UNIVERSITY OF MINNESOTA

CENTER FOR TRANSPORTATION STUDIES

ITS INSTITUTE

MANAGING SUBURBAN INTERSECTIONS THROUGH SENSING

Harini Veeraraghavan
Osama Masoud
Nikolaos P. Papanikolopoulos

Artificial Intelligence, Robotics, and Vision Laboratory
Department of Computer Science and Engineering
University of Minnesota

CTS 02-07

HUMAN-CENTERED TECHNOLOGY TO ENHANCE SAFETY AND MOBILITY

Technical Report Documentation Page

1. Report No.	2.	3. Recipient's Accession No.
CTS 02-07		

4. Title and Subtitle	5. Report Date
MANAGING SUBURBAN INTERSECTIONS THROUGH SENSING	December 2002
	6.

7. Author(s)	8. Performing Organization Report No.
Harini Veeraraghavan, Osama Masoud, Nikolaos P. Papanikolopoulos	

9. Performing Organization Name and Address	10. Project/Task/Work Unit No.
Artificial Intelligence, Robotics, and Vision Laboratory Department of Computer Science 55-240 EE/Csci Bldg. 200 Union Street SE Minneapolis, MN 55455	11. Contract (C) or Grant (G) No. (C) (G)

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
Intelligent Transportation Systems Institute Center for Transportation Studies University of Minnesota 200 Transportation and Safety Bldg. 511 Washington Ave. SE Minneapolis, MN 55455	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract (Limit: 200 words)

Increased urban sprawl and increased vehicular traffic have resulted in an increased number of traffic fatalities, the majority of which occur near intersections. According to the National Highway Safety Administration, one out of eight fatalities occurring at intersections is a pedestrian. An intelligent, real-time system capable of predicting situations leading to accidents or near misses will be very useful to improve the safety of pedestrians as well as vehicles. This project investigates the prediction of such situations using current traffic conditions and computer vision techniques. An intelligent system may gather and analyze such data in a scene (e.g., vehicle and pedestrian positions, trajectories, velocities, etc.) and provide necessary warnings. The current work focuses on the monitoring aspect of the project. Certain solutions are proposed and issues with the current implementation are highlighted. The cost of the proposed system is low and certain operational characteristics are presented.

17. Document Analysis/Descriptors		18. Availability Statement
Intersections Monitoring Vehicle detectors	Automatic vehicle monitoring Computer vision Collision avoidance systems	No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	39	

Managing Suburban Intersections Through Sensing

Final Technical Report

Prepared by:
Harini Veeraraghavan
Osama Masoud
Nikolaos P. Papanikolopoulos

Artificial Intelligence, Robotics, and Vision Laboratory
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455

December 2002

Published by:
Intelligent Transportation Systems Institute
University of Minnesota

CTS 02-07

Table of Contents

List of Figures

Executive Summary

Increased urban sprawl and increased vehicular traffic have resulted in an increased number of traffic fatalities, the majority of which occur near intersections. According to the National Highway Safety Administration, one out of eight fatalities occurring at intersections is a pedestrian. An intelligent, real-time system capable of predicting situations leading to accidents or near misses will be very useful to improve the safety of pedestrians as well as vehicles. This project investigates the prediction of such situations using current traffic conditions and computer vision techniques. An intelligent system may gather and analyze such data in a scene (e.g., vehicle and pedestrian positions, trajectories, velocities, etc.) and provide necessary warnings. The current work focuses on the monitoring aspect of the project. Certain solutions are proposed and issues with the current implementation are highlighted. The cost of the proposed system is low and certain operational characteristics are presented.

Chapter 1 Introduction

An intelligent system using computer vision techniques to monitor intersections and predict intersection collisions is proposed. The input to the system consists of grayscale image sequences of the traffic scene to be monitored. The first step in recognition is separating the background from the foreground. Traditional computer vision techniques involve using a fixed non-adaptive background image subtraction which is however not suitable for outdoor images owing to changes in illumination and lighting conditions. Standard adaptive background estimation schemes involve averaging the scene over time, which results in an approximate representation of the current scene except for the places where there is motion. This method works well for cases where the objects are continuously in motion and the majority of the background is visible most of the time. However, it does not handle situations where the objects are moving slowly, the background is covered for longer durations, etc. making it unsuitable for the present application. An approach similar to the Stauffer *et al.* [6] is used for background estimation. This method uses a mixture of Gaussian models to estimate the background. This method is very robust to scene changes and recovers very quickly when the background changes rapidly.

Once the background is separated from the foreground, the individual regions in the foreground image are extracted and a minimum bounding rectangle is computed for each. Oriented boxes are chosen as opposed to simple bounding rectangles due to the closer fit these boxes can provide as opposed to the latter type. These bounding boxes are used for the tracking and the classification of blobs in the image sequence. Several experimental results are presented in this report.

System Description

Figure 1 shows an outline of the system:

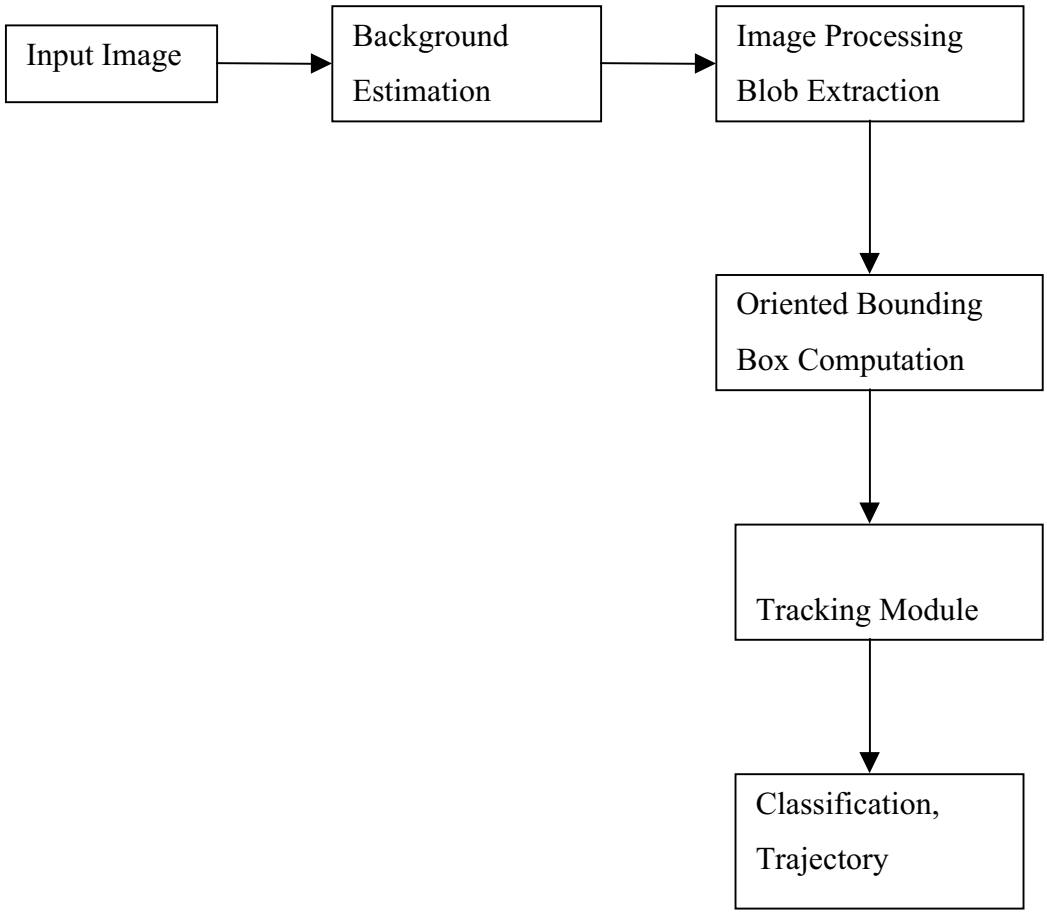

Figure 1. *Outline of the system*.

The input to the system consists of a sequence of images taken from the scene to be monitored. For the experimental setup, the image sequences are obtained from a VCR. Each image is smoothed with a Gaussian filter to remove any noise and then passed on to the Background Estimation Module. In this module, the new background is generated using a Gaussian Mixture model for each pixel in the image (explained in the following

sections). The estimated background is then passed on to the Image Processing Module. The current image is subtracted from the estimated background image and then binarized. This is then passed on to the Blob Extraction Module where the individual regions in the image are extracted using a two-pass connected region extraction. The statistics for each blob (such as area, position, length, elongation, bounding box, and velocity) are computed. After filtering spurious blobs, the remaining blobs are passed on to the next stage wherein the oriented bounding boxes are computed using principal component analysis. Then, the blobs are passed on to the tracking and classification modules.

Chapter 2 Background Estimation

Motivation

Traditional background estimation methods use a fixed non-adaptive background model. Every new image is subtracted from the background image to obtain the foreground. This method is not suitable for outdoor images owing to its incapability to handle changing lighting conditions, shadows, etc.

Other, more robust methods of background estimation include Kalman Filtering, Adaptive Gaussian modeling of the entire image, averaging the image over time, etc. Rosin *et al.* [3] use adaptive background by computing the median images over time. One other method is inter-frame differencing between a set of three images. This method is very fast, however it suffers from the problem of 'ghosts' consisting of trails behind the moving objects. A more robust method is the Pfinder [9] that uses a single Gaussian to cluster pixels in the image.

Time averaging of background images is robust and provides an approximate representation of the scene except for the places where the objects have moved. This method is quite robust in handling changes in lighting and illumination. A typical approximated background with the corresponding difference image is as shown below.

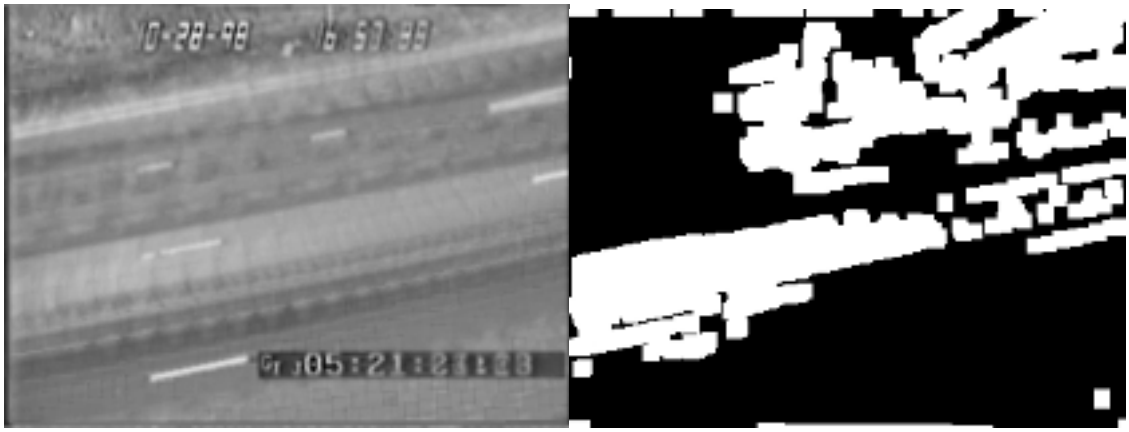

Figure 2. *Approximate background of a highway scene (left) with its difference image (right).*

However, this method requires the objects in the scene to be continuously in motion and does poorly in cases where the objects are slow-moving and when majority of the background gets covered. This method also cannot handle bimodal backgrounds, and recovers slowly when the background is uncovered—resulting in 'ghosts' behind the moving vehicle. In addition, this method assumes a fixed threshold throughout the entire image. A typical example of the ghosting effect is shown in Figure 3.

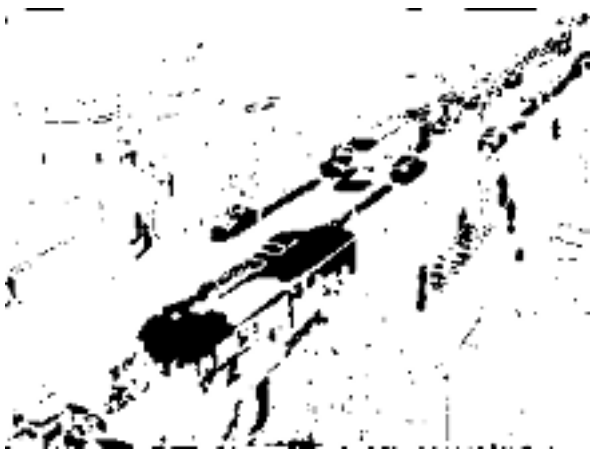

Figure 3. *Typical traffic scene with ghosts behind cars (Foreground in black and background in white).*

A typical grayscale background image is shown in Figure 4.

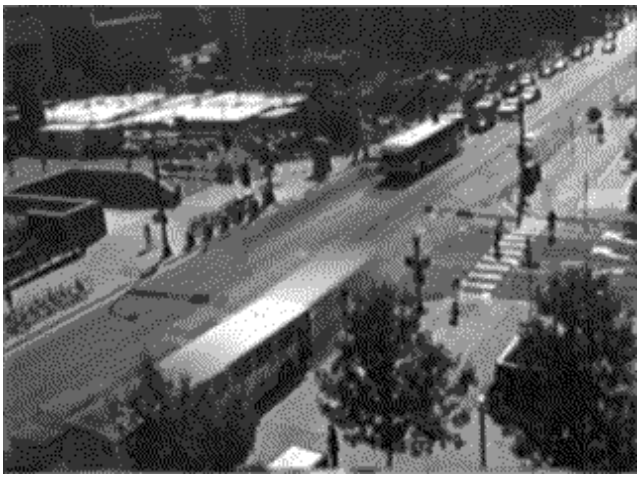

Figure 4. *Grayscale image where the ghost-effect of the bus can be seen.*

One of the main problems with the 'ghosts' occurs when there is a platoon of slow-moving vehicles. In such cases, there is the possibility that all the vehicles will be merged into one long vehicle. This causes tracking errors and possibly incorrect trajectory generation. To overcome these problems, the Gaussian Mixture Models approach used by Stauffer *et al.* [6] is implemented. The method consists of modeling a pixel value as a mixture of Gaussian distributions rather than a single Gaussian distribution. Based on the persistence and variance of each Gaussian mixture, the Gaussian(s) corresponding to the background color are determined. Those that do not fit into the background distribution are classified as foreground. This system adapts very robustly to lighting changes, slow moving objects, cluttered scenes, etc. Slow moving objects will take longer to be absorbed into the background owing to their higher variances. This method can also learn repetitive variations and maintains a model of the background distribution even when it is temporarily replaced by another distribution. As a result, recovery is very fast when the background is uncovered again.

Method

To account for the variation in the lighting intensity and the other changes that occur at a particular pixel, a mixture of adaptive Gaussians is used to approximate the process. Each Gaussian distribution is updated in every frame. The distributions corresponding to the background are isolated after the Gaussian distribution update in every frame. The distributions with least variance and maximum weight will be isolated as the background while the remaining ones correspond to the foreground. The values of a particular pixel over time are considered as "pixel process". A "pixel process" is thus just intensities of the pixel over time. These in our case are grayscale values over time. The recent history of pixel intensity is modeled as K Gaussian distributions. The probability that a pixel of particular Gaussian distribution will occur at a time is determined by

$$P(X_t) = \sum_{i=1}^{k} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \sum_{i,t})$$

where k is the number of distributions, $\omega_{i,t}$ is an estimate of the weight of the i^{th} Gaussian in the mixture at time t, $\mu_{i,t}$ and $\Sigma_{i,t}$ are the mean and covariance matrix of the i^{th}

distribution at time t, and where η is a Gaussian probability density function. This function η is given by:

$$\eta(X_t, \mu, \sum) = \frac{1}{(2\pi)^{\frac{n}{2}} \left|\sum\right|^{\frac{1}{2}}} e^{\frac{-1(X_r - \mu_t)^T \sum^{-1} (X_r - \mu_t)}{2}}.$$

k is determined by the available memory and the computational speed of the hardware. For computational reasons only the values along the main diagonal of the covariance matrix are used for computation, which in our case are scalar values.

Thus, a mixture of Gaussian distributions characterizes the distribution of recently observed pixel values in the scene. Every new pixel value will be matched against the existing distributions until a match is found. A match is defined as the pixel value within 2.5 standard deviations of a distribution. This value is just a threshold and can be altered depending on the image scenes.

When none of the distributions match, the least probable distribution is replaced by the new distribution and the mean is initialized to the pixel value of the new pixel. The variance is initialized to an initially high value and the weight to a very low value.

The prior weights of all the distributions are then updated as

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha(M_{k,t})$$

where α is the learning rate and $M_{k,t}$ is 1 for the matched distribution and 0 otherwise.

The learning rate parameter determines the speed with which the parameters in a distribution will be altered and thus determines the speed with which the background model will be learned.

The μ and σ parameters remain unchanged for the unmatched distributions while they are updated as shown below for the matched distribution:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$
$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T (X_t - \mu_t),$$
where
$$\rho = \alpha\eta(X_t \mid \mu_k, \sigma_k)$$

is the learning factor for updating the current distributions.

Background Model Estimation

As the parameters for the pixel change continuously, the background model is determined by identifying the Gaussians having maximum weights, or supporting evidence and least variance. For example, a stationary persistent object in the image would have least variance and accumulate higher weights with time. On the other hand, when a new object or a moving object occludes the background, this would result in either creation of a new distribution with a very high variance and very low weight or in increased variance in the existing distribution. That is, the models with a high ω/σ value correspond to the background.

To model the above, the Gaussian distributions are arranged in the increasing order of ω/σ in an array with the least probable ones towards the end and the most probable distributions towards the beginning. Thus, when a distribution is re-estimated, the Gaussians have to be sorted from this distribution in the beginning. The least probable distributions will move towards the end of the list and can be removed from the end and replaced with a new distribution when needed.

Thus, the first B distributions are considered as the background where

$$B = \arg\min_b (\sum_{k=1}^{b} \omega_k > T).$$

T is a threshold which gives the measure of the minimum portion of the data to be included as the background. Thus, a higher value of T results in more colors to be included in the background, while a low value results in very few colors included in the background.

One of the significant advantages of the above-described method is that even when an object is temporarily allowed to become part of the background, a background model of the previous background is maintained. Thus, when the object (e.g., a stationary vehicle) begins to move again and the background is uncovered, recovery is very fast. This completely eliminates the problem of 'ghosts.' Another advantage of this method is that the system learns the repetitive motions or noise in the image sequence and adapts it to the background model. An example of an image obtained using this method is shown in Figure 5.

Figure 5. *Image obtained from method (black-background and white-foreground)*

Chapter 3 Image Processing and Blob Extraction

Once the background is estimated, the current image is subtracted from the background to obtain the foreground. The resulting difference image is then binarized using a threshold determined most suitable for the image sequence and then subjected to erosion and dilation. Then various regions in the foreground are extracted using a two-pass connected region extraction method. The regions or blobs correspond to the individual objects in the image. Various statistics such as area of the blob, zero-, first-, and second-order moments, elongation, perimeter, length, etc. are also computed for each blob. Spurious blobs such as those due to continuous motion of the entire image (which occur at the corners of the image) are filtered out and so are the blobs which are really small in size. The remaining blobs are then passed on to the next stage where the oriented bounding boxes are computed.

The reason for using oriented bounding boxes is to obtain closer fit to elongated objects moving along a direction not perpendicular to the optical axis. This holds true especially in the case of vehicles. Simple bounding boxes will not provide a good fit to the vehicles, which might result in tracking errors. The oriented boxes provide the position of the blob. The blobs along with the oriented boxes are passed on to the next stage, namely tracking.

Oriented Bounding Box Computation

The oriented boxes are computed using the covariance matrix of the blob. The covariance matrix of each blob is given by:

$$\begin{bmatrix} M_{20} & M_{11} \\ M_{11} & M_{02} \end{bmatrix}$$

where

$$M_{ij} = \sum_{R} (x - \bar{x})^i (y - \bar{y})^j$$

and

$$M_{20} = \sigma_x^2 = \frac{1}{n} \sum_{i=0}^{n-1} (x_i - \overline{X})^2$$

$$M_{02} = \sigma_y^2 = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \overline{Y})^2$$

$$M_{11} = \sigma_{x,y} = \frac{1}{n} \sum_{i=0}^{n-1} (x_i - \overline{X})(y_i - \overline{Y})$$

where

$$\overline{X} = \frac{1}{n} \sum_{i=0}^{n-1} X_i \; and \; \overline{Y} = \frac{1}{n} \sum_{i=0}^{n-1} Y_i .$$

The above can be calculated using the zero-, first-, and second-order moments of the

blob. Once the covariance matrix is computed, principal component analysis is used to obtain the eigenvectors. Eigenspace representation gives the first two principal axes of the space represented by *M*. Eigenvalues represent the importance of these axes. Diagonalising *M* gives

$$M = \Delta^t . D . \Delta$$

where Δ represents the space change matrix and *D* the diagonal matrix. We also assume $[v_1, v_2] = \Delta$ and $D = \begin{bmatrix} e_1 & 0 \\ 0 & e_2 \end{bmatrix}$.

If $e_1 > e_2$ then we choose e_1 as the first principal axis with vector v_1 and elongation $2.e_1$. The angle made by the principal axis to the x-y image coordinate can be computed from the vector. Similarly, e_2 is chosen as the second principal axis with vector v_2 and elongation $2.e_2$. The result of applying principal component analysis is shown in Figure 6.

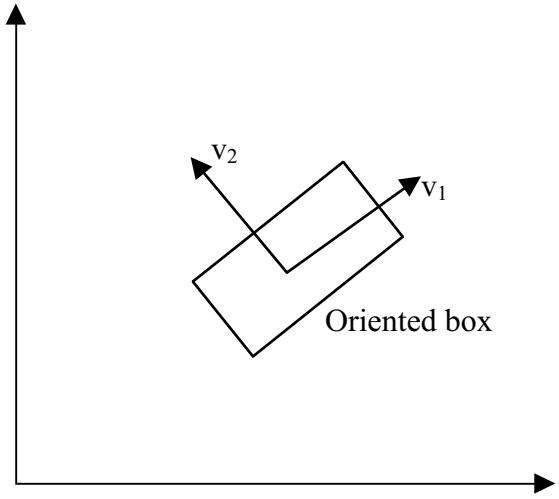

Figure 6. *Oriented box with its principal axes.*

Chapter 4 Tracking

Tracking of the objects in the scene is done at the blobs level. The blobs obtained after image subtraction and after estimation of their oriented boxes are passed on to the tracking phase. Tracking is done by obtaining the association of the new blobs obtained from the i^{th} frame with those in the $(i-1)^{th}$ frame. This is done based on the Masoud's pedestrian tracker [2]. The relation between the blobs in the two frames is expressed by an undirected, many-to-many connected bipartite graph. This is because with each new frame, the blobs can split, merge, appear or disappear. Thus, tracking is equivalent to computing the new relations between the blobs on either side of the graph. To simplify graph computation, we do not consider graphs with vertices having more than degree one. That is, from one frame to another, a blob may not participate in both a splitting and merge operation. This is known as the *parent structure constraint.*

To reduce the exponentially large number of possible graphs, we pose another constraint known as the *locality constraint.* According to this constraint, vertices or blobs can be connected only if their corresponding blobs have a bounding box overlap area which is at least half the size of the bounding box of the smaller blob. These two constraints reduce the possible number of graphs significantly. The second constraint is based on the assumption that a given blob cannot be too far away from its current position in the next frame considering the speed of the blobs in the image.

A cost function is used to compute the optimum graph. The optimum graph is the least cost graph. The cost function penalizes graphs in which blob sizes change significantly. For a perfect match, the blob sizes remain the same. The optimum graph is computed iteratively by computing the least cost graph.

The algorithm works by considering two blobs that satisfy the locality constraint and then checking if they violate the parent constraint. If they don't, they are just related to each other. On the other hand, if they do (a parent blob might be participating in a splitting as

well as a merge operation), then the least cost graph satisfying the parent constraint is computed from all the associated blobs to the current parent. The resulting graph will be an optimal graph. At the end of this stage, the velocities of the blobs are computed.

Computation of the overlap area of the bounding boxes between two blobs is tricky in this case because the rectangles are not simple rectangles and are oriented with respect to each other. There are several different ways in which overlap can occur. Computation of the overlap is done in two steps, the first step being a crude step mainly to eliminate the non-overlapping blobs and the second being a more computational stage where the actual computation of overlap is done. The steps for overlap computation are outlined in the following subsection.

Bounding Box Overlap

In the first step, the overlap is computed by taking the extreme points of the two bounding boxes and computing the overlap between the bounding rectangles formed from the extreme points. This gives a very rough estimate of the overlap between the two boxes. However, this helps to eliminate a lot of unrelated blobs and save computation.

The computation is done as illustrated in Figure 7.

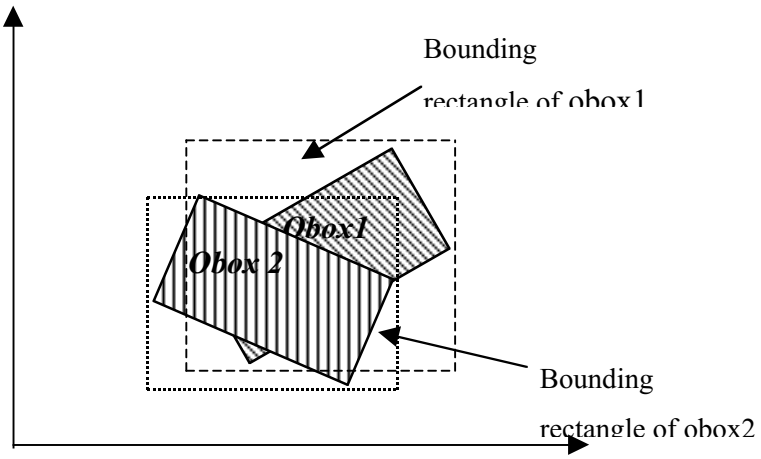

Figure 7. *Oriented box overlap computation—first step.*

The blobs having a bounding box overlap of at least half the minimum bounding box of the two are passed on to the next stage of computation. This size is just a threshold and can be perturbed based on the speed with which objects move in the scene.

In the next stage, the two oriented bounding rectangles are considered as polygons. There can be different cases of overlap based on the number of vertices of one rectangle inside the other namely, zero, one, two, three or all points inside. The total overlap area is computed by considering each rectangle inside the other and computing the number of vertices of one inside the other. Considering each vertex of a rectangle and computing the number of crossovers with the other rectangle does this. If the number of crossovers is even, then the vertex lies outside the other rectangle; otherwise, it is inside. Then, using the number of points that lie inside, the intersection points of the two rectangles are computed. The area is computed by computing the area of the polygon formed by the inside points and the intersecting points. This is repeated by considering the second rectangle inside the first. The total overlap area is the sum of the two overlap areas.

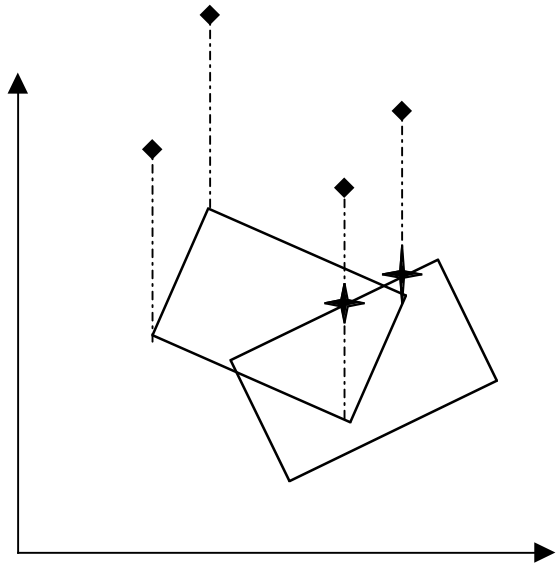

Figure 8. *Computation of the number of inside points (Crossover computed by drawing a semi-infinite line from a vertex and computing the number of lines crossed. An odd number of crossings means the point is inside. Diamond indicates a crossing).*

Chapter 5 Classification

Once the relation between the blobs is obtained, the blobs are classified as pedestrians or vehicles. We accomplish classification by using the principal axis angles obtained from the eigenvectors estimated previously during the computation of the oriented boxes. Pedestrians appear as tall thin blobs. So generally, the main principal axis angle of a pedestrian blob has an angle 90° to the x-axis. On the other hand, vehicles appear as elongated blobs. Hence, the main principal axis angle is horizontal or parallel to the x-axis. This along with certain other features is used for the classification of pedestrians and vehicles. This classification is very fast and generally provides acceptable results, but is not without faults. For example, the classification fails when there is a group of pedestrians walking close to each other. Thus, there are certain issues that need to be addressed. The results of the tracking and the Stauffer's background method are shown in the following pages.

Chapter 6 Results

Tracking examples are shown in Figure 9 using the Stauffer method. The pictures are not from consecutive scenes. The numbers are just to indicate relation between the previous and current blob.

Figure 9. *Tracking results.*

Classification examples are shown in Figure 10.

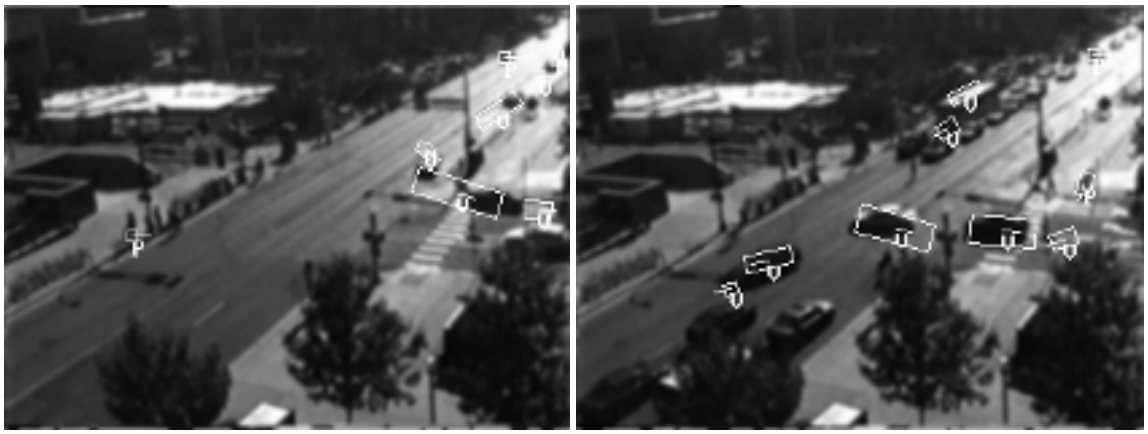

Figure 10. *Classification results.*

Some bad tracking results due to failures of the adaptive background averaging method are shown in Figure 11.

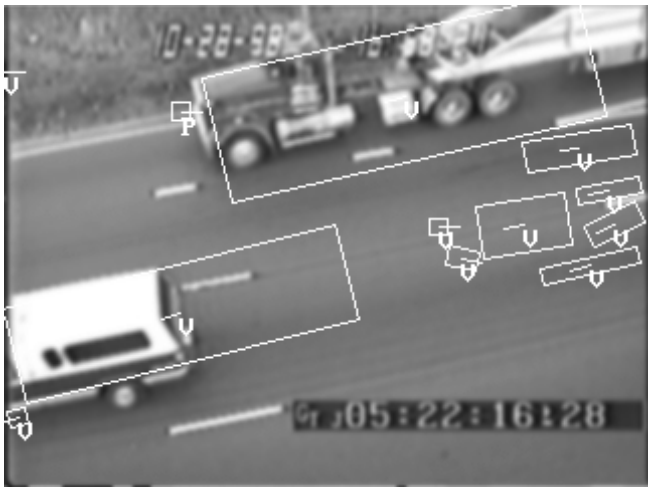

Figure 11. *Unsuccessful tracking results.*

As can be seen from the above picture, the computed vehicle size is bigger than the actual vehicle owing to the ‘ghost’ effect. One may also observe vehicles found in the empty region where there was a very intense ‘ghost’ effect as a result of which the background is misclassified as a part of the foreground.

Figure 12. *Misclassification example.*

In Figure 12, we include an example of occasional misclassification owing to problems with segmentation and shadows. In this case, the pedestrian with his/her shadow is being misclassified as a pedestrian and a vehicle.

Figure 11

Issues

One of the main issues that remain to be solved is segmentation. Owing to inaccuracies in segmentation, a single large vehicle is sometimes misclassified as two or more vehicles. Other problems include misclassification of vehicles, pedestrians, and shadows as shown in some examples above. There is also a problem with tracking that needs to be resolved. This can be handled by using predictive filtering like Kalman filters, etc. The Stauffer method of background estimation, though very fast, is computationally expensive. As a result, the images are subsampled four times and images of size 140x160 are used for the purpose of analyzing the scene to obtain frame rates. The problem with reducing the image size is obvious in that important information can be lost sometimes.

Chapter 7 Future Work

Segmentation is currently one of the biggest problems that has to be addressed. As the images are dealt with only at the low blob level, a lot of information is lost. A better way to approach the problem would be to work at higher image levels for classification and segmentation purposes. The present background estimation, while capable of producing a good estimate of the background and being very robust, is computationally expensive. As the entire image needs to be analyzed to estimate the background, the computation of background increases dramatically with the size of the image. So images are subsampled from 480 x 512 pixels to 140 x 160 pixels to speed computation. Again, the computational time increases as the number of Gaussian kernels increases. As a result (even if higher number of kernels would provide better approximation), we use six kernels at present. This issue could be addressed by using faster hardware than we have at present. Better background estimates can be obtained by using color images rather than grayscale.

Classification, at present, suffers from problems owing to shadows and segmentation. Inaccuracy in classification and occasional tracking errors need to be addressed. Improved tracking can be obtained by using Kalman filtering. Further, with camera calibration, the effects of perspective projection can be dealt with, resulting in improved classification.

More robust techniques for classifying pedestrians and vehicles include classification of objects as rigid or non-rigid. Pedestrians, being articulated entities, appear as non-rigid objects while vehicles appear as rigid objects in a scene. The work in [7] provides a method for classifying objects as rigid or non-rigid. Other techniques like skeletonization of moving objects [8] can also provide fast cues for classification of objects while others such as analyzing the nature of motion of objects can also be used for this purpose.

Chapter 8 Conclusions

This report deals with the intersection monitoring problem. A set of cameras detects situations that may evolve into collisions. Vehicles and pedestrians are detected and tracked in this context. To perform robust detection, an effective method for background estimation has been implemented. A method that provides a more accurate estimate of the foreground has also been implemented. Oriented bounding boxes are used for computing the bounding rectangles around the blobs. They provide a very close fit and as a result improved tracking compared to simple bounding boxes. These methods along with the use of principal axis angles provide a good method of classifying traffic objects in a scene. Several experimental results from real intersections are presented.

References

[1] C. Smith, C. Richards, S. A. Brandt, and N. P. Papanikolopoulos, "Visual tracking for intelligent vehicle-highway systems*", IEEE Trans. on Vehicular Technology,* vol. 45, no. 4, pp. 744-759, Nov. 1996.

[2] O. Masoud, "Tracking and analysis of articulated motion with application to human motion", Ph.D. Thesis,. Dept. of Computer Science and Engineering, University of Minnesota, 2000.

[3] P.L. Rosin and T.J. Ellis, "Detecting and classifying intruders in image sequences", 2nd *British Machine Vision Conf*., Glasgow, pp 293-300, 1991.

[4] C. H. Morimoto, D. Dementhon, L. S. Davis, R. Chellapa and R. Nelson, "Detection of independently moving objects in passive video", *In Proceedings of Intelligent Vehicles Workshop,* I. Masaki (ed.), Detroit, MI, pp. 270-275, Sept. 1995.

[5] P. Remagnino, A. Baumberg, T. Grove, D. Hogg, T. Tan, A.Worrall, and K. Baker, "An integrated traffic and pedestrian model-based vision system", In *Proceedings of BMVC97*, vol. 2, Colchester, University of Essex, UK, pp. 380-389, September 1997.

[6] C. Stauffer and W. Eric L. Grimson, "Learning patterns of activity using real-time tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, August 2000.

[7] A. Selinger and L. Wilson, "Classifying objects as rigid or non-rigid without correspondences", *DARPA Image Understanding Workshop* (IUW), Monterey, CA, November 1998, pp. 341-348.

[8] Fujiyoshi and Lipton, "Real-time human motion analysis by image skeletonization", IEEE Workshop on *Applications of Computer Vision* (WACV), Princeton, NJ, October 1998, pp. 15-21.

[9] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* July 1997, vol. 19, no. 7, pp. 780-785.