Moving Object Detection and Collision Time Approximation

M.Tech. Dissertation Report

A dissertation submitted in partial fulfilment of the requirement for the M. Tech. (Computer Science) degree of the Indian Statistical Institute

By
Somnath Panja
CS0813

Under the esteemed guidance of
Dr. C. A. Murthy
Professor
Machine Intelligence Unit
Indian Statistical Institute, Kolkata

INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road
Kolkata 700108

Acknowledgments

With great pleasure and sense of obligation I express my heartfelt gratitude to my guide Dr. C. A. Murthy (Machine Intelligence Unit). I am highly indebted to him for his invaluable guidance and his readiness for anytime help. His persisting encouragement, everlasting patience and excellent expertise in subject have benefited to an extent, which is beyond expression.

I also want to thank to Dr. Utpal Garain (CVPR, ISI Kolkata) for his motivation, encouragement.

Without wasting this valuable chance, I want to thank my classmates, friends and my family members for their consistent support.

And lastly I want to thank my parents for their consistent flow of energy by which we are able to do anything in this world.

Somnath Panja

October, 2010

Abstract

The development of robots in modern times has been limited to safe and supervised areas. As technology has advanced, new goals have been set for robot behaviour. Previous investigations and models on Street-crossing Robots have demonstrated that robots need to be able to track multiple vehicles before they decide to cross a street. This behaviour has been accomplished through the preparation of an algorithm after calculating and analyzing diverse obstacles that could put the robot in danger.

In this initial work we focused on proving this algorithm in one-way street with no more that one lane in this direction, for it is more dangerous and complicated for the robot to attempt to cross a street with multiple lanes.

This paper describes progress toward a street-crossing system for an outdoor mobile robot. The system can detect and track vehicles and calculate the time when it is safe to cross the street.

Contents

Chapter 1
Introduction

In the past, delivery robots have been confined to a single building. Imagine a robot that could travel autonomously across campus to pick up a book at the library, and then bring it back to you in your office. There are systems that are able to drive safely on sidewalks (for example, [1]); avoiding obstacles while staying on a path, but none are able to cross the street. Developing a robot that can cross a street autonomously would allow for delivery robots that could cover a number of buildings, robotic wheelchairs that could drive their users safely across intersections, and sentry robots that could patrol multiple buildings.

The detection of moving object and collision time calculation is very important for the navigation of mobile robot. Here, by “collision time” we mean the remaining time from current time to the time of intersection of a moving vehicle (or object) and the straight line perpendicular to the road passing through the robot position in the road plane.

Many existing algorithms [2]-[4] segment each video frame to determine the objects; this action can be computationally expensive, and it is not necessary if the goal is to determine the moving objects. Alternatively, Doherty and Dyck proposed an algorithm [5] that derives the objects based on the motion between frames.

Mori [6-9] has explored algorithms for tracking cars in the context of a robotic travel aid for the blind, but he does not address the street crossing

problem explicitly. His vehicle detection and tracking results are quite impressive, but his tracking results depict a single oncoming vehicle. In fact, his algorithm explicitly expects to detect at most two vehicles in a frame.

Mori's algorithm needs to detect road boundaries in its initial setup. The algorithm uses dynamic tracking windows overlying a street lane at known distances from the robot. His distance, width, velocity, and collision time estimates are based on a known world distance to a pixel coordinate in the image plane. We believe that this fixed camera constraint is not appropriate for a robot trying to make its way across the street.

A street-crossing mobile robot must be able to detect and track all moving and stationary vehicles in the visible scene in real time, as the robot itself moves through the world. The system must be able to determine the "collision time" of all tracked vehicles. Being able to determine collision time implies knowing distance and tracking distance over time.

As a safety precaution, human street crossers often make eye contact with drivers. A person's pose and gaze indicate a desire to cross. A person may become impatient or frustrated with a busy street. Some people assume drivers will stop if they walk out in front of their cars.

A robot can't make eye contact and must signal its intention to cross in some nonhuman fashion. The street-crossing robot we are developing is infinitely patient, and will be a safe and conservative street crosser. The robot will only attempt to cross the street when the minimum collision

time of all tracked vehicles is greater than the robot's crossing time by some margin of safety.

1.1 Road-map of the report

In this initial work, we focused on crossing the street in a one-way road with one lane in one direction and there is no traffic signal. (See Fig. 1 for a diagram of the robot's camera view. Fig. 2 shows an image from the robot's left-facing camera.)

In chapter 3, we have discussed the some related works that have been done.

In chapter 4, we have discussed the whole algorithm. In section 4.1 we have listed out our assumptions towards success of the algorithm. In section 4.2.1 we have discussed how the robot will determine length of the path to cross the road. Then in section 4.2.2 we have discussed about the moving object detection, tracking algorithm, velocity and acceleration calculation of each moving vehicle or moving object (those are needed to track for the robot to cross the road safely.). In that section we have also discussed about how to calculate collision time.

Chapter 5 contains the Summary and Future scope of this work. And finally the appendix contains some of the matlab-code used for the experiment purposes.

Chapter 2
Motivation

From the beginning of robotics, robots have been in constant evolution. This evolution in the construction of robots is a direct consequence of the necessity to adapt to new and more complex tasks. It is clear that research and new work with robots is becoming more interesting and necessary in technological and social activities. This investigation attempts to develop a better tracking system capable of allowing a robot to detect vehicles, obstacles, pedestrians, motions, collision time etc. when the robot tries to cross a street.

According to these past investigations, new robots being built have more efficient and complete equipment that makes the task of crossing safely more possible. The efficiency of a tracking system will also depend on other factors including luminosity of the environment, speed of vehicles, creation of shadows, noises, etc. These could pose development problems.

Chapter 3
Related Work

There had not been much design development on the robot vision system towards street crossing robot because it was quite new. Problem of moving a robot through unknown environment has attracted much attention over past two decades. There have been some developments on unmanned system and related technologies like unmanned aircraft towards the new dream of building of smart robots.

Systems that use computer vision to track vehicles in real time generally come under two major headings: automated driving systems [10–16] and systems that monitor and analyze traffic scenes [17–25]. Unfortunately, most methods from these domains are unsuitable or inappropriate for the street crossing problem.

Some traffic analysis systems use a homography or projective transform between image and world coordinates [26, 27, 28]. This transform is useful for gathering traffic statistics such as vehicle count, speed and congestion. It requires off-line camera initialization or road markers to support on-line camera calibration. This is a reasonable approach in the context of fixed camera traffic surveillance systems, but invalid for a mobile robot.

Some vehicle tracking systems exploit symmetries that arise from the viewing angle. For example, the rear view of a car from an autonomous vehicle is generally symmetric. These systems use models based on

vehicle gray level or color symmetries and vehicle edge symmetries or aspect ratios [10, 11, 13]. From the vantage point of a robot waiting to cross the street, the symmetry of an approaching or passing car is ever-changing. Another type of symmetry is based on the relatively uniform palette of gray values on the road surface [29, 30]. Here is another assumption that is generally true of highways and generally false in a typical street-crossing scene. The road model needs to stay current with changing lighting conditions. Some researcher also uses Mori scan and history tracking [32].

Almost all of the algorithms presented in the literature include some kind of lane masking or road boundary detection to limit the search space for vehicles. In the context of traffic analysis systems, highway lanes are generally straight and relatively easy to detect given the overhead camera view. In the context of autonomous highway vehicles, lane detection is fundamental to following the curvature of the road. Again, the camera view is conducive to detecting lane boundaries reliably as long as the road does not have sharp bends. The street crossing robot has a sidelong view of the roadway at street level. Perfect detection of the road boundaries requires some cooperation from the real world. The street must be generally flat, straight, and free of traffic at the time of detection; for this application, explicit road boundary detection is an unrealistic expectation. Implicitly, road boundaries are being detected as the left and right limits of motion in the scene. As a practical matter, it is unrealistic to assume that vehicles remain wholly within their lane at all times.

Chapter 4
Algorithm

We made some assumptions towards success of this algorithm. So, before going into detail of the algorithm we'll discuss those assumptions.

4.1 Assumptions

Following are the assumptions that we made towards success of the algorithm.

1. The street should be one-way traffic i.e. the vehicles are moving in one direction either from left to right or from right to left.
2. The street including the position of the robot is a plane.
3. Only one camera is used as the robot's eye at a height of the robot.
4. Initially the robot is positioned at one side of the road at a safe distance let's say 5 feet away from the road from where it will start journey. The robot will cross the road along perpendicular line to the road from the robot's initial position. The camera is focusing along that line i.e. the robot is facing towards the line.
5. The robot can see the path to cross the road. The robot will not cross the road if it can not see the path. Let's call it as robot's path.

4.2 Procedure

The overall procedure is divided into two stages viz. 1. offline processing 2. final processing

4.2.1 Offline processing

There are two steps which can be performed offline before moving vehicle detection. One of them is the calculation of length of the path to travel to cross the road and another is scale factor calculation.

4.2.1.1 Calculation of length of the path to travel

In this section, we'll describe how we have calculated the length of the path to travel to reach the robot to other side of the road. Following are the steps:

1. **Capturing background images**

Let all conditions stated in section 4.1 are satisfied and hence robot can see its path. Let us consider the figure 4.1. Let 'O' is the initial position of the robot. Rectangle ABCD is the road. Vehicles are moving in the direction B to A (i.e. in the direction BA). X axis is parallel to the road side. Y axis is across the road and perpendicular to the X axis. Y axis is along the robot's path to travel to cross the road. XY plane is the same plane as road plane. Z axis is perpendicular XY plane through the

robot's initial position 'O' as shown in the figure 4.1. The point H be the position of the camera at a height of the robot, let's say 5 feet. Suppose, initially camera is facing along Y axis at the point 'H'. In that position camera makes 90 degree angle with the Z axis. Now we'll rotate the camera in ZY plane to see along $\overrightarrow{HO}$ and for each 10 degree decrement of the angle with negative Z axis we'll take a background image. So we'll take 10 such background images during rotation from 90 degree to 0 degree in ZY plane. Let's number these frames from 1 to 10. Background images are taken when no vehicle is present on the road. Then in the following section we'll decide for which frame the distance between the column axis and the road edge is minimum. Then we'll find out in which angle that image was taken. Now if we know the focal length of the camera we can find out the length of the path to travel to cross the road.

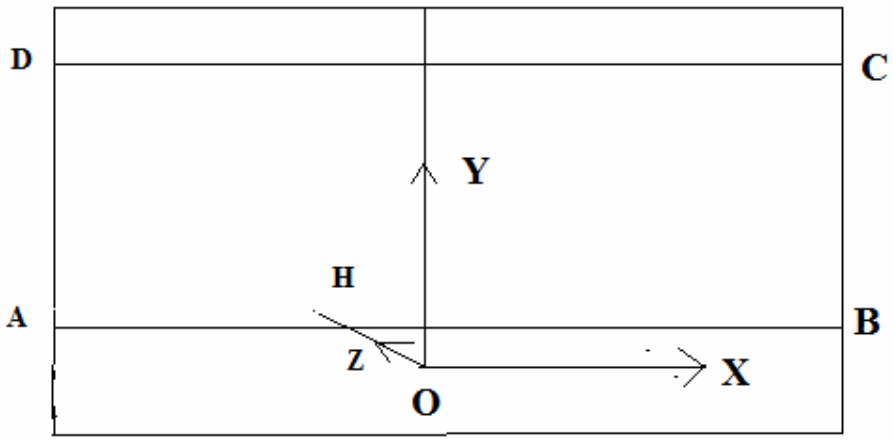


figure 4.1

1. Length of the path calculation

Step 2.1

This step we'll execute for each frame from frame 1 to frame 10.

A. Edge extraction

We'll detect edges in the current frame. You can use any edge detection algorithm with a threshold. After edge extraction we'll get a binary image. I used Canny edge detection algorithm to detect edges. It gives a good result.

B. Road edge detection

We assume that rows and columns in an image are evaluated as shown in figure 4.2. We'll perform following computation on edge detected binary image. For the first frame we'll consider that the initial row for the following computation is equal to floor of the one third of total number of rows present in the image. Then we'll search each row, say I, from the initial row to 1st row of the image until we get a row that is a possible road's side edge on the opposite side to the robot's standing position. A row will be a possible road edge if the following conditioned are satisfied.

1. Consider a horizontal strip of 21 of rows with centred at row I, i.e. consider a block of rows [I-10 I+10], provided (I-10)>0. If (I-10) <=0 consider the rows [1 I+10]. Then count the total

number of white pixels in these rows. To be a possible road side edge the count must be greater than 1000.

2. If the 1st condition is satisfied then consider a horizontal block [I+10 I+50]. Count the total number of white pixels in those rows. The row, I, to be a road side edge this count must be less than 500 since this block is on the road so there should not be much edges visible in this portion.
3. If 1st and 2nd conditions are satisfied and (I-50) <=0 then we say that row I is the road side edge which is in other side of the road in which the robot is standing. Else consider a horizontal strip of rows, [I-50 I-10], and count the total number of white pixels on those rows. Since this block is out side of the road, the row I to be a road's side edge the count must be greater than 2500.

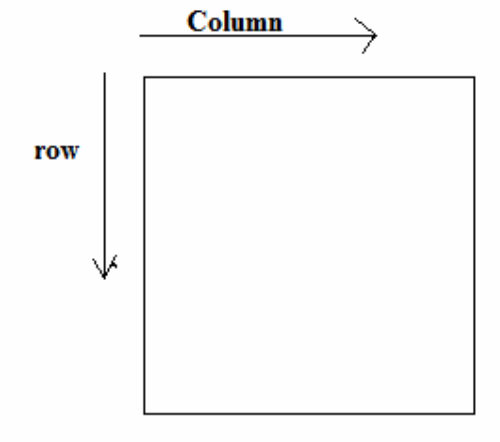


figure 4.2

Following are the steps to perform to evaluate the length of the path:

1. For the first frame take initial frame is equal to floor of the one third of total number of rows present in the image.
2. For each row, I, from the initial row to 1 repeat the following steps
    i. check if condition 1, 2 and 3 are satisfied for a possible road edge. If conditions are satisfied mark the row as road edge for that frame. Assign initial row for the next frame is this row. Go to the next frame. Else if this is the first frame or its previous frame has not detected any road edge then assign initial row for the next frame same as this frame. Else if the previous has detected the road edge then the required frame is the current frame and go to step 2.2.

Step 2.2

Now we have got the required frame. Find the camera angle when this image was captured. Let the angle is Θ. Let camera focal length is f and the camera height is h. Let vertical dimension of the camera image format is U.

Then vertical view angle, $\alpha = 2 \arctan (U/2f)$

Path length= h tan ((Θ+ (α/2)) pi/180)

Now we have calculated the path length to travel to cross the road.

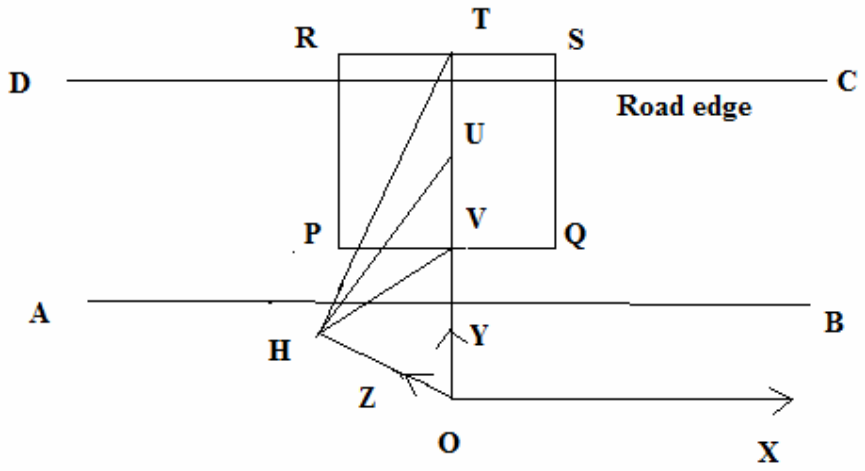


figure 4.3

The above shows the final situation of the above algorithm. Let BC and DA are road edges. Vehicles are moving along negative direction of X axis. OH is the robot. Camera is positioned at point H. We want to detect the CD road edge. Let PQSR are the final image or required frame that we have got after execution of Road edge detection algorithm described in the above step. The line segment OT is the robot path. So we are calculating the length of the line segment OT. The point V is the middle point of the line segment PQ. Here,

Camera angle of the final frame with HO (i.e. along negative Z axis) = $\Theta = \angle OHU$

Camera vertical view angle = $\alpha = \angle VHT$. Height of the robot = $\overline{OH}$ =h

4.2.1.2 Scale factor calculation

Now we rotate the camera towards right direction (with respect to figure 4.3) parallel to XY plane to make an angle of 45 degree with the positive X axis. In this position camera makes 90 degree with the Z axis and 45 degree with the Y axis. Now if we capture image frame it will look like the figure 4.4.

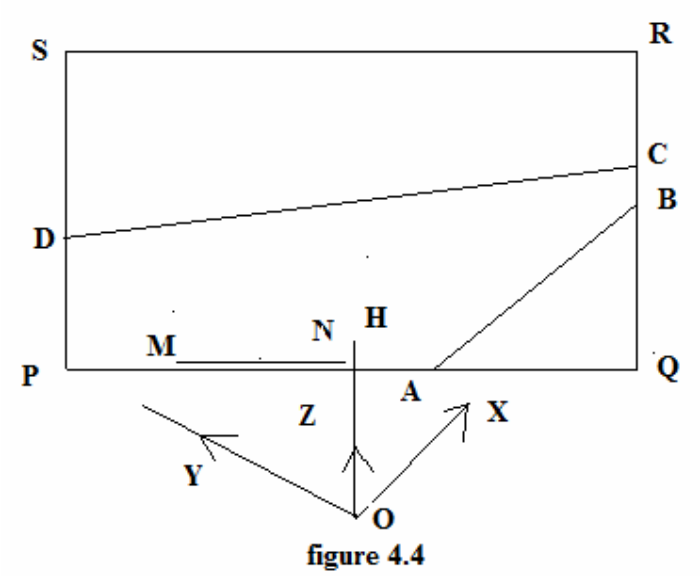


figure 4.4

In the above figure the line segment AB and CD describes road edges. XYZ is the reference system. OH is the robot. H is the camera position making an angle 45 degree with the X axis and 90 degrees with the Z axis. Now we'll find scale factor near PQ on the left side from the middle of the line segment PQ. To find scale factor we simply measure one distance, say MN, in the object space with a measurement tape. Let this

distance be d. Let its corresponding distance in the image plane as shown in the figure 4.4 be di. Then,

$$\text{Scale factor} = \frac{di}{d}$$

The use of scale factor we'll see when we'll calculate collision time of the moving vehicle and the robot.

4.2.2 Final processing

Now at the same position of the camera as described in section 4.2.1.2 (see figure 4.4 for reference) the robot capture video frames and perform this processing for each video frame. A number of processing has been performed to find the time at which the robot is safe to cross the street. First we have to detect moving vehicles (or pedestrians). Then we'll track those moving vehicles. Then we'll calculate the velocity, acceleration and collision time for each moving vehicle. These procedures are done for each frame of the video until the robot can detect a safe time to cross the road.

4.2.2.1 Moving vehicle detection

To detect vehicles, a number of processing steps are taken. Movement in the image is found by double differencing successive frames of the scene. The double differenced image is then filtered using a 5x5 median filter to remove noise and background motion. Edges are extracted using

an edge detection algorithm. The highest edge points are candidates for a line marking the top of a car.

4.2.2.1.1 Image double differencing

The main tracking feature of the algorithm is motion. Image differencing is a useful technique used for extracting scene motion. There are two general methods: reference frame differencing and inter frame differencing. Both methods are sensitive to background and camera motion, but the reference frame (or background subtraction) method is unsuitable for a street-crossing robot. A reference frame must be grabbed when the scene is stationary and free of objects that could move while the tracking process is active. Alternatively, a reference frame could be computed by observing the constant parts of the image over time. Next, the reference frame needs to be updated on-line to stay current with lighting changes. Finally, whenever the robot moves, the reference frame becomes invalid. It is highly unlikely that the world will cooperate whenever the algorithm needs a clean reference frame.

We have used a variation of simple image subtraction called “double differencing”. The double difference method takes three consecutive frames, performs two subtractions against the middle frame, then ANDs the subtraction results to obtain a single motion image.

4.2.2.1.2 Noise filtering

A 5x5 median filter is used to remove camera noise and tiny flutters due to background motion. Most motion due to wind on trees and telephone wires are ignored by the assumption that vehicles stay in contact with the road plane.

4.2.2.1.3 Edge extraction

An edge detector is used to delineate the motion edges. Here we don't want to find weak edges since we don't want to consider edges corresponding to background motion. So a suitable threshold should be used to get edge extracted binary image. Sobel is used frequently in the literature because it is efficient and provides generally good results. In the context of vehicle tracking, we expect to find mostly horizontal and vertical edges due to the typically rectangular contour of vehicles.

4.2.2.1.4 Bounding box computation

In this step we'll detect each maximal bounding box on the edge image satisfying certain conditions. Each box will correspond to one moving vehicle or a possible track collision. The conditions to be satisfied to be a bounding box are below

I. It is a rectangular region of the edge detected image.
II. The number of white pixels (assuming white pixels are corresponds to edge pixels) in the first row of the bounding box will be greater than 7.
III. There are no consecutive five rows in that box having number of white pixels in each row are zero.

IV. The number of white pixels in the 1st column of the bounding box is greater than five.

V. There is no consecutive five columns in that box having number of white pixels in each column less than or equal to six.

The above conditions must be satisfied to be bounding box. We say a bounding box as maximal bounding box if no superset of that bounding box is a bounding box.

Our aim is to find all such bounding boxes in the current edge detected image. Following are the steps that I followed to compute such maximal bounding box.

Procedure (Edge Image)

1. Count the number of white pixels in each row and save in an array. Traverse that array to find all such horizontal strips satisfying conditions II and III for bounding box.
2. Now for each such horizontal strip do the following steps
   2.1 Count the number of white pixels in each column and save it in an array.
   2.2 Traverse that array to find where to satisfy condition IV for the bounding box. Now starting from this column we'll find where the condition V has been failed. Then we'll call the function Procedure with input image as sub region of the current horizontal strip starting from the column where condition IV was satisfied to the column where the condition V was failed. If the condition V does not fail up to the size of the input Edge Image we check whether the difference of row

and column dimensions between the sub region of the horizontal strip where the conditions IV and V have been satisfied and the input Edge Image is less than 5. If it is then we call this is a maximal bounding box. Otherwise we'll call the function Procedure with input as the sub region.

4.2.2.1.5 Tracking Algorithm

Once the object areas are determined in each frame, the tracking algorithm is needed to trace the objects from frame to frame. Here object areas are the maximal bounding boxes computed in the previous section. In this section, we will present a rule-based algorithm using the information of the object trajectories, sizes, grayscale distribution, and textures. Variables based on the information are first computed, and then the tracking results are decided based on variable values. I have got idea from [31].

The variables for object trajectories are the object position coordinates. To decide the object position, we define the centroid of an object (c_x, c_y) as

$$c_x = (\sum_{(i,j)\in \mathbf{O}} p_{i,j} \cdot i)/(\sum_{(i,j)\in \mathbf{O}} p_{i,j}),$$

$$c_y = (\sum_{(i,j)\in \mathbf{O}} p_{i,j} \cdot j)/(\sum_{(i,j)\in \mathbf{O}} p_{i,j});$$

where O is the set of coordinates of an object area and $p_{i,j}$ is the value of the edge image at position (i, j). Each object is then corresponding to a point. Furthermore, we assume that the object trajectories are close to straight lines in a few adjacent frames and the object acceleration rate is a constant in these frames.

The next comparison we'll perform from frame number four onwards. From the previous frame information record we'll get the predicted position of this moving object. By comparing the predicted positions and real positions, it is possible to achieve trajectory-based tracking.

Assuming the frame rate is adequate, the sizes of the objects should not change dramatically between adjacent frames. The dispersion variable is used for tracking the objects based on size. The dispersion of an object, $disp$, is defined as

$$disp = (\sum_{(i,j)\in \mathbf{O}} \sqrt{(i-c_x)^2 + (j-c_y)^2} \cdot p_{i,j}) / (\sum_{(i,j)\in \mathbf{O}} p_{i,j});$$

where (c_x, c_y) is the object centroid, while O and $p_{i,j}$ have the same definition as in equations that defined the centoid. When the dispersions are computed in each frame, we can track the objects by comparing them.

The grayscale distribution of an object usually does not change too much, given that the lighting condition stays relatively constant between consecutive frames. In other words, the span of grayscale values for the same object is similar from frame to frame. The variables that we use,

based on grayscale distribution, are the mean of the whole grayscale range gr_m, the mean of the 10% pixels of largest grayscale value gr_h, and the mean of the 10% pixels of smallest grayscale value gr_l. These three variables will indicate the grayscale span of the object. Hence, grayscale based tracking can be achieved by matching the variables of the objects in different frames.

The last variable is based on object texture. The surfaces of the objects are usually not homogeneous. If we consider the grayscale variations on the object, they are usually different from object to object. These differences are reflected in the wavelet transform coefficients. A variable denoted by tx, that can roughly indicate the texture property of an object is the mean of the 10% of the pixels with the largest values in the constructed "edge" image. Generally speaking, large values indicate more textures on the object.

Because there are extremes that violate the assumptions that we have made, it is obvious that none of the above four sets of variables will be accurately tracking the objects all the time. Therefore, we cannot just depend on one set of the variables and need to integrate four sets together.

In the current frame, each object is associated with four sets of variables. Each existing track in the previous frame will also produce four sets of variables. Therefore, the tracking problem becomes finding the best matches between the objects and the existing tracks. A natural way to do this is to compute the differences between the variable values and then threshold the differences. If there are m objects in the current frame and n existing tracks in the previous frame, there will be a total of $m \times n$ sets of differences to evaluate. Considering the situations of new tracks,

ceased tracks and track collisions, we have to make sure that the variables of an object are similar to those of an existing track when we extend the track to that object. Since we have four sets of variables, the first rule that we use is that at least three sets of differences must be less than the threshold. Otherwise, we will not consider that object as a possible extension for the track.

After we evaluate $m \times n$ sets of we'll have a matrix of size $m \times n$. The elements of the matrix show how many sets of variable differences between the specific object and the certain track is less than the threshold. As discussed in the previous paragraph, an element greater than 2 corresponds to a possible track extension. However, it is obvious that there usually will not be one and only one eligible element in each row and each column of the matrix. Therefore, we develop the following strategy.

First, we start from the simple cases. If an element is the only one eligible in its row and its column, there is no ambiguity. We simply extend the corresponding track to the corresponding object and simplify the matrix by eliminating the row and the column or by putting a value -1 along the row and along the column that the element is in. Second, suppose $e_{i,j}$ is the only eligible element in row i, if all other eligible elements in column j are not the only element in their corresponding rows, we will extend track j to object i, and eliminate the $i\text{th}$ row and $j\text{th}$ column to simplify the matrix. A similar procedure is performed when $e_{i,j}$ is the only eligible element in column j and all other eligible elements in row i are not the only one in their corresponding columns. After using the above two rules repeatedly, the eligible elements left in the matrix correspond to complicated situations that cannot be solved

using simple thresholding. We then adopt a weighted sum as the cost function, which is described by the equation

$$\begin{aligned} dif &= w_{tr}(|c_x^f - c_x^t| + |c_y^f - c_y^t|) + w_{disp}(|disp^f - disp^t|) \\ &\quad + w_{gr}(|gr_l^f - gr_l^t| + |gr_m^f - gr_m^t| + |gr_h^f - gr_h^t|) \\ &\quad + w_{tx}(|tx^f - tx^t|), \end{aligned}$$

where dif is the cost function, $w_{tr}, w_{disp}, w_{gr}, w_{tx}$ are the weights, and superscripts f and t whether the variable is computed from the current frame or the track. For each eligible element in the matrix, a weighted sum is computed. Then for each column, the row producing the smallest dif value is selected, which is equivalent to finding the best matching object of an existing track.

After the process described above, there may be tracks and objects left with no matches. These situations often correspond to new tracks, ceased tracks, and track collision. We first consider the possibility of track collision by examining the variable values. If the mean values of the predicted positions of several existing tracks are “close” to one of the objects’ (compatible with the dispersion values), we then evaluate the dispersion values. If the object dispersion value is larger than the largest dispersion value of the tracks and smaller than the summation of the dispersion values of all those tracks, we will mark the object and the tracks as possible track collision. The rest of unmatched objects and tracks are then labeled as new tracks and ceased tracks.

4.2.2.1.5 Velocity and acceleration calculation

Now we'll calculate the velocity and acceleration of each moving vehicle (as recognized by bounding boxes) in the image frame in terms of pixel coordinates if the current frame number is greater than or equal to two. We'll compute the velocity from the second frame onwards and we'll save the current velocity to be available in the next frame for further computation. To calculate current velocity we'll first compute instantaneous velocity. By instantaneous velocity we mean the velocity that is obtained by considering the current position of the vehicle in the current frame and its corresponding position in the previous frame.

Instantaneous velocity along X direction = ($c_x^f - c_x^t$) . frame rate pixels/sec

Instantaneous velocity along Y direction = ($c_y^f - c_y^t$) . frame rate pixels/sec

If the velocity of the vehicle in the previous frame is negative then

velocity along X direction in the current frame = average of the instantaneous velocity in X direction in the current frame and velocity of this vehicle in X direction in the previous frame

and

velocity along Y direction in the current frame = average of the instantaneous velocity in Y direction in the current frame and velocity of this vehicle in Y direction in the previous frame.

Otherwise keep instantaneous velocity as the current velocity and save if for further computation in the current frame as well as for the next frame.

A similar idea has been used to calculate the acceleration of each moving vehicle in the current frame if the current frame number is greater than or equal to three. First we calculate the instantaneous acceleration depending on the velocity of the current and previous frame. Then if the previous acceleration along x or y direction is non zero then averaging the instantaneous acceleration and the acceleration of the corresponding moving vehicle in the previous frame in x and y direction separately we'll get the acceleration of the moving vehicle in the current frame in X and Y direction else keep the acceleration same as instantaneous acceleration.

Then we'll predict the position of each moving vehicle in the next frame. Here, we assume that the object trajectories are close to straight lines in a few adjacent frames and the object acceleration rate is a constant in these frames. The object location coordinates in the next frame for each moving vehicle in the current frame is predicted using the equation

$$S = vt + \frac{1}{2}at^2,$$

where υ is the initial speed and α is the acceleration rate.

4.2.2.1.5 Collision time calculation

As I have stated earlier, by collision time I mean the remaining time from the current time to the time at which the front position of the vehicle touches the robot path.

Consider the figure 4.5. Here, the XYZ coordinate system is the object space reference frame and the xy coordinate system with the point S as origin is the image reference frame. E is the centroid of a moving vehicle at the current frame and F was its centroid in the previous frame. The camera position is positioned at the same position as described in section

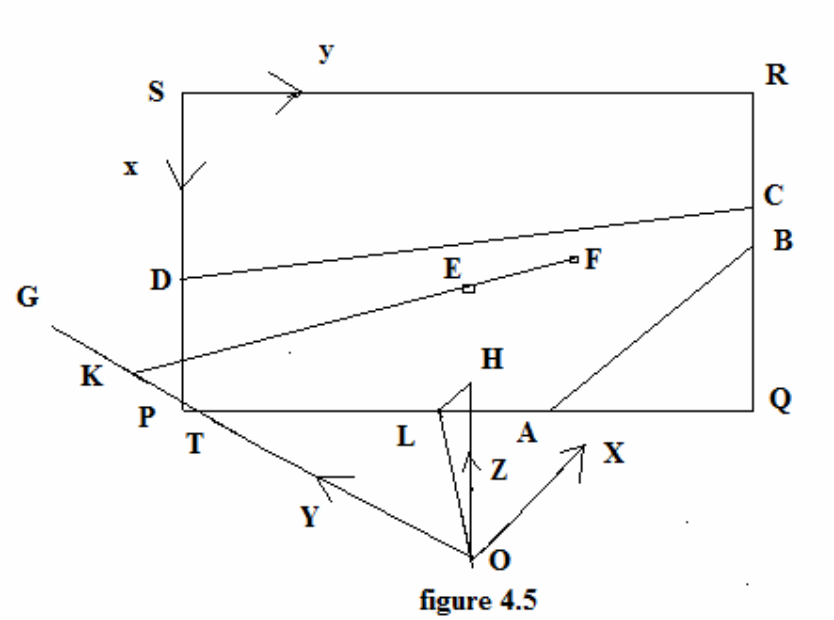


figure 4.5

4.2.1.2. PQRS is the image frame. L is the middle point of the line segment PQ. The robot will cross the road along Y axis. The collision time is the time that the vehicle (shown in the figure 4.5) takes to reach at the point K. If we know the vertical dimension of the camera image format and the camera focal (lens focal length) length then we can find the vertical view angle, α of the camera as described in the section 4.2.1.1. Then we can get the angle $\angle OHL$ = 90- (α/2). Now the triangle OHL is a right angle triangle and we know the length of the line segment OH = height of the robot and the angle $\angle LOH$ is 90 degree. So we can get the length of the OL in object reference XYZ. Then we'll find the length of the line segment OL in image coordinate system xy by multiplying its distance in the object plane (XYZ coordinates system)

with scale factor. Then we'll get the coordinates of the point T with respect to xy coordinate system. The line OK makes a 45 degree angle with the positive direction of X axis. So we get the equation of the line OK with respect to xy coordinate system. Now we get the equation of the line EF using centroid of the moving object in the current frame and its corresponding position in the previous frame. So we can find the coordinate of the point K. Here, we consider the point K to be correctly detected if the x coordinate of K lies in the interval [100 1600] and the y coordinate of the point K lies in [-1000 700]. These values depend on the dimension of the image. I used a camera with image row dimension is 720 and column dimension is 1280. If the coordinates of K are not in this range we will not consider that point for further computation instead we'll consider the corresponding point which we had in the previous frame for the corresponding vehicle. If we detect the point K correctly then compute the effective point as given below

x coordinate of the effective point = (.2) . x coordinate of the currently computed point + (.8) . x coordinate of the corresponding point for the same vehicle in the previous frame

and

y coordinate of the effective point = (.2) . y coordinate of the currently computed point + (.8) . y coordinate of the corresponding point for the same vehicle in the previous frame

Since in reality road is not a plane we are imposing 80% weight age to the history. Then we save this point for further computation in the current frame and the next frame.

Then find the distance of the effective point from the centroid of the vehicle in the current frame. We'll subtract the length of the vehicle which half of the column dimensions of the bounding box from this distance. Since we are interested in when the robot touches the robot path.

Now we'll perform next computation considering this effective distance and calculate the time required to reach at that position using the equation

$$S = vt + \frac{1}{2}at^2,$$

where υ is the initial speed and α is the acceleration rate. Take the positive root of the solution as the collision time.

4.2.2.1.5 Robot start time to cross the road

Let the robot maintain a constant speed during the time to cross the road and it is known. We know the path length to travel to cross the road. So we know how much time the robot will take to cross the road. If the velocity of a moving vehicle in positive y direction is greater than 20 pixels per frame we'll ignore that frame since this vehicle is moving in left to right direction relative to the figure 4.5. If the collision time of each moving vehicle present in the current frame as calculated from the previous section is greater than time required for robot to cross the road plus a safety time of 2 seconds then we'll say this is a safe time for the robot to journey to cross the road.

.

Chapter 5

Results

Result of the procedure stated in section 4.2.1.1(Calculation of length of the path to travel)

Original image

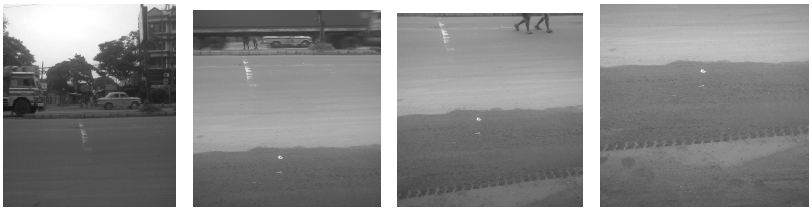
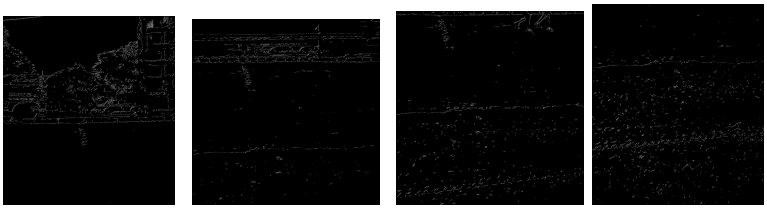


Image1 Image2 Image3 Image4

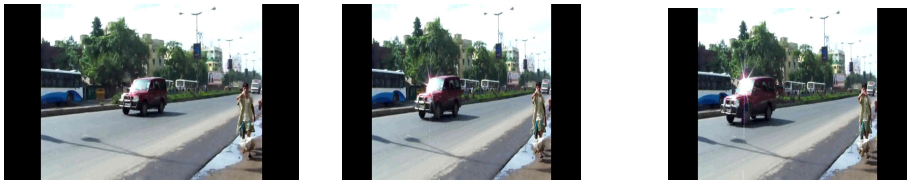
Corresponding edge detected images:



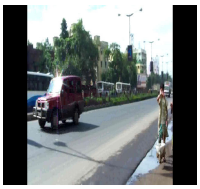
Edge_im1 Edge_im2 Edge_im3 Edge_im4

Img1 is captured perpendicular to the z axis. Other frames are also captured as described in the section 4.2.1.1.The remaining sex images are not given in the picture. The final result of the procedure is Image4.

Results of the procedure stated in section 4.2.2



1st frame of a video 5th frame of a video 8th frame of a video



10th frame of a video

Results of calculation of collision time for the vehicle shown in the image are given below

Frame number	Collision time calculated by the algorithm	Collision time in reality
4th frame	0.66 second	0.54 sec
5th frame	0.53 second	.51 second
6th frame	0.54 sec	0.48 second
7th frame	0.52 sec	0.46 second
8th frame	0.43 second	0.36 second
10th frame	0.29 second	0.30 second

This shows the algorithm works fine.

Chapter 6

Summary and Future work

The current algorithm is able to identify vehicles in one lane. Multiple vehicles can be found travelling in that lane. In this algorithm no vehicle can pass through the robot path. During crossing we are not making any decision we have only detected the starting time of the robot.

Extensions to the current system could include multilane and intersection crossings. For these extensions, other scene features could be used as clues to find safe crossing locations and safe crossing times. For example, the robot could decide to follow a pedestrian across the street. Traffic light changes could also be used to help the system reason about safe crossing times.

Appendices

Some codes

Bounding box computation:

bounding_box1.m

```
function k=bounding_box1(BW1)
    global record
    global component
    [row11 column11]=size(BW1);
    k=0;
    col_arr11=zeros(row11,1);
    %'1.............................'
    %a_row11=row11
    %a_column11=column11
    tmp_arr=zeros(1,4);
    tmp1=0;
    tmp2=0;
    tmp3=0;
    tmp4=0;
    tmp_count=0;
    tmp_flag=0;
    for i11=1 : row11
       col_arr11(i11,1)=sum(BW1(i11,:));
       %'2.......................'
       %col_arr11(i11)=col_arr11(i11)/255;
    end
    count11=0;
    row_mat=zeros(row11,2);
    i12=1;
    %'3.............................'
    while( i12<= row11)
        %'4.........................'
        if(col_arr11(i12,1)>7)
            row12=i12;
            %row_mat(1,1)=row12;
            %flag11=0;
            i13=i12;
            count11=count11+1;
            row_mat(count11,1)=row12;
            %'5.........................'
            %b_i12=i12
            %b_count11=count11
            while (i13<=row11)
                %'6.....................'
                if(((i13+5)<=row11)&&(col_arr11(i13+1,1)==0)&&
(col_arr11(i13+2,1)==0)&&(col_arr11(i13+3,1)==0)&&
(col_arr11(i13+4,1)==0)&& (col_arr11(i13+5,1)==0))
                  % flag11=1;
                   row13=i13;
                   row_mat(count11,2)=row13;
                   i12=i13+5;
                   %'7.................'
                   break;
                else
                   row13=i13;
```

```
                    row_mat(count11,2)=row13;
                    i13=i13+1;
                    i12=i13;
                    %'8.....................'
                end
                %'9.............................'
            end
            %'10.....................'
        else
            i12=i12+1;
            %'11.........................'
        end
        %'12................'
    end
    %'13............'
    row_arr11=zeros(count11,column11);
    %c_count11=count11
    for i14=1 : count11
        for j14=1: column11

row_arr11(i14,j14)=sum(BW1(row_mat(i14,1):row_mat(i14,2),j14));
            %'14.................'
        end
        count12=0;
        col_mat=zeros(column11,2);
        j15=1;
        while( j15<= column11)
            if(row_arr11(i14,j15)>5)
                col12=j15;
                %row_mat(1,1)=row12;
                %flag11=0;
                j13=j15;
                count12=count12+1;
                col_mat(count12,1)=col12;
                while (j13<=column11)

if(((j13+5)<=column11)&&(row_arr11(i14,j13+1)<=6)&&
(row_arr11(i14,j13+2)<=6)&&(row_arr11(i14,j13+3)<=6)&&
(row_arr11(i14,j13+4)<=6)&& (row_arr11(i14,j13+5)<=6))
                        % flag11=1;
                        col13=j13;
                        col_mat(count12,2)=col13;
                        if (((row11-(row_mat(i14,2)-
row_mat(i14,1)+1))<5)&&((column11-(col_mat(count12,2)-
col_mat(count12,1)+1))<5))
                            component=component+1;
                            %'15................'
                            %d_record_component_1=record(component,1)
                            %d_record_component_2=record(component,2)
                            %d_record_component_3=record(component,3)
                            %d_record_component_4=record(component,4)

record(component,1)=record(component,1)+row_mat(i14,1);

record(component,2)=record(component,2)+row_mat(i14,2);

record(component,3)=record(component,3)+col_mat(count12,1);

record(component,4)=record(component,4)+col_mat(count12,2);
                            k=1;
                            %'16................'
```

```
                            %d_k=k
                            %d_record_component_1=record(component,1)
                            %d_record_component_2=record(component,2)
                            %d_record_component_3=record(component,3)
                            %d_record_component_4=record(component,4)
                            %k
                            return ;
                        else
                            %size(BW1)
                            %row_mat(i14,1)
                            %row_mat(i14,2)
                            %col_mat(count12,1)
                            %col_mat(count12,2)
                            %'17............'
                            %if(component>0)

%e_record_component_1=record(component+1,1)

%e_record_component_2=record(component+1,2)

%e_record_component_3=record(component+1,3)

%e_record_component_4=record(component+1,4)
                            %end
                            if(tmp_count==0)
                                tmp_arr(1,:)=record(component+1,:);

                            end
                            tmp_count=tmp_count+1;
                            tmp1=tmp_arr(1,1);
                            tmp2=tmp_arr(1,2);
                            tmp3=tmp_arr(1,3);
                            tmp4=tmp_arr(1,4);

if((tmp_count~=1)&&((record(component+1,1)>0)||(record(component+1,2)
>0)||(record(component+1,3)>0)||(record(component+1,4)>0)))
                                record(component+1,1)=0;
                                record(component+1,2)=0;
                                record(component+1,3)=0;
                                record(component+1,4)=0;
                            end
                            if(tmp_count~=1)

record(component+1,1)=record(component+1,1)+tmp_arr(1,1)+row_mat(i14,
1)-1;

record(component+1,2)=record(component+1,2)+tmp_arr(1,2)+row_mat(i14,
1)-1;

record(component+1,3)=record(component+1,3)+tmp_arr(1,3)+col_mat(coun
t12,1)-1;

record(component+1,4)=record(component+1,4)+tmp_arr(1,4)+col_mat(coun
t12,1)-1;
                            else

record(component+1,1)=record(component+1,1)+row_mat(i14,1)-1;

record(component+1,2)=record(component+1,2)+row_mat(i14,1)-1;

record(component+1,3)=record(component+1,3)+col_mat(count12,1)-1;
```

```
record(component+1,4)=record(component+1,4)+col_mat(count12,1)-1;
                            end
                            %tmp_arr(1,:)=record(component+1,:);

%e_record_component_1=record(component+1,1)

%e_record_component_2=record(component+1,2)

%e_record_component_3=record(component+1,3)

%e_record_component_4=record(component+1,4)

BW2=BW1(row_mat(i14,1):row_mat(i14,2),col_mat(count12,1):col_mat(coun
t12,2));
                            p=1;
                            %p
                            k=bounding_box1(BW2);
                            %k
                           % if (k==3)

if((k==3)&&((record(component+1,1)>0)||(record(component+1,2)>0)||(re
cord(component+1,3)||(record(component+1,4)>0))))
                                %'18.........'
                                %if(component>0)

%f_record_component_1=record(component+1,1)

%f_record_component_2=record(component+1,2)

%f_record_component_3=record(component+1,3)

%f_record_component_4=record(component+1,4)
                                %end
                                if(tmp_count~=1)

record(component+1,1)=record(component+1,1)-tmp1-row_mat(i14,1)+1;

record(component+1,2)=record(component+1,2)-tmp2-row_mat(i14,1)+1;

record(component+1,3)=record(component+1,3)-tmp3-
col_mat(count12,1)+1;

record(component+1,4)=record(component+1,4)-tmp4-
col_mat(count12,1)+1;
                                else

record(component+1,1)=record(component+1,1)-row_mat(i14,1)+1;

record(component+1,2)=record(component+1,2)-row_mat(i14,1)+1;

record(component+1,3)=record(component+1,3)-col_mat(count12,1)+1;

record(component+1,4)=record(component+1,4)-col_mat(count12,1)+1;
                                end


%f_record_component_1=record(component+1,1)

%f_record_component_2=record(component+1,2)
```

```
%f_record_component_3=record(component+1,3)

%f_record_component_4=record(component+1,4)
                            end
                            j15=j13+5;
                            break;
                        end

                    else
                        col13=j13;
                        col_mat(count12,2)=col13;
                        if (j13== column11)
                            if (((row11-(row_mat(i14,2)-
row_mat(i14,1)+1))<5)&&((column11-(col_mat(count12,2)-
col_mat(count12,1)+1))<5))
                                component=component+1;
                                %'19.............'
                            %d_record_component_1=record(component,1)
                            %d_record_component_2=record(component,2)
                            %d_record_component_3=record(component,3)
                            %d_record_component_4=record(component,4)

record(component,1)=record(component,1)+row_mat(i14,1);

record(component,2)=record(component,2)+row_mat(i14,2);

record(component,3)=record(component,3)+col_mat(count12,1);

record(component,4)=record(component,4)+col_mat(count12,2);
                                k=2;
                                %k
                                %'20.................'

                            %d_record_component_1=record(component,1)
                            %d_record_component_2=record(component,2)
                            %d_record_component_3=record(component,3)
                            %d_record_component_4=record(component,4)
                                return ;
                            else
                                %'21.................'
                             %if(component>0)

%d_record_component_1=record(component+1,1)

%d_record_component_2=record(component+1,2)

%d_record_component_3=record(component+1,3)

%d_record_component_4=record(component+1,4)
                             %end
                             if(tmp_count==0)
                                tmp_arr(1,:)=record(component+1,:);

                            end
                            tmp_count=tmp_count+1;
                            tmp1=tmp_arr(1,1);
                            tmp2=tmp_arr(1,2);
                            tmp3=tmp_arr(1,3);
                            tmp4=tmp_arr(1,4);
```

```
if((tmp_count~=1)&&((record(component+1,1)>0)||(record(component+1,2)
>0)||(record(component+1,3)>0)||(record(component+1,4)>0)))
                                record(component+1,1)=0;
                                record(component+1,2)=0;
                                record(component+1,3)=0;
                                record(component+1,4)=0;
                            end
                            if(tmp_count~=1)

record(component+1,1)=record(component+1,1)+tmp_arr(1,1)+row_mat(i14,
1)-1;

record(component+1,2)=record(component+1,2)+tmp_arr(1,2)+row_mat(i14,
1)-1;

record(component+1,3)=record(component+1,3)+tmp_arr(1,3)+col_mat(coun
t12,1)-1;

record(component+1,4)=record(component+1,4)+tmp_arr(1,4)+col_mat(coun
t12,1)-1;
                            else

record(component+1,1)=record(component+1,1)+row_mat(i14,1)-1;

record(component+1,2)=record(component+1,2)+row_mat(i14,1)-1;

record(component+1,3)=record(component+1,3)+col_mat(count12,1)-1;

record(component+1,4)=record(component+1,4)+col_mat(count12,1)-1;
                            end
                                p=2;
                                %p
                                %'22.................'

%d_record_component_1=record(component+1,1)

%d_record_component_2=record(component+1,2)

%d_record_component_3=record(component+1,3)

%d_record_component_4=record(component+1,4)

BW3=BW1(row_mat(i14,1):row_mat(i14,2),col_mat(count12,1):col_mat(coun
t12,2));
                                k=bounding_box1(BW3);
                              %  if(k==3)

if((k==3)&&((record(component+1,1)>0)||(record(component+1,2)>0)||(re
cord(component+1,3)||(record(component+1,4)>0))))
                                   %'23.............'
                                   %if(component>0)

%d_record_component_1=record(component+1,1)

%d_record_component_2=record(component+1,2)

%d_record_component_3=record(component+1,3)

%d_record_component_4=record(component+1,4)
                                   %end
```

```
                                if(tmp_count~=1)

record(component+1,1)=record(component+1,1)-tmp_arr(1,1)-
row_mat(i14,1)+1;

record(component+1,2)=record(component+1,2)-tmp_arr(1,2)-
row_mat(i14,1)+1;

record(component+1,3)=record(component+1,3)-tmp_arr(1,3)-
col_mat(count12,1)+1;

record(component+1,4)=record(component+1,4)-tmp_arr(1,4)-
col_mat(count12,1)+1;
                                    %'24.................'
                                else

record(component+1,1)=record(component+1,1)-row_mat(i14,1)+1;

record(component+1,2)=record(component+1,2)-row_mat(i14,1)+1;

record(component+1,3)=record(component+1,3)-col_mat(count12,1)+1;

record(component+1,4)=record(component+1,4)-col_mat(count12,1)+1;
                                end


%d_record_component_1=record(component+1,1)

%d_record_component_2=record(component+1,2)

%d_record_component_3=record(component+1,3)

%d_record_component_4=record(component+1,4)
                             end


                             end

                         end
                         j13=j13+1;
                         j15=j13;
                     end
                 end
             else
                 j15=j15+1;
             end
         end
        %col_arr11(i11)=sum(BW(i11,:));
        %col_arr11(i11)=col_arr11(i11)/255;
    end

    if(k==0)
        k=3;
        %'25.................'
        %k
    end
end
```

**********end**********

Main program:

Main.m

```
global record
global component
image=strvcat('m1.jpg
','m2.jpg','m3.jpg','m4.jpg','m5.jpg','m6.jpg','m7.jpg','m8.jpg','m9.
jpg','m10.jpg');
o_image=strvcat('im_m1.jpg
','im_m2.jpg','im_m3.jpg','im_m4.jpg','im_m5.jpg','im_m6.jpg','im_m7.
jpg','im_m8.jpg','im_m9.jpg','im_m10.jpg');
canny_image=strvcat('canny1_m1.jpg
','canny1_m2.jpg','canny1_m3.jpg','canny1_m4.jpg','canny1_m5.jpg','ca
nny1_m6.jpg','canny1_m7.jpg','canny1_m8.jpg','canny1_m9.jpg','canny1_
m10.jpg');
readerobj =
mmreader('C:\Users\SOMNATH\Desktop\sparrow\obj_detection\dissertation
\100_5042.avi');

vidFrames = read(readerobj,[830 842]);
%numFrames=get(readerobj, 'numberOfFrames');
%disp(numFrames) ;
no_frames=10;
comp_mat=zeros(no_frames,1);
info_mat=zeros(2,15,4,4);
dfx_arr=zeros(2,15,1);
rec=zeros(2,15,4);
poly=zeros(1,3);
length=15 ;
scale_factor=51.95;
length_in_pixel=length*scale_factor;
row_max=720;
column_max=1280;
robot_cross_time=3;
frame_rate=30;

for i1=1: no_frames
    imwrite(vidFrames(:,:,:,i1),o_image(i1,:),'jpg');
    prev_im=rgb2gray(vidFrames(:,:,:,i1));
    %imshow(prev_im)

    next_im=rgb2gray(vidFrames(:,:,:,i1+1));
    %imshow(next_im)
    next_next_im=rgb2gray(vidFrames(:,:,:,i1+2));
    diff_im1 = imabsdiff(next_im,prev_im);
    diff_im2 = imabsdiff(next_next_im,next_im);
    diff_im=diff_im1 & diff_im2;
    %diff_im(1,1);
    %diff_im=double(diff_im);
    diff_im=immultiply(diff_im1 , diff_im);
    %diff_im(200:300,:)
    %pause
    %for i2=1: 2
        diff_im = medfilt2(diff_im, [5 5]);
    %end
    imwrite(diff_im,image(i1,:),'jpg');
    %BW = edge(diff_im,'canny');
    %BW = edge(diff_im,'log');
    %BW = edge(diff_im,'prewitt');
```

```
    BW = edge(diff_im,'sobel');
    %imwrite(BW,'canny_edge_053.jpg');
    imwrite(BW,canny_image(i1,:),'jpg');
    if(i1==5)
        BW140 = edge(diff_im,'sobel');
        imwrite(diff_im,'image_140.jpg');
        imwrite(BW140,'canny_edge140_053.jpg');
    end

    BW1=BW;
    [row col]=size(BW);
    %BW1=BW1(296:458,482:704);
    %global record
    record=zeros(20,4);
    component=0;
    k=bounding_box1(BW1);%, component,record);
    BW11=BW;
    BW12=next_im;
    comp_mat(i1,1)=component;
    rec(2,1:component,:)=record(1:component,:);
    maximal_bounding_boxes=record(1:component,:)
    %i1

%*********************************************************************
**
    for i21=1: component
       %%sum=0;
       sum3=0;
       for var_counter=record(i21,3): record(i21,4)

sum3=sum3+sum(BW(record(i21,1):record(i21,2),var_counter));
       end
       sum1=0;
       sum2=0;
       for i22=record(i21,1) : record(i21,2)
           sum1=sum1+(sum(BW(i22,record(i21,3) :
record(i21,4)))).*i22;

       end
       for i23=record(i21,3) : record(i21,4)
           sum2=sum2+(sum(BW(record(i21,1) :
record(i21,2),i23))).*i23;

       end
       cx=sum1./sum3;
       cy=sum2./sum3;
       value_i21=i21;
       info_mat(2,i21,1,1)=cx;
       info_mat(2,i21,1,2)=cy;
       %info_mat(2,i21,1,1)
       %info_mat(2,i21,1,2)
       mat=zeros(record(i21,2)-record(i21,1)+1,record(i21,4)-
record(i21,3)+1);
       for i24=1 : record(i21,2) - record(i21,1) +1
           for i25=1  : record(i21,4) - record(i21,3) +1
               mat(i24,i25)=sqrt((i24 +record(i21,1)-1 -cx)^2  +(i25
+ record(i21,3) -1 -cy)^2);
           end

       end
```

```
D=mat.*BW(record(i21,1):record(i21,2),record(i21,3):record(i21,4));
       disp=sum(D(1:record(i21,2)-record(i21,1)+1,1:record(i21,4)-
record(i21,3)+1));
       disp=sum(disp(1,:))./sum3;
       info_mat(2,i21,2,1)=disp;
       mat11=zeros(1,(record(i21,2)-record(i21,1)+1)*(record(i21,4)-
record(i21,3)+1));
       i27=record(i21,4)-record(i21,3)+1;
       for i26=1 : (record(i21,2)-record(i21,1)+1)
            mat11(1,(i26-1)*i27+1:i26*i27)=BW12(record(i21,1)+i26-
1,record(i21,3):record(i21,4));
       end
       %clear mat11;
       avg=mean(mat11);
       D21=sort(mat11);
       i28=(record(i21,2)-record(i21,1)+1)*(record(i21,4)-
record(i21,3)+1);
       i29=int32(i28/10);
       avg1=mean(D21(1,i28-i29:i28));
       avg2=mean(D21(1,1:i29));
       info_mat(2,i21,3,1)=avg;
       info_mat(2,i21,3,2)=avg1;
       info_mat(2,i21,3,3)=avg2;
       for i26=1 : (record(i21,2)-record(i21,1)+1)
            mat11(1,(i26-1)*i27+1:i26*i27)=BW11(record(i21,1)+i26-
1,record(i21,3):record(i21,4));
       end
       d22=sort(mat11);
       avg4=mean(d22(1,i28-i29:i28));
       info_mat(2,i21,4,1)=avg4;
    end
    if((i1>=2)&&(component>0))
        rel_mat=zeros(component,comp_mat(i1-1,1));
        for i30=1: component
            for i31=1: comp_mat(i1-1,1)
                count5=0;
                if(i1>=5)
                    displacement=sqrt((info_mat(2,i30,1,1)-
info_mat(1,i31,4,2))^2   +   (info_mat(2,i30,1,2)-
info_mat(1,i31,4,3))^2);
                    if(displacement-info_mat(2,i30,1,3) < 100)
                        count5=count5+1;
                    end
                end
                if(abs(info_mat(2,i30,2,1)-info_mat(1,i31,2,1))<100)
                    count5=count5+1;
                end
                if ((abs(info_mat(2,i30,3,1)-
info_mat(1,i31,3,1))<20)&&(abs(info_mat(2,i30,3,2)-
info_mat(1,i31,3,2))<20)&&(abs(info_mat(2,i30,3,3)-
info_mat(1,i31,3,3))<20))
                   count5=count5+1;
                end
                if (abs(info_mat(2,i30,4,1)-info_mat(1,i31,4,1))<10)
                    count5=count5+1;
                end
                rel_mat(i30,i31)=count5;
            end
        end
        previous_track=zeros(component,1);
```

```
        next_object=zeros(comp_mat(i1-1,1),1);
        flag36=0;
        while(flag36==0)
            flag30=0;
            for i32=1 : component
                count6=0;
                prob_col=-1;
                for i33=1 : comp_mat(i1-1,1)
                    if(rel_mat(i32,i33)>=3)
                        count6=count6+1;
                        prob_col=i33;
                    end
                end
                if(count6==1)
                   count7=0;
                   for i34=1: component
                       if(rel_mat(i34,prob_col)>=3)
                           count7=count7+1;
                       end
                   end
                   if (count7==1)
                       rel_mat(i32,1:comp_mat(i1-
1,1))=ones(1,comp_mat(i1-1,1)).*(-1);

rel_mat(1:component,prob_col)=ones(component,1).*(-1);
                       previous_track(i32,1)=prob_col;
                       next_object(prob_col,1)=i32;
                       flag30=1;
                   end
                   if (count7>1)
                      flag35=0;
                      for i35=1: component
                         if((i35~=i32)&& (rel_mat(i35,prob_col)>=3))
                             count8=0;
                             %for i36=1: comp_mat(i1-1,1)
                             %    if(rel_mat(i35,))
                             %end
                             arr30=find(rel_mat(i35,:)>2);
                             [row30 col30]=size(arr30);
                             if(col30==1)
                                 flag35=1;
                                 break;
                             end
                         end
                      end
                      if (flag35==0)
                            rel_mat(i32,1:comp_mat(i1-
1,1))=ones(1,comp_mat(i1-1,1)).*(-1);

rel_mat(1:component,prob_col)=ones(component,1).*(-1);
                           previous_track(i32,1)=prob_col;
                           next_object(prob_col,1)=i32;
                           flag30=1;
                      end
                   end
                end

            end
            for j32=1 : comp_mat(i1-1,1)
                count10=0;
                prob_row=-1;
```

```
                for j33=1 : component
                    if(rel_mat(j33,j32)>=3)
                        count10=count10+1;
                        prob_row=j33;
                    end
                end
                if(count10==1)
                   count11=0;
                   for j34=1: comp_mat(i1-1,1);
                       if(rel_mat(prob_row,j34)>=3)
                           count11=count11+1;
                       end
                   end
                   if (count11==1)
                       rel_mat(prob_row,1:comp_mat(i1-
1,1))=ones(1,comp_mat(i1-1,1)).*(-1);
                       rel_mat(1:component,j32)=ones(component,1).*(-
1);
                       previous_track(prob_row,1)=j32;
                       next_object(j32,1)=prob_row;
                       flag30=1;
                   end
                   if (count11>1)
                      flag45=0;
                      for j35=1: comp_mat(i1-1,1)
                         if((j35~=j32)&& (rel_mat(prob_row,j35)>=3))
                             count20=0;
                             %for i36=1: comp_mat(i1-1,1)
                             %    if(rel_mat(i35,))
                             %end
                             arr31=find(rel_mat(:,j35)>2);
                             [row31 col31]=size(arr31);
                             if(row31==1)
                                 flag45=1;
                                 break;
                             end
                         end
                      end
                      if (flag45==0)
                             rel_mat(prob_row,1:comp_mat(i1-
1,1))=ones(1,comp_mat(i1-1,1)).*(-1);

rel_mat(1:component,j32)=ones(component,1).*(-1);
                             previous_track(prob_row,1)=j32;
                             next_object(j32,1)=prob_row;
                             flag30=1;
                      end
                   end
                end

            end
            if(flag30==0)
               flag36=1;
            end
        end
        %diff_matrix=zeros(15,15);
        %row_count=0;
        %col_count=0;
        %row_arr10=0;
        [row_dif,col_dif,v_dif]=find(rel_mat>=3);
        size_dif=size(row_dif);
```

```
        dif_count=size_dif(1);
        %dif_count
        dif_count_2=size_dif(2);
        %row_dif(1)
        if((dif_count>0) && (dif_count_2>0))
            diff_matrix=zeros(dif_count,1);
            for i50=1 : dif_count
                row_dif(1)
                diff_matrix(i50,1)=(abs(info_mat(2,row_dif(i50),1,1)-
info_mat(1,col_dif(i50),1,1))+abs(info_mat(2,row_dif(i50),1,2)-
info_mat(1,col_dif(i50),1,2))+abs(info_mat(2,row_dif(i50),2,1)-
info_mat(1,col_dif(i50),2,1))+abs(info_mat(2,row_dif(i50),3,1)-
info_mat(1,col_dif(i50),3,1))+abs(info_mat(2,row_dif(i50),3,2)-
info_mat(1,col_dif(i50),3,2))+abs(info_mat(2,row_dif(i50),3,3)-
info_mat(1,col_dif(i50),3,3))+abs(info_mat(2,row_dif(i50),4,1)-
info_mat(1,col_dif(i50),4,1)))*(.25);
            end
            max_column=max(col_dif);
            for i51=1 : max_column
                index=find(col_dif==i51);
                size_index=size(index);
                size_count_index=size_index(1);
                if(size_count_index>0)
                    min_diff=diff_matrix(index(1),1);
                    min_index=1;
                    i52=1;
                    for i52=1: size_count_index
                        if(min_diff>diff_matrix(index(i52),1))
                            min_diff=diff_matrix(index(i52),1);
                            min_index=i52;
                        end
                    end

if((previous_track(row_dif(index(min_index)),1)==0) &&
(next_object(col_dif(index(min_index)),1)==0))

previous_track(row_dif(index(min_index)),1)=col_dif(index(min_index))
;

next_object(col_dif(index(min_index)),1)=row_dif(index(min_index));

rel_mat(row_dif(index(min_index)),1:comp_mat(i1-
1,1))=ones(1,comp_mat(i1-1,1)).*(-1);

rel_mat(1:component,col_dif(index(min_index)))=ones(component,1).*(-
1);
                    end
                end
            end
        end
        [row_dif1,col_dif1,v_dif1]=find(rel_mat>0);
        size_dif1=size(row_dif1);
        dif_count1=size_dif1(1);
        dif_count1_2=size_dif1(2);
        if((dif_count1>0)&&(i1>1)&&(dif_count1_2>0))
            for i60=1: dif_count1
                %a=1
                if(sqrt((info_mat(2,row_dif1(i60),1,1)-
info_mat(1,col_dif1(i60),4,2)^2+info_mat(2,row_dif1(i60),1,2)-
info_mat(1,col_dif1(i60),4,3)^2))<500)
```

```
                    if((previous_track(row_dif1(i60),1)==0) &&
(next_object(col_dif1(i60),1)==0))
                        previous_track(row_dif1(i60))=col_dif1(i60);
                        next_object(col_dif1(i60),1)=row_dif1(i60);
                        rel_mat(row_dif1(i60),1:comp_mat(i1-
1,1))=ones(1,comp_mat(i1-1,1)).*(-1);

rel_mat(1:component,col_dif1(i60))=ones(component,1).*(-1);
                    elseif(next_object(col_dif1(i60),1)==0)
                        next_object(col_dif1(i60),1)=row_dif1(i60);

rel_mat(1:component,col_dif1(i60))=ones(component,1).*(-1);
                    end
                end
            end
        end

        previous_track
        next_object

%*****************************************************************
        vehicle_count=0;
        flag_distance=0;

        for i21=1: component
            if(previous_track(i21,1)~=0)
            %vx=(info_mat(2,i21,1,1)-
info_mat(1,previous_track(i21,1),1,1))*frame_rate;
            %vy=(info_mat(2,i21,1,2)-
info_mat(1,previous_track(i21,1),1,2))*frame_rate;
            %cx_result=info_mat(2,i21,1,1)
            %cy_result=info_mat(2,i21,1,2)
            vx=(info_mat(2,i21,1,1)-
info_mat(1,previous_track(i21,1),1,1));
            vy=(info_mat(2,i21,1,2)-
info_mat(1,previous_track(i21,1),1,2));
            %info_mat(2,i21,1,1)
            %info_mat(1,previous_track(i21,1),1,1)
            %info_mat(2,i21,1,2)
            %info_mat(1,previous_track(i21,1),1,2)
            %vx
            %vy
            if(info_mat(1,previous_track(i21,1),1,4)<0)

vx=(.5)*vx+(.5)*info_mat(1,previous_track(i21,1),1,3);

vy=(.5)*vy+(.5)*info_mat(1,previous_track(i21,1),1,4);
            end
            %vx
            %vy
            info_mat(2,i21,1,3)=vx;
            info_mat(2,i21,1,4)=vy;
            poly(1,2)=vy;
            if(i1>=3)
                %fx=(info_mat(2,i21,1,3)-
info_mat(1,previous_track(i21,1),1,3))*frame_rate;
                %fy=(info_mat(2,i21,1,4)-
info_mat(1,previous_track(i21,1),1,4))*frame_rate;
                fx=(info_mat(2,i21,1,3)-
info_mat(1,previous_track(i21,1),1,3));
```

```
                fy=(info_mat(2,i21,1,4)-
info_mat(1,previous_track(i21,1),1,4));
                if(info_mat(1,previous_track(i21,1),2,3))

fx=(.5)*fx+(.5)*info_mat(1,previous_track(i21,1),2,2);

fy=(.5)*fy+(.5)*info_mat(1,previous_track(i21,1),2,3);
                end
                %fx
                %fy
                info_mat(2,i21,2,2)=fx;
                info_mat(2,i21,2,3)=fy;
                poly(1,1)=(.5)*fy;
                if(i1>=4)
                    %dfx=(info_mat(2,i21,2,2)-
info_mat(1,previous_track(i21,1),2,2))*frame_rate;
                    %dfy=(info_mat(2,i21,2,3)-
info_mat(1,previous_track(i21,1),2,3))*frame_rate;
                    dfx=(info_mat(2,i21,2,2)-
info_mat(1,previous_track(i21,1),2,2));
                    dfy=(info_mat(2,i21,2,3)-
info_mat(1,previous_track(i21,1),2,3));
                    if(info_mat(1,previous_track(i21,1),4,4))

dfx=(.5)*dfx+(.5)*dfx_arr(1,previous_track(i21,1),1);

dfy=(.5)*dfy+(.5)*info_mat(1,previous_track(i21,1),4,4);
                    end
                    %dfx
                    %dfy
                    dfx_arr(2,i21,1)=dfx;
                    info_mat(2,i21,4,4)=dfy;
                    %poly(1,1)=(1/6)*dfy;

%predicted_cx=info_mat(2,i21,1,1)+((1./6)*dfx*(1/30)^3+(.5)*fx*(1/30)
^2+vx*(1/30));

%predicted_cy=info_mat(2,i21,1,2)+((1./6)*dfy*(1/30)^3+(.5)*fy*(1/30)
^2+vy*(1/30));

%predicted_cx=info_mat(2,i21,1,1)+((1/6)*dfx+(.5)*fx+vx);

%predicted_cy=info_mat(2,i21,1,2)+((1/6)*dfy+(.5)*fy+vy);
                    predicted_cx=info_mat(2,i21,1,1)+((.5)*fx+vx);
                    predicted_cy=info_mat(2,i21,1,2)+((.5)*fy+vy);
                    info_mat(2,i21,4,2)=predicted_cx;
                    info_mat(2,i21,4,3)=predicted_cy;
                    %cx_1=info_mat(2,i21,1,1)
                    %cy_1=info_mat(2,i21,1,2)

predicted_cx_1=info_mat(1,previous_track(i21,1),4,2);

predicted_cy_1=info_mat(1,previous_track(i21,1),4,3);
                    x=((row_max-(column_max/2)+length_in_pixel
)*(info_mat(1,previous_track(i21,1),1,1)-info_mat(2,i21,1,1))-
info_mat(2,i21,1,1)*info_mat(1,previous_track(i21,1),1,2)+info_mat(1,
previous_track(i21,1),1,1)*info_mat(2,i21,1,2))/(info_mat(2,i21,1,2)-
info_mat(1,previous_track(i21,1),1,2)+info_mat(1,previous_track(i21,1
),1,1)-info_mat(2,i21,1,1));
                    y=x-(row_max-(column_max/2)+length_in_pixel );
                    %x
```

```
                    %y
                    if((y<700)&&(y>-1000) && (x > 100 ) && (x<1600))
                        if(info_mat(1,previous_track(i21,1),2,4)>0)

avg_x=(.2)*x+(.8)*info_mat(1,previous_track(i21,1),2,4);

avg_y=(.2)*y+(.8)*info_mat(1,previous_track(i21,1),3,4);
                            info_mat(2,i21,2,4)=avg_x;
                            info_mat(2,i21,3,4)=avg_y;
                        else
                            info_mat(2,i21,2,4)=x;
                            info_mat(2,i21,3,4)=y;
                        end
                    else

info_mat(2,i21,2,4)=info_mat(1,previous_track(i21,1),2,4);

info_mat(2,i21,3,4)=info_mat(1,previous_track(i21,1),3,4);
                    end
                    x_avg=info_mat(2,i21,2,4);
                    y_avg=info_mat(2,i21,3,4);
                    %x;
                    %y;

                    %intersection_count=0;

                    v=sqrt((vx)^2+(vy)^2);
                    poly(1,2)=v;
                    %v
                    f=sqrt((fx)^2+(fy)^2);
                    poly(1,1)=(.5)*f;
                    %f
                    df=sqrt((dfx^2)+(dfy^2));
                    %df
                    if(info_mat(2,i21,2,4)>0)

collision_distance=(sqrt((info_mat(2,i21,2,4)-
info_mat(2,i21,1,1))^2+(info_mat(2,i21,3,4)-info_mat(2,i21,1,2))^2)-
(info_mat(2,i21,1,2)-rec(2,i21,3)));
                        %collision_distance=abs(info_mat(2,i21,3,4)-
info_mat(2,i21,1,2))-(info_mat(2,i21,1,2)-rec(2,i21,3));
                        poly(1,3)=-collision_distance;
                        %avg_y1=info_mat(2,i21,3,4)
                        %cy1=info_mat(2,i21,1,2)
                        %c1=rec(2,i21,3)
                        %collision_distance
                        syms t;
                        %1sol=solve(df*(t^3)+3*f*(t^2)+6*v*t-
6*collision_distance);
                        sol=roots(poly);
                        [row_root col_root]=size(sol);
                        required_time=-10;
                        for i100=1: row_root
                            if((isreal(sol(i100)))&&(sol(i100)>0))
                                required_time=sol(i100);
                                break;
                            end
                        end
                        s=-10;
                        if(required_time~=-10)
                            s=double(required_time);
```

```
                            collision_time=s
                        end
                        if(vy>20)
                            vehicle_count=vehicle_count+1;
                        elseif((s>=0)&&
((double(s))>((robot_cross_time+2)*30)))
                            vehicle_count=vehicle_count+1;

                        end
                    else
                        flag_distance=1;
                    end
                end
            end
            end
        end
        if((vehicle_count==component)&&(flag_distance==0))
            'The frame and the time that that robot starts walking to
cross the road safely'
            imshow(vidFrames(:,:,:,i1));
            pause(10)
            vidFrames_final = read(readerobj,[720+i1+89 720+i1+91]);
            imshow(vidFrames(:,:,:,1));
            pause(10)
            imshow(vidFrames_final(:,:,:,2));
            break;
        end
    end
    if(component==0)
    'The frame and the time that that robot starts walking to cross
the road safely'
    imshow(vidFrames(:,:,:,i1));
    pause(10)
    vidFrames_final = read(readerobj,[720+i1+89 720+i1+91]);
    imshow(vidFrames(:,:,:,1));
    pause(10)
    imshow(vidFrames_final(:,:,:,2));
     break;
    end
    info_mat(1,:,:,:)=info_mat(2,:,:,:);
    rec(1,:,:)=rec(2,:,:);
    dfx_arr(1,:,:)=dfx_arr(2,:,:);
end
```

References:

[1] H. A. Yanco, “Shared user-computer control of a robotic wheelchair system, ” Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, September 2000.

[2] D. Wang, Unsupervised Video segmentation Based On Watersheds And Temporal Tracking, *IEEE Trans. Circuits Syst. Video Technol.*, 8(5):539-546, September 1998.

[3] G. L. Foresti, Object Recognition And Tracking For Remote Video Surveillance, *IEEE Trans. Circuits Syst. Video Technol.*, 9(7):1045-1062, October 1999.

[4] A. J. Lipton, H. Fujiyoshi, R. S. Patil, Moving Target Classification And Tracking From Real-time Video, *Applications of Computer Vision, 1998. WACV ’98. Proceedings., Fourth IEEE Workshop on*, pp. 8-14, 1998.

[5] Y. Wang, R.E. Van Dyck, J. F. Doherty, Tracking Moving Objects in Video Sequences, *Proc. Conference on Information Sciences and Systems*, Princeton, NJ, March 2000.

[6] N. M. Charkari, K. Ishii, and H. Mori, “Proper selection of sonar and visual sensors for vehicle detection and avoidance,” Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, Vol. 2 , 12-16 Sept. 1994, pp. 1110 –1117.

[7] N. M. Charkari and H. Mori, “A new approach for real time moving vehicle detection,” Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, 26-30 July 1993, pp. 273 – 278.

[8] H. Mori and N. M. Charkari, “Shadow and rhythm as sign patterns of

obstacle detection," International Symposium on Industrial Electronics, 1993, pp. 271--277.
[9] H. Mori, N. M. Charkari, T. Matsushita, "On-line vehicle and pedestrian detection based on sign pattern," IEEE Trans. on Industrial Electronics, Vol. 41, No. 4, pp. 384-391, Aug. 1994.
[10] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele, "Stereo vision based vehicle detection," IEEE Intelligent Vehicles Symposium, Detroit, MI, October 2000, pp. 39-44.
[11] M. Betke, E. Haritaoglu, and L. Davis, "Highway scene analysis in hard real-time," Intelligent Transportation Systems, IEEE, July 1997.
[12] F. Dellaert and C. Thorpe, "Robust car tracking using Kalman filtering and Bayesian templates," Proc. of the SPIE - Int. Soc. Opt. Eng., Vol. 3207, October 1997.
[13] U. Regensburger and V. Graefe, "Visual recognition of obstacles on roads," Proc. IROS '94, pp. 980–987, Munich, Germany, 1994.
[14] M. B. van Leeuwen and F. C. A. Groen, "A platform for robust real-time vehicle tracking using decomposed templates," Intelligent Autonomous Systems Group, Faculty of Science, University of Amsterdam, Amsterdam, The Netherlands, unpublished.
[15] M.B. van Leeuwen and F.C.A. Groen, "Vehicle detection with a mobile camera," Intelligent Autonomous Systems Group, Faculty of Science, University of Amsterdam, Amsterdam, The Netherlands, unpublished.
[16] L. Zhao and C. Thorpe. "Qualitative and quantitative car tracking from arange image sequence," Computer Vision and Pattern Recognition, pages 496-501, 1998.
[17] L. A. Alexandre and A. C. Campilho, "A 2D image motion detection method using a stationary camera," RECPAD98, 10th Portuguese Conference on Pattern Recognition, Lisbon, Portugal, 1998.

[18] E. Atkociunas and M. Kazimianec, “Aspects in traffic control system development,” Vilnius University, Faculty of Mathematics and Informatics, Jyvaskyla, 2002.
[19] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. “A real-time computer vision system for vehicle tracking and traffic surveillance,” Transportation Research: Part C, vol 6, no 4, 1998, pp 271-288.
[20] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, “A realtime computer vision system for measuring traffic parameters,” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1997.
[21] R. Cucchiara, M. Piccardi, A. Prati, and N. Scarabottolo, “Real-time detection of moving vehicles,” Proc International Conference on Image Analysis and Processing, Venice, Italy, pp. 618 – 623, September 1999.
[22] D.J. Dailey and L. Li, “Video image processing to create a speed sensor,” ITS Research Program, Final Research Report, College of Engineering, University of Washington, Seattle, Washington, March 1999.
[23] N. Ferrier, S. Rowe, and A. Blake, “Real-time traffic monitoring,” Proc. 2nd IEEE Workshop on Applications of Computer Vision, pp. 81–88, 1994.
[24] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, S. Russel, “Towards robust automatic traffic scene analysis in real-time,” Proc. Int'l Conf. Pattern Recognition, pp. 126–131, 1994.
[25] D. R. Magee, “Tracking multiple vehicles using foreground, background and motion models,” University of Leeds, School of Computer Studies, Research Report Series, (Submitted to European Conference on Computer Vision, May 2002), December 2001.

[26] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. “A real-time computer vision system for vehicle tracking and traffic surveillance,” Transportation Research: Part C, vol 6, no 4, 1998, pp 271-288.
[27] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, “A realtime computer vision system for measuring traffic parameters,” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1997.
[28] N. Ferrier, S. Rowe, and A. Blake, “Real-time traffic monitoring,” Proc. 2nd IEEE Workshop on Applications of Computer Vision, pp. 81–88,1994.
[29] M. Betke and H. Nguyen, “Highway scene analysis from a moving vehicle under reduced visibility conditions,” Proc. of the International Conference on Intelligent Vehicles, IEEE Industrial Electronics Society, Stuttgart, Germany pp. 131–136. Oct. 1998.
[30] C. Tzomakas and W. von Seelen, “Vehicle detection in traffic scenes using shadows,” Internal Report IRINI 98-06, Institut fur Neuroinformatik, Ruhr-Universitat Bochum, Germany, August 1998.
[31] Yiwei Wang and John F. Doherty, Robert E. Van Dyck “Moving Object Tracking in Video
[32] Michael Baker and Holly A. Yanco “A Vision-Based Tracking System for a Street-Crossing Robot” Submitted to ICRA-04.