of the power line objects, and the power-line-specific features are embedded into the conditional likelihood measurement process of the particle filter. Because of the fusion of multiple information sources, the detection of the power line is more effective and robust than the previous approach. We also propose a framework of cascaded particle filters that takes advantage of the independence property of different aspects of the object state, which is especially suitable for the power line tracking problem given the fact that in the radar field-of-view all the visible power lines are parallel and the two parameters in the Hough transform domain are relatively independent. The simulation result validates the significant performance improvement of the new algorithm based on particle filter tracking over the previous heuristic adaptive algorithm. It can also achieve real-time performance of about 10 fps using a desktop PC with a 3.40 GHz CPU.

Chapter 5 Conclusion and Future Works

In this chapter, we present the concluding remarks and summarize the main contributions of the dissertation. We also identify future research directions along the line of works we have accomplished so far.

5.1 Conclusion

In this dissertation, we focus on developing automatic power line detection algorithms for the 94-GHz active millimeter-wave radar video. The main goal of the automatic power line detection is to avoid power-line-strike accidents of the helicopters and improve their flight safety. With the automatic power line detection algorithms, the necessity for the pilot to manually inspect the radar video can be avoided. Furthermore, in some difficult cases when the power lines are difficult to be recognized, the automatic power line detection algorithms can provide additional assistance. In order for the algorithms to be practical, they need to be able to be performed in real time (at least 10 frames per second) with detection accuracy higher than 99%. This puts a constraint on the complexity of the algorithms that can be used.

In Chapter 2, an overview of the background information for the power line detection algorithms is presented. We provide a review of the basic architecture of the power line imaging radar system, common imaging results, and the interpretation of the radar images. We identify the main characteristics of the power lines in the radar video as they are temporally correlated parallel straight lines with certain features known as the Bragg pattern, and the main challenge for detecting the power lines as differentiating the power lines from the noise lines due to the

strong noise from the ground return that can often occludes the power lines in the radar video. The main idea of the power line detection algorithm is to utilize the power-line-specific characteristics: a straight line detection algorithm to utilize the straight power line characteristic, a classification algorithm for utilizing the Bragg pattern to differentiate the true power lines and the noise lines, and a tracking algorithm for utilizing the temporal correlation. Thus, brief reviews of straight line detection algorithms, classification algorithms, and tracking algorithms are also provided in this chapter.

Chapter 3 presents a power line detection algorithm that consists of a pre-processing step, a straight line detection algorithm, a classifier, and a post-processing step. In the pre-processing step, we propose an overlapping slice processing method which facilitates parallel processing to achieve real-time performance while improving the detection accuracy. Based on the straight line characteristic of the power lines, the straight line detection algorithm first finds the straight lines in the radar images as power line candidates. We use the Hough transform to detect straight lines due to its simplicity, desired properties, and the good performance. A Support Vector Machine classifier is trained to further detect the true power lines from the line candidates. In the post-processing, we propose an adaptive algorithm which utilizes the temporal correlation and parallel-line properties of the power lines by adjusting a frame-wise indicating probability of containing power lines to achieve a robust and accurate detection, especially in the presence of heavy noise. Simulation results show that this power line detection algorithm can achieve excellent frame-level result (to indicate a frame contains power-lines or not) with a 99.84% correct detection rate and about 10 fps processing speed with a PC that has a 2.5 GHz dual core CPU and 2GB memory.

Although the algorithm presented in Chapter 3 achieves excellent frame-level result, the power-line-level result (to identify each individual power line in each frame) is not as good. Chapter 4 further improves the robustness of the power line detection algorithm by better utilizing the temporal correlation property of the power lines in the form of tracking instead of the heuristic adaptive algorithm presented in Chapter 3. Conventional particle filter tracking algorithm is adapted to better suit the power-line-specific characteristics, and we propose a cascaded particle filters tracking algorithm. Unlike the conventional particle filter which tracks the state of the object as a whole, the cascaded particle filters track the individual features of the object states separately, given the independence of the features. In our case, the orientation of the power lines and their distance from the helicopter is much more independent, thus we propose to track these two features independently. The cascaded particle filters tracking algorithm is suitable for power line tracking also because the parallel power lines have the same orientation, so the orientation can be first tracked, and then each individual power line can be tracked along the tracked direction. The structure of cascaded particle filters reduces the dimensionality of the tracking problem, and also allows the parallel power line properties to be naturally embedded into it. Because of the effective utilization of the unique power line parallel characteristic and the temporal correlation, the detection can be much more robust in the power line level compared to the algorithm presented in Chapter 3.

In summary, in this work we exploit the power-line features in the radar video, and the effective utilization of these features results in the automatic detection algorithms of the power lines. These algorithms are tested with real-world flight testing data, and they have shown good

performance under a variety of flight conditions. The main contributions of this thesis are summarized as follows:

1. To the best of our knowledge, we are the first to propose the use of formal image processing, machine learning, and object tracking techniques for automatic power line detection in active millimeter-wave radar video. Specifically, we propose a framework using a straight line detection algorithm, a classifier, and an object tracker to automatically detect the power lines in active millimeter-wave radar video.
2. We propose suitable features to characterize the Bragg pattern of power lines so that the classifier can utilize them to distinguish the true power lines and noise lines for a reliable classification.
3. We propose an overlapping slice processing method which facilitates parallel processing to achieve a real-time performance while improving the detection accuracy.
4. We propose an adaptive algorithm which utilizes the temporal correlation and parallel-line properties of the power lines by adjusting the probability of a frame containing power lines to achieve a more robust and accurate detection especially in the presence of heavy noise.
5. We propose an approach which integrates the power line detector into a particle filter tracking framework for more effective utilization of temporal correlation. It can achieve much more robust power line detection.
6. We propose a cascaded particle filters structure for dimensionality-reduced tracking. The cascaded particle filters also allow the parallel power line property to be naturally embedded into it. The result is higher robustness and lower computation cost.

5.2 Future Works

Given the unique radar imaging platform, we identify some directions of future works that go both within and beyond the scope of power line detection. We summarize these possible future works in this section.

- Further investigation of the feature vector

As shown in the cross-validation results of Section 3.6.1, the proposed 14-dimenisonal feature vector is capable of representing the Bragg pattern and differentiating the power line pattern from the noise line pattern through a SVM classifier. However, it is not yet clear whether it is possible that a feature vector with a subset of the features can achieve similar performance, or which features among the 14-dimensional feature vector are the main features that can separate the power lines from the noise lines. Feature selection not only can choose the essential features and save computation of classification but also can give us insights about how to classify the two groups from a machine's perspective, and help us design even better feature sets that can better separate the two groups and improve the classifier's performance.

- Detection of the towers

Towers often are also the visible objects in the radar video, as can be seen in Figure 2.5 and Figure 2.6, as the large blobs of pixels with strong intensities. It should be noted that in Figure 2.5 there are only two towers, while the other tower-like blobs are from the saturation signal of the IF channel, which is an artifact of the radar image synthesis system and has been corrected in later datasets. In Figure 4.3, there are also towers visible in the radar images. When towers are present, it is for sure that there are also power lines present, though the opposite is not necessarily true. The correct detection of the towers can give us more evidence of the presence of

power lines. However, how to reliably detect the towers is also a challenging problem due to its irregular shape and the strong noise return from the ground that often can form blobs of strong pixels. Another challenge is how to probabilistically combine the confidence from the tower detection and the power line detection for more robust decision making.

- Compression of the radar video

The uncompressed radar video is large in size. Even though on-fly processing and display is not affected by the large data amount, it constrains the total length of the video that can be recorded for research or archive purposes. Mature video compression technologies for general video are prevalent, but due to the special characteristics of the radar video, new compression techniques that are better suited for the radar video will be useful.

- On-flight testing

Currently the power-line detection algorithm is only tested with pre-recorded datasets. We have eight datasets that contain near a thousand frames which can well represent various flight conditions, yet on-flight testing can further confirm the effectiveness of the power line detection algorithms in the actual working situation, especially under challenging conditions such as rainy or foggy weather.

- De-noising of the radar video

The radar video often suffers from strong noise from the ground return and other noise sources, and the noise significantly degrades the visual quality of the radar video to the viewer, plus that the noise often occludes the power lines. In fact, the major challenge of the power line detection algorithm is how to take care of the noise. The tracking algorithm can provide robustness of detection in the face of noise. However, we would also like to explicitly deal with

the noise problem and enhance the radar video. The signal characteristic of the ground return noise is very different from that of a meaningful object – it is quite random in both the spatial domain and the temporal domain. The enhancement of the radar video could possibly further improve the performance of the power line detection algorithms. Markov Random Field (MRF) and its descendants have been used to model image and video, and various de-noising algorithms have been developed based on it for regular RGB images and videos. We would like to investigate the difference between radar video and regular RGB video and study the applicability of MRF for radar video de-noising.

- Detection of other kinds of objects

In addition to power lines, other visible objects in the active millimeter-wave radar video include flying birds, roads, moving vehicles, power line towers, and buildings. The detection of flying birds is particularly useful because flying birds can also be threatening to the safety of aircrafts. But the problem is that it is hard to collect radar video that contains flying birds for investigation. For detecting other types of objects such as roads, we need to further investigate the features these objects have in the radar video. Since the radar will not be affected by the lighting condition, object detection at night will be a particularly useful application.

- Fusion of radar video and regular video

As an example application for this topic, it is usually too dark for a train pilot to see the outside environment at night. If the daytime video of outside environment is available, we could extract features from the radar video to synchronize and register the radar video with the regular daytime video so that the outside environment could be presented to the pilot of the train. This can provide useful contextual environment information to the train pilot. For night-time

surveillance applications using a helicopter, after the registration of the features, the daytime video containing the scene surrounding the building or the target can be presented to the pilot of the helicopter.

In general, the characteristics of the radar video that it is not affected by the lighting condition make it very useful for detecting objects at night. More investigation in this aspect can be conducted so that it can reach its full capability in reliable detection at night and under poor weather condition, and can be incorporated into a multimodal system as a useful modality for the robust detection of objects.

References

[1] J. S. Harris, "Data Show 50 U.S.-registered Helicopters Involved in Wire-strike Accidents From 1996 Through 2000," *Helicopter Safety*, vol. 28, no. 4, 2002.

[2] R. Appleby, P. Coward and J. Sanders-Reed, "Evaluation of a Aassive Millimeter Wave (PMMW) Imager for Wire Detection in Degraded Visual Conditions," *Proceedings of SPIE*, vol. 7309, April 2009.

[3] C. Migliaccio, B. Nguyen, C. Pichot, N. Yonemoto, K. a. Y. K. Yamamoto, H. Nasui, W. Mayer, A. Gronau and W. Menzel, "Millimeter-Wave Radar for Rescue Helicopters," in *9th International Conference on Control, Automation, Robotics and Vision*, 2006.

[4] K. Sarabandi and M. Park, "A Radar Cross-Section Model for Power Lines at Millimeter-Wave Frequencies," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 9, pp. 2353-2360, 2003.

[5] K. Sarabandi, L. Pierce, Y. Oh and F. Ulaby, "Power Lines: Radar Measurements and Detection Algorithm for Polarimetric SAR Images," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 2, pp. 632-643, 1994.

[6] M. Park, "Millimeter-Wave Polarimetric Radar Sensor for Detection of Power Lines in Strong Clutter Background," University of Michigan, 2003.

[7] L. Shapiro and G. Stockman, Computer Vision, Prentice Hall, 2001.

[8] D. S. Goshi, Y. Liu, K. Mai, L. Bui and Y.-C. Shih, "Recent Advances in 94 GHz FMCW Imaging Radar Development," in *Microwave Symposium Digest*, 2009.

[9] D. S. Goshi, Y. Liu, K. Mai, L. Bui and Y.-C. Shih, "Cable Imaging with an Active W-band Millimeter-Wave Sensor," in *Microwave Symposium Digest*, 2009.

[10] R. Appleby and R. Anderton, "Millimeter-wave and Submillimeter-wave Imaging for Security and Surveillance," *Proceedings of the IEEE*, vol. 95, no. 8, pp. 1683-1690, 2007.

[11] D. Petkie, F. De Luciab, C. Castob, P. Helmingerc, E. Jacobsd, S. Moyerd, S. Murrille, C. Halfordf, S. Griffinf and C. Franckg, "Active and Passive Millimeter and Sub-Millimeter-wave Imaging," *Proceedings of SPIE*, 2005.

[12] S. Oka, H. Togo, N. Kukutsu and T. Nagatsuma, "Latest Trends in Millimeter-Wave Imaging Technology," *Progress in Electromagnetics Research*, vol. 1, pp. 197-204, 2008.

[13] E. Grossman and A. Miller, "Active Millimeter-Wave Imaging for Concealed Weapon Detection," *Proceedings of SPIE*, vol. 5077, p. 62, 2003.

[14] D. Sheen, D. McMakin and T. Hall, "Three-dimensional Millimeter-Wave Imaging for Concealed Weapon Detection," *IEEE Transactions on Microwave Theory and Techniques*, vol. 49, no. 9, pp. 1581-1592, 2001.

[15] K. Sarabandi and M. Park, "Millimeter-wave Radar Phenomenology of Power Lines and a Polarimetric Detection Algorithm," *IEEE Transactions on Antennas and Propagation*, vol.

47, no. 12, pp. 1807-1813, 1999.

[16] P. Hough, "Method and Means for Recognizing Complex Patterns". US Patent 3,069,654, December 1962.

[17] R. Duda and P. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM,* vol. 15, no. 1, pp. 11-15, 1972.

[18] C. Kimme, D. Ballard and J. Sklansky, "Finding Circles by an Array of Accumulators," *Communications of the ACM,* vol. 18, no. 2, pp. 120-122, 1975.

[19] S. Tsuji and F. Matsumoto, "Detection of Ellipses by a Modified Hough Transformation," *IEEE Transactions on Computers,* vol. 100, no. 8, pp. 777-781, 1978.

[20] D. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition,* vol. 13, no. 2, pp. 111-122, 1981.

[21] L. Xu, E. Oja and P. Kultanen, "A New Curve Detection Method: Randomized Hough Transform (RHT)," *Pattern Recognition Letters,* vol. 11, no. 5, pp. 331-338, 1990.

[22] N. Kiryati, Y. Eldar and A. Bruckstein, "A Probabilistic Hough Transform," *Pattern Recognition,* vol. 24, no. 4, pp. 303-316, 1991.

[23] X. Yu, H. Lai, S. Liu and H. Leong, "A Gridding Hough Transform for Detecting the Straight Lines in Sports Video," in *IEEE International Conference on Multimedia and Expo (ICME),* 2005.

[24] J. Illingworth and J. Kittler, "A Survey of the Hough Transform," *Computer Vision, Graphics, and Image Processing,* vol. 44, no. 1, pp. 87-116, 1988.

[25] V. Leavers, "Which Hough Transform?," *CVGIP: Image Understanding,* vol. 58, no. 2, pp. 250-264, 1993.

[26] W. Press, "Discrete Radon Transform Has an Exact, Fast Invere and Generalizes to Operations Other Than Sums Along Lines," *Proceedings of the National Academy of Sciences,* vol. 103, no. 51, pp. 19249-19254, 2006.

[27] M. Fischler and R. Bolles, "Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[28] Y. Cheng and S. Lee, "A New Method for Quadratic Curve Detection Using K-RANSAC with Acceleration Techniques," *Pattern Recognition,* vol. 28, no. 5, pp. 663-682, 1995.

[29] M. Fischler, J. Tenenbaum and H. Wolf, "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique," *Computer Graphics and Image Processing,* vol. 15, no. 3, pp. 210-223, 1981.

[30] N. Merlet and J. Zerubia, "New Prospects in Line Detection by Dynamic Programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 18, no. 4, pp. 426-431, 1996.

[31] C. Steger, "An Unbiased Detector of Curvelinear Structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 2, no. 20, pp. 113-125, 1998.

[32] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active Contour Models," *International Journal of Computer Vision,* vol. 1, no. 4, pp. 321-331, 1988.

[33] E. Mortensen and W. Barrett, "Intelligent Scissors for Image Composition," in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, 1995.

[34] P. Perez, A. Blake and M. Gangnet, "JetStream: Probabilistic Contour Extraction with Particles," in *Proceedings of 8th International Conference on Computer Vision (ICCV)*, 2001.

[35] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, no. 3, pp. 249-268, 2007.

[36] S. K. Murthy, "Automatic Construction of Decision Trees from Data: A Multi-disciplinary Survey," *Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 345-389, 1998.

[37] L. A. Breslow and D. W. Aha, "Simplifying Decision Trees: A Survey," *The Knowledge Engineering Review*, vol. 12, no. 1, pp. 1-40, 1997.

[38] F. Rosenblatt, Principles of Neurodynamics, Spartan Book, 1962.

[39] G. P. Zhang, "Neural Networks for Classification: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 30, no. 4, pp. 451-462, 2000.

[40] R. Gutierrez-Osuna, "Texas A&M University Pattern Recognition and Intelligent Sensor Machines Lecture Notes 10: Fisher Linear Discriminants," [Online]. Available: http://research.cs.tamu.edu/prism/lectures/pr/pr_l10.pdf.

[41] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques, The MIT Press, 2009.

[42] T. Mitchell, Machine Learning, McGraw Hill, 1997.

[43] S. Arya, T. Malamatos and D. Mount, "Space-Time Trade-offs for Approximate Nearest Neighbor Searching," *Journal of the ACM*, vol. 57, no. 1, pp. 1-54, 2009.

[44] V. Vapnik, The Nature of Statistical Learning Theory, New York: Springer-Verlag, 2000.

[45] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting," in *Computational Learning Theory*, Springer, 1995, pp. 23-37.

[46] Y. Freund and R. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771-780, 1999.

[47] R. Caruana and A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms," in *23rd International Conference on Machine Learning*, 2006.

[48] A. Yilmaz, O. Javed and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1-45, 2006.

[49] R. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35-45, 1960.

[50] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," University of North Carolina at Chapel Hill, Chapel Hill, NC, 1995.

[51] T. Broida and R. Chellappa, "Estimation of Object Motion Parameters from Noisy Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 90-99, 1986.

[52] D. Beymer and K. Konolige, "Real-time Tracking of Multiple People Using Continuous

Detection," in *IEEE International Conference on Computer Vision Frame-Rate Workshop*, 1999.

[53] R. Rosales and S. Sclaroff, "3D Trajectory Recovery for Tracking Multiple Objects and Trajectory Guided Recognition of Actions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[54] M. Isard and A. Blake, "Condensation - Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5-28, 1998.

[55] M. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174-188, 2002.

[56] K. Nummiaro, E. Koller-Meier and L. Van Gool, "An Adaptive Color-Based Particle Filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99-110, 2003.

[57] S. Zhou, R. Chellappa and B. Moghaddam, "Visual Tracking and Recognition Using Appearance-Adaptive Models in Particle Filters," *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1491-1506, 2004.

[58] C. Yang, R. Duraiswami and L. Davis, "Fast Multiple Object Tracking via a Hierarchical Particle Filter," in *IEEE International Conference on Computer Vision*, 2005.

[59] J. Vermaak, A. Doucet and P. Perez, "Maintaining Multi-Modality Through Mixture Tracking," in *IEEE International Conference on Computer Vision*, 2003.

[60] Z. Khan, T. Balch and F. Dellaert, "An MCMC-based Particle Filter for Tracking Multiple Interacting Targets," in *European Conference on Computer Vision*, 2004.

[61] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier and L. Van Gool, "Robust Tracking-by-detection Using a Detector Confidence Particle Filter," in *IEEE International Conference on Computer Vision*, 2009.

[62] H. Peng, F. Long and C. Ding, "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 1226-1238, August 2005.

[63] A. Doucet, N. D. Freitas and N. Gordon, Sequential Monte Carlo Methods in Practice, Springer Verlag, 2001.

[64] K. Okuma, A. Taleghani, N. d. Freitas, J. J. Little and D. G. Lowe, "A Boosted Particle Filter: Multi-target Detection and Tracking," Springer, 2004, pp. 28-39.

[65] P. Viola and M. Jones, "Robust Real-time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137-154, 2004.