Train track extraction and haze removal

A Project Report

submitted by

KAMMULA RUPADITYA

in partial fulfilment of the requirements

for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2016

THESIS CERTIFICATE

This is to certify that the thesis titled **Train track extraction and haze removal** submitted by **Kammula Rupaditya** to the Indian Institute of Technology, Madras, for the award of the degree of **Dual Degree**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. A.N. Rajagopalan
(Research Guide)
Professor
Dept. of Electrical Engg.
IIT Madras
Chennai - 600 036.

Place: Chennai
Date:

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my project guide Dr. A. N. Rajagopalan for his valuable guidance and motivation throughout the course of the project. I am very grateful to him for providing his valuable time to guide me during my project.

It is a privilege to be a student at IIT Madras. The time I got to spend at the Image Processing and Computer Vision Lab has been extremely fruitful, given the extremely healthy research environment that exists in the lab. Thanks to all my teachers for the academic insight obtained from them. I also acknowledge the excellent facilities provided by the institute to the students.

I am indebted to my parents for their unconditional love, support and motivation. I dedicate this thesis to them.

ABSTRACT

KEYWORDS: Train tracks, haze removal, dark channel prior.

Safety is very important for any railway system. A train is controlled by its driver and the safety of a train depends on the vigilance of the driver. The driver of the train must be constantly aware of the tracks and the surrounding environment. The the field of view of a train driver must contain the space between two rails in front of the train and the near lateral area (left and right side) of these rails. A support system that can quickly identify such regions can be very helpful by improving the reaction time of the driver.

This thesis is an attempt to develop a safety system for trains which can detect the train tracks and also remove haze in hazy scenarios. The detected tracks can help the driver or an automation to quickly identify the region of interest and look for possible obstacles. Haze removal helps in improving the visibility of the driver and makes it possible to detect tracks in bad weather. A well known technique used for haze removal for outdoor scenes uses a prior called the dark channel prior to remove haze from the hazy scene. An efficient implementation of this technique is discussed.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

ROI	Region of Interest
POV	Point of View
HT	Hough Transform
DCP	Dark Channel Prior
ERTMS	European Rail Traffic Management System
ETCS	European Train Control System
DEQUE	Double Ended Queue
FPS	Frames Per Second
GPU	Graphics Processing Unit
PC	Personal Computer

CHAPTER 1

INTRODUCTION

Sensor based support for driver assistance to improve safety has been in development for auto-mobiles recently. These vary from total automation to alerting the driver of possible collisions. Similar support systems can also be envisaged for railways. Detecting the tracks becomes a fundamental task for such systems. It can help in delimiting the search space for possible obstacles and signs. This can help the driver in quickly assessing the region of interest (ROI) and can also be used for automation of the obstacle detection process.

Yet another useful functionality of these type of support systems is haze removal. Haze limits the driver's perception and can make it hard to notice the signs and to fixate on the tracks. Haze removal can help the driver get a better view of the tracks, signs and the environment.

1.1 Previous Works

Existing train anti-collision systems such as the european railway traffic management system (ERTMS) or european train control system (ETCS) rely on infrastructure equipments. They work by having configured fixed blocks and track-side equipment that monitors individual signals and passing this information to trains via track-mounted transponders or radio link. This permits the train to reach its maximum permitted speed while maintaining safe braking distance. Some safety systems exist for rail-road crossing which uses stationary cameras mounted on the ground as described in Sheikh *et al.* (2004), Xue *et al.* (2008) and Silar and Dobrovolny (2013).

The problem of track detection is very similar to detecting roads. There are road lane markings detection systems that are mature enough to be commercially used for driver assistance. Some of the road detection techniques such as Yu and Jain (1997) Wang *et al.* (2004) use lane markings as guidance and some variant of Hough transform (HT)

to detect the lanes. These techniques can be extended to detect tracks. One example of track detection is Kaleli and Akgul (2009) where a dynamic programming based approach is used to detect tracks. Other techniques also exist for track detection (Ross (2010)) which assumes a clothoid model to represent the tracks and uses it to extract them. Maire and Bigdeli (2010) take a rectangular region in-front of the camera and detect a pair of lines in it and use it to detect tracks.

Haze removal is a challenging problem because haze depends on the depth of the scene. A single image haze removal is an under-constrained problem. Many methods have been proposed that use multiple images or additional depth information. Polarization based methods (Schechner *et al.* (2001), Shwartz *et al.* (2006)) use multiple images taken with different polarizations. Depth based methods (Narasimhan and Nayar (2003)) require additional depth information from user input or known 3D models.

Single image haze removal uses strong priors or assumptions to compensate for the limited constraints. Tan (2008) is based on the observation that haze free image must have higher contrast compared to its input hazy image and removes haze by maximizing the local contrast of the input hazy image. An image prior called dark channel prior (DCP) was introduced in He *et al.* (2011) which helps in removing the haze by estimating the transmission map of the input hazy image.

1.2 Objectives and Scope

Railway tracks usually have a high contrast with the environment and are distinguishable. They have certain properties which can be exploited to make the detection possible. The train moves along the tracks so they are not affected by motion blur i.e they still appear sharp while the other edges in the environment are blurred. From the point of view (POV) of the engine, they also appear as a pair of lines moving towards each other. These properties can be observed in the images included in Fig 1.1 which shows 4 images in which we would like to detect the tracks. We first use simple edge detectors to find the edges followed by line detection using Hough transform. Then, we exploit these properties to detect the tracks given a scene that contains them.

We limit ourselves to situations where the tracks are completely straight or approximately straight. For cases where there is a curvature or when multiple tracks cross over,

a different technique will be needed.

Figure 1.1: Examples of rail tracks.

Fig 1.2 shows some examples of hazy images from which we aim to remove the haze component. Many algorithms exist to remove haze. However, for practical purposes we need one that can be used in real-time. A dark channel prior (DCP) based approach works here because it is an outdoor scene which satisfies the DCP property and the algorithm can be designed to run fast and reliably. We skip the refinement step in the standard DCP method proposed by He *et al.* (2011) to save on speed as this doesn't affect the output too much and is not an absolute necessity for videos. The most demanding operation of this method is estimation of the dark channel. We present an optimized algorithm using the data structure double ended queue (DEQUE) to improve the running time of this operation by 7-8 times when compared to the normal method.

Figure 1.2: Examples of hazy images of tracks.

1.3 Organization of the thesis

In chapter 2, we begin by covering the technical background related to track extraction. Then, our algorithm is discussed in detail.

In chapter 3, we discuss haze removal by using dark channel prior. The idea of dark channel prior is first presented followed by a detailed discussion of our algorithm.

In chapter 4, the performance analysis of the algorithm is presented along with results.

In chapter 5, we present the conclusions of the thesis.

CHAPTER 2

TRACK EXTRACTION

2.1 Introduction

In this chapter, an algorithm for track extraction will be discussed. Railway tracks are differentiable from the environment and an edge filter will help highlight the tracks. We assume that the scene is viewed from the POV of the driver. This makes the tracks appear like a pair of lines moving towards each other. As the train moves along the tracks, the tracks do not appear blurred in the image since they have a uniform composition while the other strong edges in the environment get blurred. These properties can be observed in Fig 2.1. This assumption is exploited to devise an algorithm for extraction of the rail pairs which form the tracks. The Sobel operator is used for edge filtering and the lines are detected using HT.

Figure 2.1: image of a rail tracks.

2.2 Technical Background

In this section, we discuss the theory of Sobel operator and Hough transform to detect straight lines which are used in our method .

2.2.1 Sobel edge filter

The Sobel operator is used for edge detection. The Sobel operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions and is therefore relatively inexpensive in terms of computations.

The operator uses two 3x3 kernels which are convolved with the input image A to produce two gradient images, one for the horizontal Gx and one for the vertical Gy. The computations for Gx, Gy and the final gradient magnitude image G are as follows:

$$Gx = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \tag{2.1}$$

$$Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A \tag{2.2}$$

$$G = \sqrt{Gx^2 + Gy^2} \tag{2.3}$$

Figure 2.2: Example of an image and its gradient magnitude.

2.2.2 Hough transform for detecting lines

Hough transform can be used to detect straight lines (Duda and Hart (1972)). In general, a straight line $y = mx + c$ can be uniquely represented as a point (m,c) in the parameter space. However, there is a problem when m is unbounded (i.e vertical) with this representation. This would give rise to unbounded values of the slope parameter m. Thus, for computational reasons, the Hesse normal form $r = x\cos\theta + y\sin\theta$ where r is the distance from origin to the closest point on the line and θ is the angle between the X-Axis and the line joining the closet point on the line to origin is used. Therefore it is now possible to uniquely represent each line in the (r,θ) parameter space (also known as Hough space)within bounded values.

For a given point (x_0, y_0) in the image, we have all the possible lines passing through it as $r = x_0\cos\theta + y_0\sin\theta$. The plot of r vs θ is a sinusoid. As an example, plot for x_0 = 8 y_0 = 6 is shown in Fig 2.3.

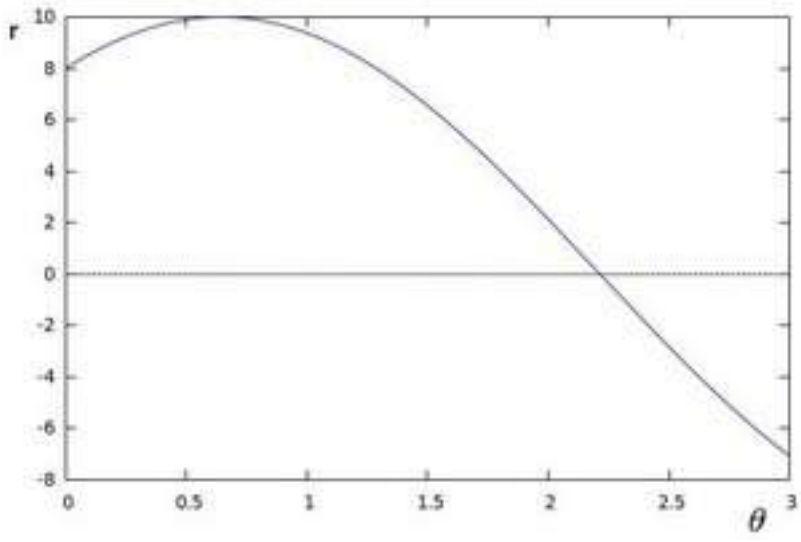

Figure 2.3: Plot of a single sinusoid.

Thus, each point on the image produces a sinusoid in the Hough space graph r vs θ. The intersection of sinusoids implies that there is a line passing through these points. The above plot combined with plot of x_0 = 4 y_0 = 9 and x_0 = 12 y_0 = 3 is shown in Fig 2.4

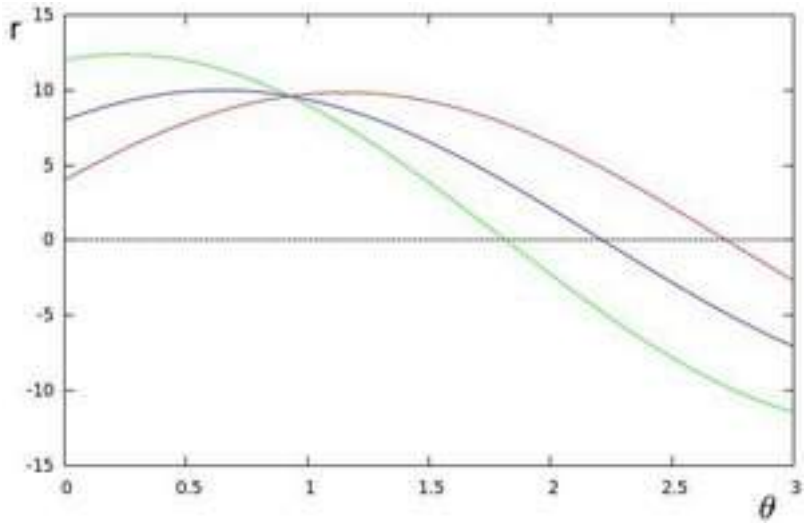

Figure 2.4: Plot of multiple sinusoids intersecting at a point.

The more number of curves intersecting at a point means that there are more number of points on that line. A desired threshold can be set to use those intersection points as lines.

Fig 2.5 shows an example of the lines detected by the HT. The left image is the original image and the detected lines found in the input image are shown in the right image.

Figure 2.5: Example of lines detected by HT.

2.3 Tracks extraction

In our method, we start by computing the gradient image and threshold it to obtain a binary image. Then we use HT to detect all the lines in the binary image. We then find all possible pairing of the detected lines and check each pair for the conditions that rail tracks will satisfy (Equations 2.5, 2.6, 2.9 and 2.10). Finally of all the pairs satisfying these conditions, we pick that pair which is of the longest length. The flow-chart of our algorithm is given in Fig 2.6

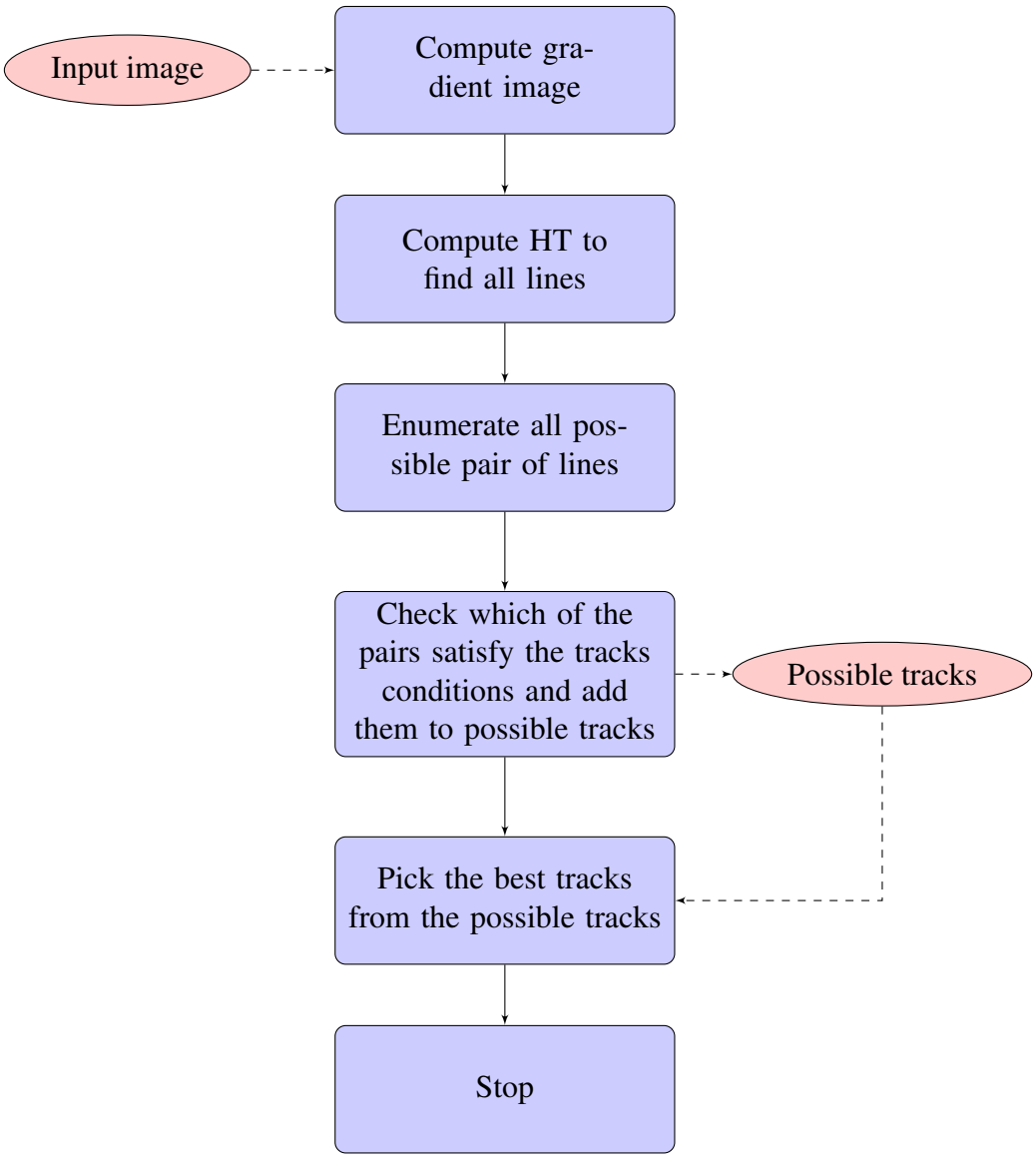

Figure 2.6: Flow-chart of the track extraction algorithm.

2.3.1 Computing the Gradient image

To represent images, the OPENCV Mat data-structure is used. We begin by applying a Gaussian blur with a kernel of size 3 x 3 pixels. This is followed by a conversion to Gray scale image. Inbuilt OpenCV functions are used for both these operations. We then proceed to compute the gradient image using Sobel operator. The gradient-x and gradient-y are computed and then combined in a 3:1 ratio as opposed to 1:1 ratio. This is done to highlight the rail edges which have larger gradient along x than along y. This will also suppress those edge which have stronger y gradient which will be removed later when we threshold the image. Fig 2.7 shows the gradient image of Fig 2.1.

Figure 2.7: Gradient image of Fig 2.1.

2.3.2 Detecting straight lines

Now we threshold the gradient image to obtain a binary image. This operation is as follows

$$B(x,y) = \begin{cases} 255 & \text{if} G(x,y) > t_0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

where $B(x,y)$ is the intensity of the binary image at pixel (x,y) and $G(x,y)$ is the intensity of the gradient image at pixel (x,y). t_0 is the threshold for the binary image. Fig 2.7 shows the binary image obtained when a threshold of 150 is used (i.e t_0 = 150). The threshold is chosen to be 150 because we found that the rail edges have magnitude

stronger than 150 while other unimportant edges had around 80-130 magnitude. This way the rail edges are preserved while removing unimportant edges.

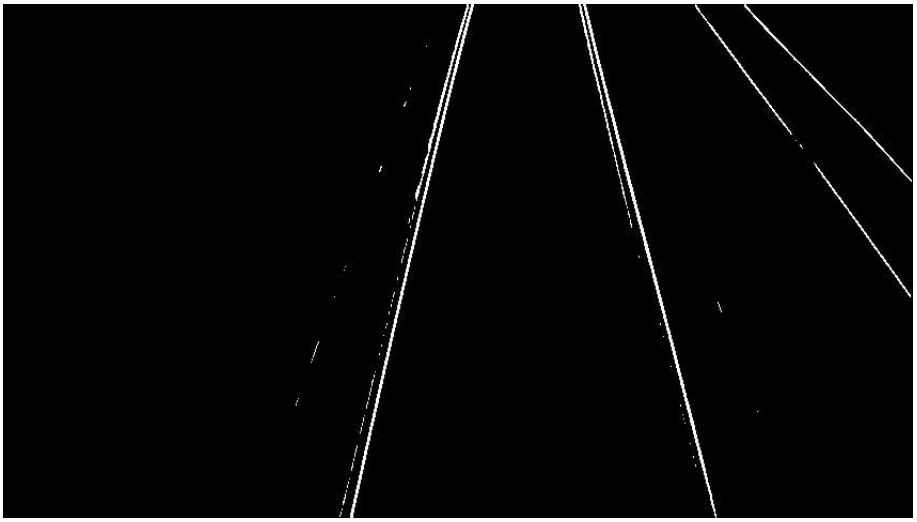

Figure 2.8: The binary image of Fig 2.6 when threshold of 150 is used.

Next, we proceed to apply the Hough transform on the binary image to get the lines. A line is represented by the data structure Vec4i which is a vector of four integers. These four integers hold the end points of the line segment. For end points (x_1, y_1) and (x_2, y_2), Vec4i is organised as $[x_1, y_1, x_2, y_2]$. The OpenCV function **Houghlines** is used to get the lines. Although this function returns line segments, we refer to them as lines as they can be extended. Fig 2.9 shows all the lines detected by HT on Fig 2.8 in red color. Fig 2.10 shows the lines as detected on the original image.

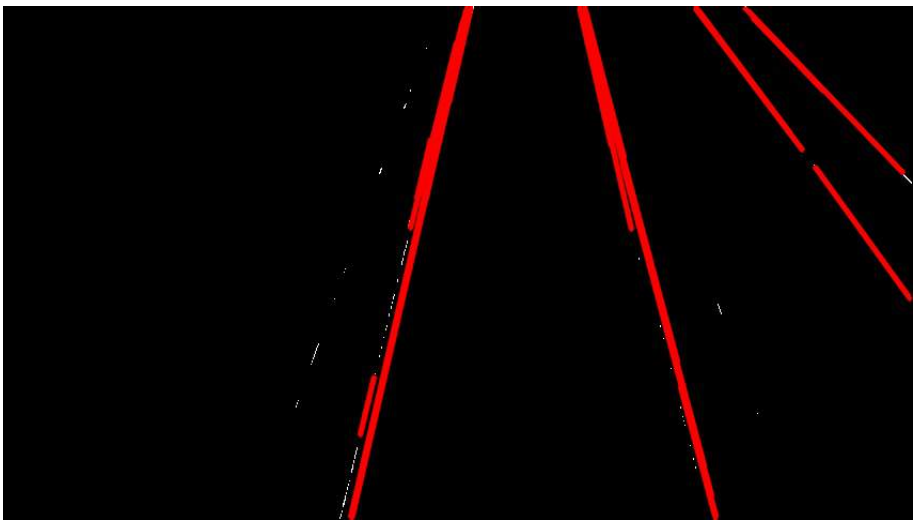

Figure 2.9: All the lines detected by HT on Fig 2.7.

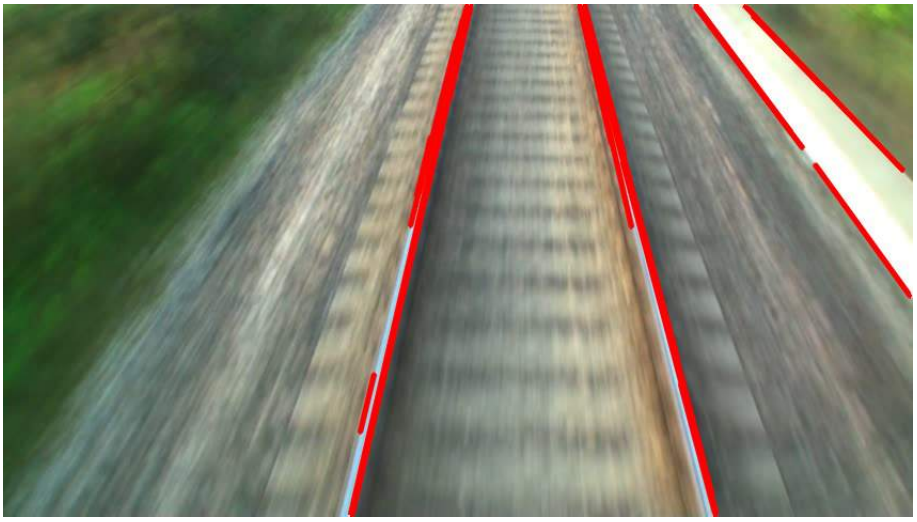

Figure 2.10: All the lines detected in Fig 2.1.

2.3.3 Selecting the tracks from the set of rail pairs

Next, we enumerate all the possible pair of lines from the set of lines detected by the HT and check them for two conditions to determine if they can indeed be train tracks.

The first condition is the intersection criteria which checks if a pair of line segments intersect with each other. For this purpose, we compare the end points of both the line segments. Let the first line segment have end points (x_1, y_1),(x_2, y_2) and the second line segment have end points (X_1, Y_1),(X_2, Y_2). If the line segments do not intersect, then the following equations are satisfied i.e

$$(x_1 - X_1)(x_2 - X_2) > 0 \tag{2.5}$$

$$(y_1 - Y_1)(y_2 - Y_2) > 0 \tag{2.6}$$

The second condition is the slope criteria which follows from our assumption of drivers POV. From the POV of the driver, the lines corresponding to tracks should appear as converging at a point near the center starting from the bottom of the frame. With the assumption that the camera is positioned between the pair of lines that form the tracks, the left line will have a positive slope and the right line will have a negative slope. The slopes of the lines previously assumed are given by the equations 2.7 and

2.8 respectively. The left rail and right rail are first marked by their respective x co-ordinates. Then we check the slope of the left rail and right rail. These two steps can be combined and written in a single conditions as Equations 2.9 and 2.10

$$m_1 = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.7}$$

$$m_2 = \frac{Y_2 - Y_1}{X_2 - X_1} \tag{2.8}$$

$$(X_1 - x_1) * m_1 > 0 \tag{2.9}$$

$$(x_1 - X1) * m_2 > 0 \tag{2.10}$$

If a pair of lines satisfy both the criteria, they are added to a set of possible tracks. We finally pick the best pair from the set of possible tracks. The pair with the longest combined length is taken as the best pair (i.e, combined length is the cost function). For the lines assumed earlier, the cost function can be written as

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2} \tag{2.11}$$

Figs 2.11 and 2.12 show best pair and the next best pair of lines detected as tracks respectively.

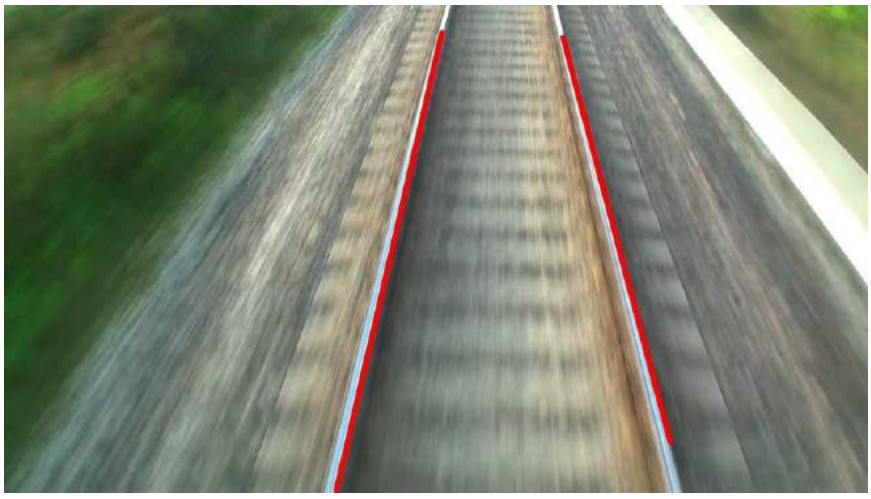

Figure 2.11: Best pair of lines detected as tracks.

Figure 2.12: Next best pair of lines.

After this, we extend the pair of lines in both directions till they meet in one direction and to the bottom of the frame in the other so that the entire track is selected. Fig 2.13 shows the tracks extended as mentioned. We can also use a mask to reduce the region to look for lines if there are other strong edges in the environment such as electric poles on track-side or another set of tracks on the side.

Figure 2.13: The final output of our algorithm.

More examples of tracks detection are given in Fig 2.14. The left column contains the input images while the right column contains the corresponding output by our algorithm.

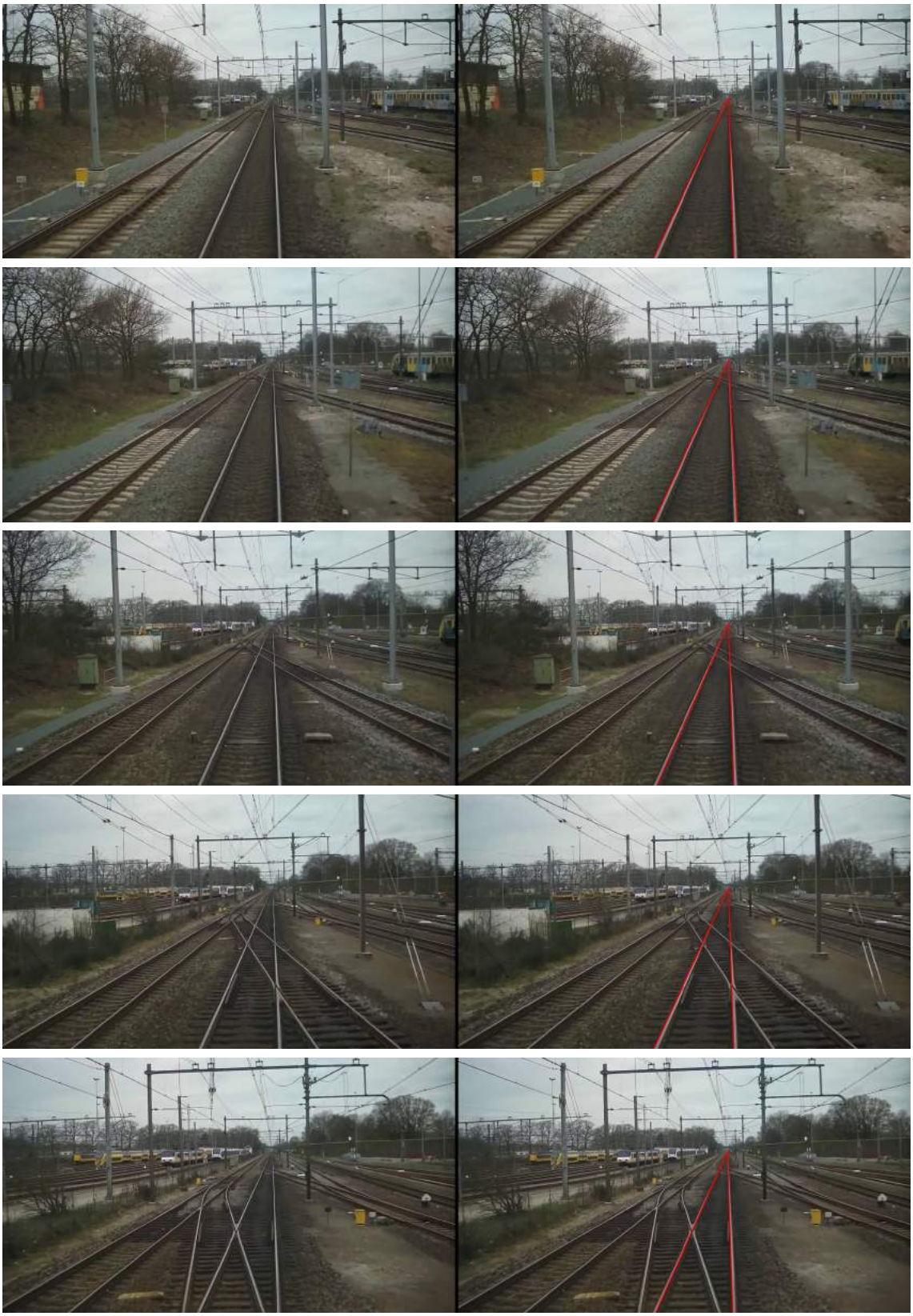

Figure 2.14: Examples of rail tracks detection.

CHAPTER 3

HAZE REMOVAL

3.1 Introduction

The model used to describe hazy images is given by

$$I(x) = J(x)t(x) + A(1 - t(x)) \tag{3.1}$$

where I is the observed image, J is the scene radiance(desired image), A is the global atmospheric light and t is the medium transmission. The goal of haze removal is to recover J, A and t given I which is an N-pixel image. Thus, we have $3N$ constraints to estimate $4N$+3 values.

Figure 3.1: A hazy image.

3.1.1 Estimating atmospheric light

The brightest pixels in the hazy image are considered to be the most haze-opaque. This is true when atmospheric light is the only source of illumination and sunlight can be ignored. The scene radiance is given by

$$J(x) = R(x)A \tag{3.2}$$

where $R(x)$ is the reflectance of the scene points. In this case, the haze imaging equation can be rewritten as

$$I(x) = R(x)At(x) + A(1 - t(x)) \tag{3.3}$$

When pixels at infinite distance exist(i.e $t(x)$=0) which is often the case for the images we are concerned with because of the presence of sky, the atmospheric light A can be considered to be equal to the brightest pixels of the image.

3.1.2 Dark channel prior

Dark channel prior is based on the observation that in general non-sky patches of an outdoor image, atleast one color channel has some pixels with intensities close to zero. Which means that the minimum of the intensities in such a patch is close to zero.

For an image J, its dark channel J^{dark} is given by

$$J^{dark}(x) = \min_{y \epsilon \Omega(x)} (\min_{c \epsilon B,G,R} (J^c(y))) \tag{3.4}$$

where c is a color channel of J and Ω(x) is a local patch centred at x. The dark channel prior observation says that, $J^{dark} \rightarrow 0$ for pixels in the non-sky region.

3.1.3 Estimating the transmission

Rearranging equation 3.1, we get

$$\frac{I^c(x)}{A^c} = t(x)\frac{J^c(x)}{A^c} + 1 - t(x) \tag{3.5}$$

for each channel. Assuming that the transmission in a local patch Ω(x) is constant, let it be $\tilde{t}$(x). Applying min on both sides, we have

$$\min_{y \epsilon \Omega(x)} \left(\min_{c \epsilon B,G,R} \left(\frac{I^c(y)}{A^c} \right) \right) = \tilde{t}(x) \min_{y \epsilon \Omega(x)} \left(\min_{c \epsilon B,G,R} \left(\frac{J^c(y)}{A^c} \right) \right) + 1 - \tilde{t}(x) \tag{3.6}$$

As J is the haze free image, the dark channel of this should be zero. Therefore,

$$\min_{y\epsilon\Omega(x)}(\min_{c\epsilon B,G,R}(J^c(y)) = 0 \tag{3.7}$$

Also, A^c is positive.

$$\min_{y\epsilon\Omega(x)}\left(\min_{c\epsilon B,G,R}\left(\frac{J^c(y)}{A^c}\right)\right) = 0 \tag{3.8}$$

$\tilde{t}$ can be estimated as

$$\tilde{t}(x) = 1 - \min_{y\epsilon\Omega(x)}\left(\min_{c\epsilon B,G,R}\left(\frac{I^c(y)}{A^c}\right)\right) \tag{3.9}$$

$\min_{y\epsilon\Omega(x)}\left(\min_{c\epsilon B,G,R}\left(\frac{I^c(y)}{A^c}\right)\right)$ is the dark channel of the normalized hazy image. Note that, this is not a good approximation for sky regions as hazy image is very close to A^c. Hence, in the sky region, we have

$$\min_{y\epsilon\Omega(x)}\left(\min_{c\epsilon B,G,R}\left(\frac{I^c(y)}{A^c}\right)\right) \to 1 \tag{3.10}$$

3.1.4 Recovering the scene radiance

Rearranging equation 3.1,

$$J(x) = \frac{I(x) - A}{t(x)} + A \tag{3.11}$$

When t(x) is close to zero, the noise gets amplified. So, we set a lower bound to t(x) which is t_0

$$J(x) = \frac{I(x) - A}{\max(t(x), t_0)} + A \tag{3.12}$$

3.2 Our Model

We begin by estimating the atmospheric light A and dark channel of the input hazy image. Next we use the estimated A and dark channel to compute the transmission

map of the image. Finally, we use the transmission map to remove haze from the input image. The flow-chart of our algorithm is shown in Fig 3.2.

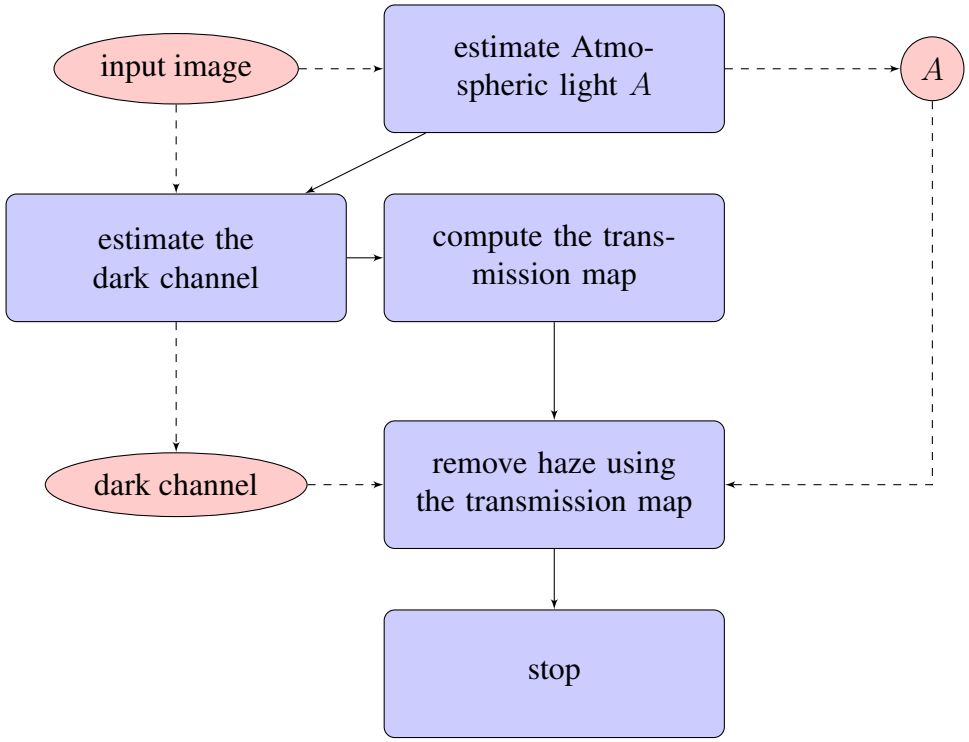

Figure 3.2: Flow-chart of haze removal algorithm.

3.2.1 Estimating atmospheric light A

To estimate the atmospheric light A, we split the image into the R, G, B channels and calculate the max intensity in each channel. Equation 3.13 shows how each channel of A is calculated.

$$A^c = \max I^c \quad c \epsilon B, G, R \tag{3.13}$$

3.2.2 Estimating the dark channel

To calculate the dark channel for a pixel (x, y) we take a patch of size 15 x 15 centred at (x, y) and find the minimum of the minimum of each channel(B,G,R). This is done at all the pixels of the hazy image. This whole process can be optimized by using a double-ended queue(DEQUE) data structure. A DEQUE is an abstract data type which behaves as a queue to which elements can be added or removed from the front(head) or back(tail). While the insertion/deletion operation is inefficient at the middle of the

DEQUE, it can be performed in O(1) at the head/tail ends. This makes it very useful in designing an efficient algorithm to estimate the dark channel. In our algorithm, we insert and remove each and every pixel only once and we do it at the head/tail ends of the DEQUE. For an NxN image and patch size KxK, this reduces the time complexity from $O(N^2K^2)$ to $O(N^2)$. We use the inbuilt C++ implementation for which the documentation can be found at cplusplus.com. The algorithm is as follows:

Algorithm 1 Estimating Dark Channel using DEQUE

Require: (a) Matrices src,dark_bgr, temp_channel, dark_channel (b) A DEQUE Dq (c) WIN_SIZE

src matrix is the input image
for i from 0 to N_{rows} **do**
  for j from 0 to N_{cols} **do**
    dark_bgr(i,j) = min of B,G,R channels of src(i,j)
  end for
end for
win_center = $\frac{\text{WIN\_SIZE}-1}{2}$
for i from 0 to N_{rows} **do**
  clear Dq
  for j from 0 to N_{cols} **do**
    while Dq$\neq \Phi$ **and** front element y co-ordinate < j-win_center **do**
      remove the front element Dq
    end while
    while Dq$\neq \Phi$ **and** back element y co-ordinate < j-win_center **and** back element intensity$\geq$intensity of current pixel of dark_bgr **do**
      remove the back element of Dq
    end while
    add dark_bgr(i,j) to the back of Dq
    temp_channel(i,j) = front element of Dq
  end for
end for
for j from 0 to N_{cols} **do**
  clear Dq
  for i from 0 to N_{rows} **do**
    while Dq$\neq \Phi$ **and** front element x co-ordinate < i-win_center **do**
      remove the front element Dq
    end while
    while Dq$\neq \Phi$ **and** back element x co-ordinate < i-win_center **and** back element intensity$\geq$intensity of current pixel of temp_channel **do**
      remove the back element of Dq
    end while
    add temp_channel(i,j) to the back of Dq
    dark_channel(i,j) = front element of Dq
  end for
end for

We first compute dark_bgr matrix which has the minimum intensity of the three

channels of each pixel.i.e

$$\text{dark\_bgr}(x, y) = \min_{c \epsilon B,G,R} I^c(x, y) \tag{3.14}$$

which is then used for the next part. This algorithm then makes two passes. In the first pass, we go across the columns and insert the pixels into the DEQUE. Because of the manner in which we insert elements into the DEQUE, the first element of deque at each pixel say (x, y), is the minimum of all the k-pixels ($(x - k + 1, y)$ to (x, y)) before it and this value is stored in a new matrix at the location $(x - \frac{k-1}{2}, y)$. Let this new matrix be called temp_channel. The new matrix elements can be written as

$$\text{temp\_channel}(x, y) = \min_{\text{x-k+1} \leq \text{i} \leq \text{x}} \text{dark\_bgr}(i, y) \tag{3.15}$$

This new array is then used to make a second pass, now across the rows. Finally we will have a new array which has the dark channel of the input image. The computation of dark_channel matrix from the temp_channel matrix can be written as

$$\text{dark\_channel}(x, y) = \min_{\text{y-k+1} \leq \text{j} \leq \text{y}} \text{temp\_channel}(x, j) \tag{3.16}$$

Fig 3.3 shows the dark channel computed by out algorithm.

Figure 3.3: Dark channel of the hazy image Fig 3.1.

3.2.3 Computing the transmission map

Next, we compute the transmission map following the procedure described in subsection 3.1.3. The atmospheric light and the dark channel estimated earlier are used for this step. Fig 3.4 shows the tranmission map computed using the dark channel Fig 3.3

Figure 3.4: Transmission map of the hazy image Fig 3.1.

3.2.4 Removing haze

We finally use the transmission map computed earlier to remove the haze. This step is based on equation 3.12. We use the threshold $t_0 = 0.1$. Fig 3.5 shows the final output after removing haze.

Figure 3.5: Haze removed image from the Fig 3.1.

3.3 Results

Fig 3.7 shows some results of our method. The left side column has the input hazy images and the corresponding outputs are shown in the right side column.We can notice that there is still haze around the objects like poles, sign-boards and trees. This is because we have skipped the refinement step as proposed in He *et al.* (2011) since it takes too long to process and is impractical for real-time videos. The alternative to this step as proposed by Kil *et al.* (2013) was found to be unreliable because of the significant presence of sky-region and a large portion of pixels being close to the value of A. The other alternative proposed by Hsieh *et al.* (2014) resulted in only marginal improvements for the haze around the objects while the output appeared more saturated.

Fig 3.6 shows an example where the tracks detection algorithm presented in Chapter 2 fails to detect tracks on the hazy image but is able to detect the tracks after removing haze from the input image. The top-left image is the input image. The failed output of the tracks detection algorithm is the top-right image. Bottom-left image is the output of haze removal algorithm and the tracks detected are shown in the bottom-right image.

Figure 3.6: Examples tracks detection in hazy image after haze removal

Figure 3.7: Examples of haze removal

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 Experiments

The algorithm designed to detect the train tracks was tested on over 4000 frames of size 596 x 336 pixels and 30000 frames of size 1280 x 720 pixels with good success. These frames were obtained from videos found on the internet. The most demanding step of the algorithm is the dark channel estimation which has a time complexity of $O(N^2)$ and complexity of every other process is of same or lower order. So the time complexity of the system is $O(N^2)$. The average running time was found to be 35 ms for each frame. This means the algorithm can be used to process a video of frame rate upto 29 FPS with one frame delay. When coupled with haze removal algorithm, the average running time was increased to 100 ms. So a video of frame rate upto 10 FPS can be processed with a one frame delay. With a GPU based implementation as explained in OpenCV, a further speed up of 10 - 20 times can be achieved.

The system was developed in C++ using OpenCV libraries and other C/C++ libraries.The experiments were performed on a system with 8 GB RAM and intel i7-3537U processor with no processes running in the background.

4.2 Results

Figs 4.1 and 4.2 show the input and the output of the algorithm over various frames of two different videos. The left column shows the input image and the right column shows the output for the corresponding input image. The scene in both these examples is day-time and not hazy with straight or approximately straight tracks. The detection of the tracks is done accurately in both these cases. Fig 4.3 shows the input and output of the haze removal algorithm along with tracks detection on various frames of a hazy video. The input hazy images are shown in the left column and their corresponding output in the right column.

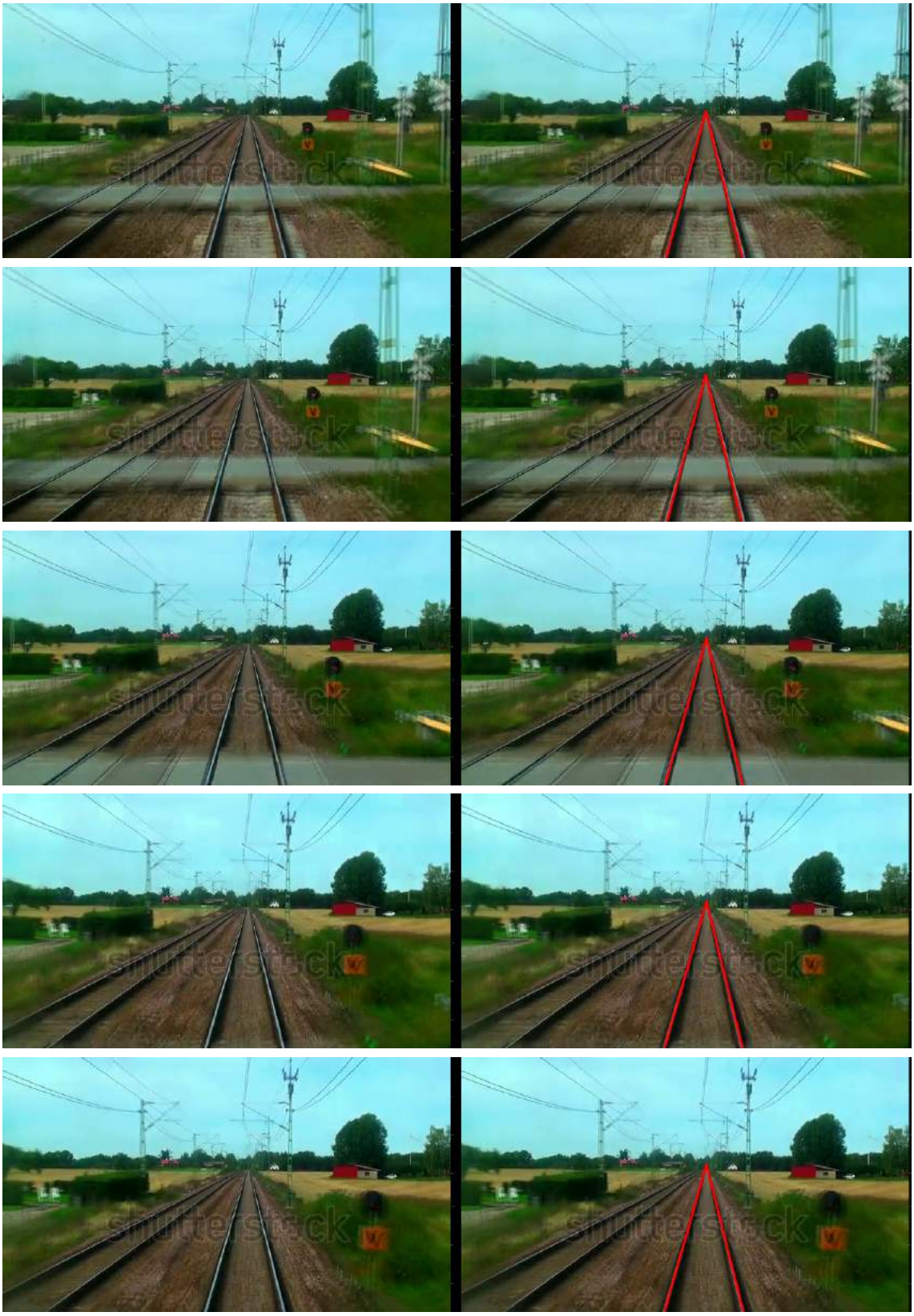

Figure 4.1: Examples of rail tracks detection.

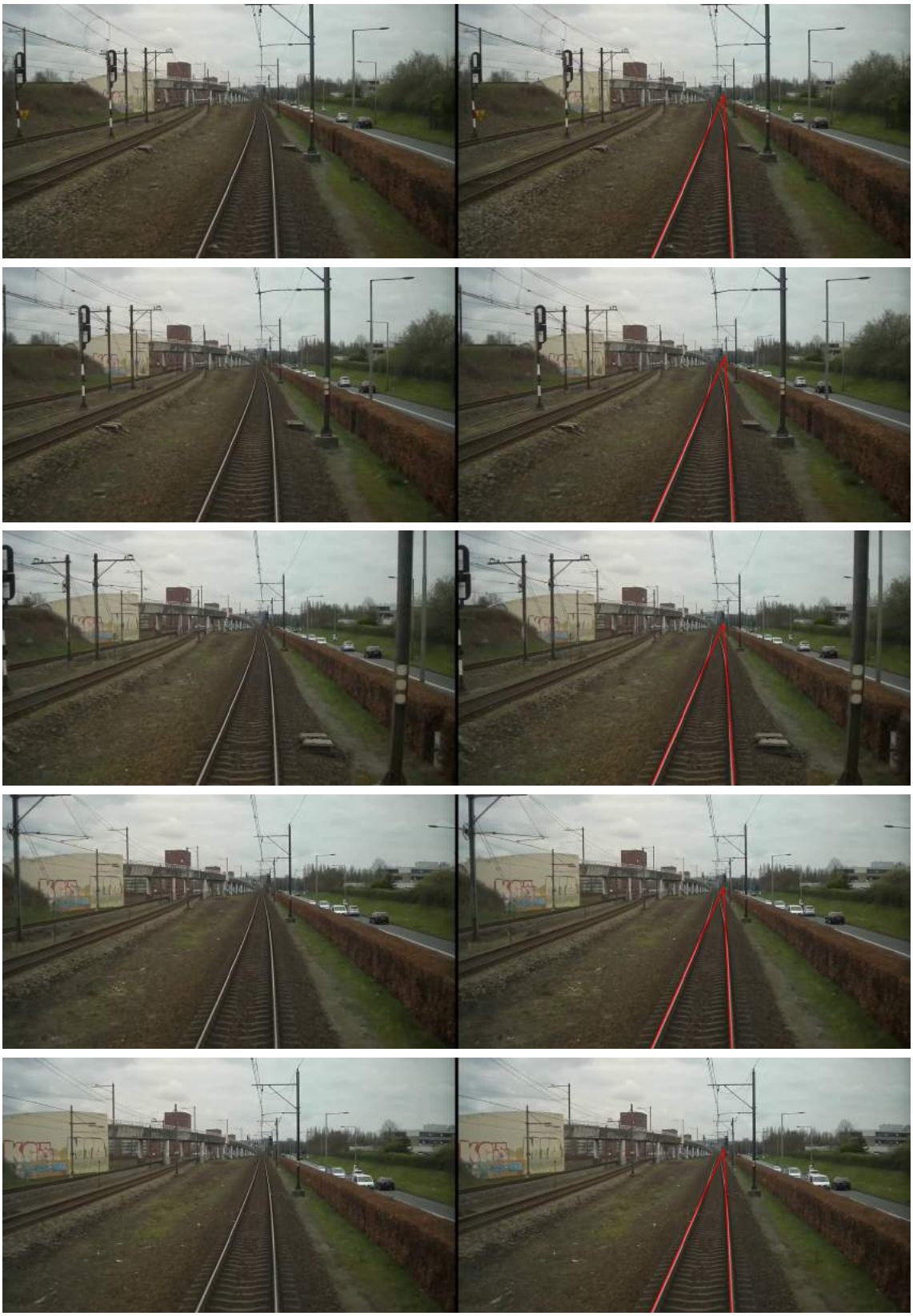

Figure 4.2: Examples of rail tracks detection.

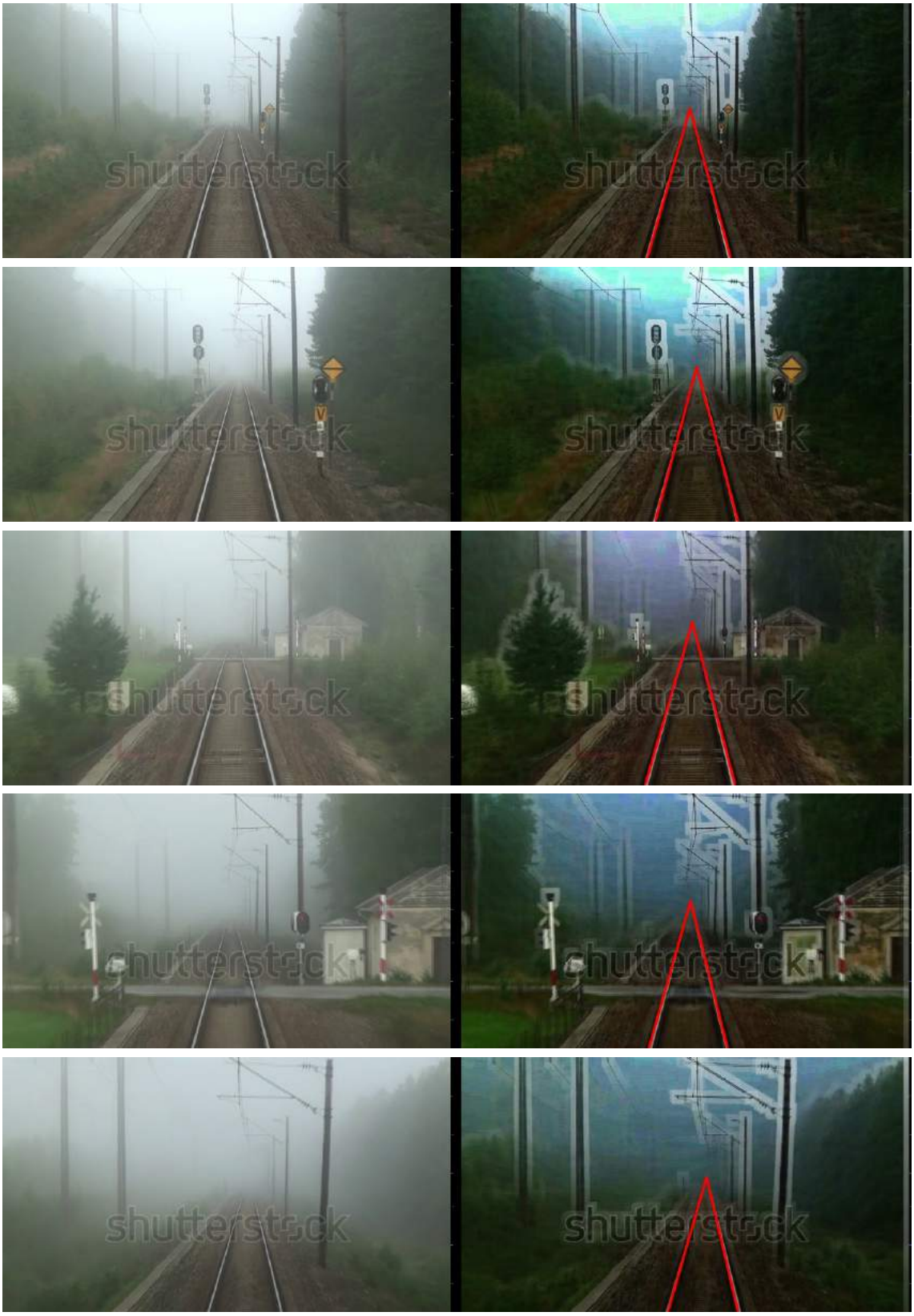

Figure 4.3: Examples of tracks detection combined with haze removal.

CHAPTER 5

CONCLUSIONS

In this thesis, we have proposed a system to detect the train tracks given a video sequence from a camera mounted on the train. With the help of Sobel operator and Hough Transform, we find all the lines in the image. Then, tracks are detected by checking the possible pairs of lines for some conditions as discussed in section 2.3.3. The detected tracks can be used as a ROI to look for obstacles by driver or an obstacle detection automation.

We have also developed a method to remove haze when the scene is hazy using the dark channel prior method. We designed an optimized algorithm to estimate the dark channel which is the most demanding operation of this method using the DEQUE data structure which improves the speed by 5 times and makes it possible to run the algorithm on real-time videos.

We have achieved reasonable processing time for the videos on the PC. With an embedded systems implementation, the system proposed can perform these tasks real-time on a live video sequence to assist the driver by detecting the tracks and removing haze when the scene is hazy. Our system does not detect the haze automatically and user input is needed to inform the system about presence of haze in the scene.

REFERENCES

1. **cplusplus.com** (). `http://www.cplusplus.com/reference/deque/deque/`.

2. **Duda, R. O.** and **P. E. Hart** (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, **15**(1), 11–15.

3. **ERTMS** (). `http://www.ertms.net/?page_id=42/`.

4. **ETCS** (). `http://uic.org/ETCS`.

5. **He, K.**, **J. Sun**, and **X. Tang** (2011). Single image haze removal using dark channel prior. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **33**(12), 2341–2353.

6. **Hsieh, C.-H.**, **Y.-S. Lin**, and **C.-H. Chang**, Haze removal without transmission map refinement based on dual dark channels. *In Machine Learning and Cybernetics (ICMLC), 2014 International Conference on*, volume 2. IEEE, 2014.

7. **Kaleli, F.** and **Y. S. Akgul**, Vision-based railroad track extraction using dynamic programming. *In Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*. IEEE, 2009.

8. **Kil, T. H.**, **S. H. Lee**, and **N. I. Cho**, Single image dehazing based on reliability map of dark channel prior. *In Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, 2013.

9. **Maire, F.** and **A. Bigdeli**, Obstacle-free range determination for rail track maintenance vehicles. *In Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*. IEEE, 2010.

10. **Narasimhan, S. G.** and **S. K. Nayar**, Interactive (de) weathering of an image using physical models. *In IEEE Workshop on Color and Photometric Methods in Computer Vision*, volume 6. France, 2003.

11. **OpenCV** (). `http://opencv.org/platforms/cuda.html`.

12. **Ross, R.**, Vision-based track estimation and turnout detection using recursive estimation. *In Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, 2010.

13. **Schechner, Y. Y.**, **S. G. Narasimhan**, and **S. K. Nayar**, Instant dehazing of images using polarization. *In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1. IEEE, 2001.

14. **Sheikh, Y. A.**, **Y. Zhai**, **K. Shafique**, and **M. A. Shah**, Visual monitoring of railroad grade crossing. *In Defense and Security*. International Society for Optics and Photonics, 2004.

15. **Shwartz, S.**, **E. Namer**, and **Y. Y. Schechner**, Blind haze separation. *In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2. IEEE, 2006.

16. **Silar, Z.** and **M. Dobrovolny**, The obstacle detection on the railway crossing based on optical flow and clustering. *In Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*. IEEE, 2013.

17. **Tan, R. T.**, Visibility in bad weather from a single image. *In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008.

18. **Wang, Y.**, **E. K. Teoh**, and **D. Shen** (2004). Lane detection and tracking using b-snake. *Image and Vision computing*, **22**(4), 269–280.

19. **Xue, J.**, **J. Cheng**, **L. Wang**, and **X. Gao**, Visual monitoring-based railway grade crossing surveillance system. *In Image and Signal Processing, 2008. CISP'08. Congress on*, volume 2. IEEE, 2008.

20. **Yu, B.** and **A. K. Jain**, Lane boundary detection using a multiresolution hough transform. *In Image Processing, 1997. Proceedings., International Conference on*, volume 2. IEEE, 1997.