Multi-view Intelligent Vehicle Surveillance System

S. Denman, C. Fookes, J. Cook, C. Davoren, A. Mamic,
G. Farquharson, D. Chen, B. Chen and S. Sridharan
Image and Video Research Laboratory
Queensland University of Technology
GPO Box 2434, 2 George Street,
Brisbane, Queensland 4001
{s.denman,c.fookes,j.cook}@qut.edu.au

Abstract

This paper presents a multi-view intelligent surveillance system used for the automatic tracking and monitoring of vehicles in a short-term parking lane. The system has the ability to track multiple vehicles in real-time across four cameras monitoring the area using a combination of both motion detection and optical flow modules. Automated alerts of events such as parking time violations, breaching of restricted areas or improper directional flow of traffic can be generated and communicated to attending security personnel. Results are shown using surveillance data captured from a real multi-camera network to illustrate the robust and real-time performance of the system.

1. Introduction

The prevalence of surveillance systems for tracking and monitoring humans and vehicles using video information is expanding rapidly in recent times. Improvements in computing power, increased safety concerns, increased global threats and a greater awareness and pursuit in developing country's and organisation's security capabilities have all contributed to this realization. Vehicle tracking systems in particular are finding increased use in a range of intelligent transportation systems and applications [5], including measuring traffic flow parameters, detecting accidents [6, 7], autonomous guided vehicles, and for surveillance monitoring in security applications.

Extracting moving objects from video information is an extensively studied problem in computer vision and has spawned a wide range of segmentation, detection, prediction and heuristic techniques. Most vehicle tracking systems can be classified into four broad categories: model-based, region-based, active contour-based, or feature based tracking [2]. The design of algorithms for traffic and vehicle monitoring systems is simplified considerably by taking advantage of the following constraints, which are common to most traffic surveillance applications,

- When the camera is not in motion, most of the image remains relatively static. This allows simple motion detection techniques to be used to segment vehicles from the background.
- Most roads are flat; while the view field is still essentially three dimensional, motion modeling and prediction may be simplified by restricting velocity parameters to two dimensions.
- While in transit, vehicles exhibit no articulated motion; this greatly simplifies shape/contour estimations and models (however, especially when intersections are considered, they can exhibit significant pose variance).

Most vehicle recognition and tracking systems generally consist of a motion segmentation stage followed by a tracking stage. Subsequent to this, depending on the specific system design and application, a recognition module may be incorporated that will use data from the rest of the system to make inferences about the objects being tracked. These may include car/pedestrian discrimination, type of vehicle and the make of car. Predominantly, some of the major problems still hindering the performance of vehicle tracking systems include operation in uncontrolled illumination and weather conditions, vehicle occlusions, drastically changing vehicle poses, and effectively tracking in a multi-camera network.

This paper presents an automatic vehicle monitoring system that is capable of tracking vehicles across a surveillance network consisting of four cameras. Events of interest are detected in this system and are subsequently communicated to attending security personnel. The outline of the paper is as follows. Section 2 will provide an overview of the surveillance system and software design while Section 3 will present the tracking system itself. Results will be presented in Section 4 and finally the paper is concluded in Section 5.

Proceedings of the IEEE International Conference
on Video and Signal Based Surveillance (AVSS'06)
0-7695-2688-8/06 $20.00 © 2006 IEEE

2 Surveillance System Overview

Although this paper presents only a vehicle tracking system, the system design adopted is quite modular and generic to allow for future expansion. The Human-Machine Interface (HMI) or Graphical User Interface (GUI) presented to the user will allow a large degree of control, including selection of input video feeds, selection of desired trackers (eg. vehicle or pedestrian trackers or a combination, etc), and control over the output viewing characteristics. The overall system design, the data flow which takes place between the modules, and software layers comprising the system are shown in Figures 1, 2 and 3 respectively.

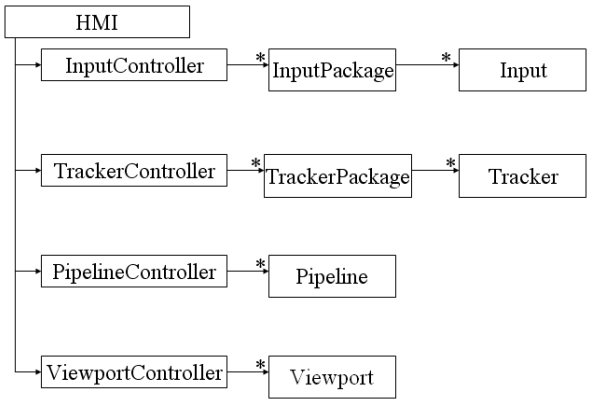

Figure 1. Surveillance Software System Overview

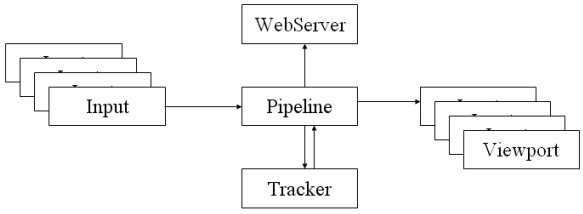

Figure 2. Data Flow

The system consists of three main layers. These layers are the GUI layer, the controller layer and the device layer. The system has been designed as a Model-View-Controller based application; where the GUI constitutes both the View and Controller segments and the Input, Pipeline, Tracker and Output constitute the Model. The GUI layer is used to gain user input for the system as well as to provide some restrictions on what the user can and cannot do. The Controller layer is used by the GUI in order to create the Pipeline and to arbitrate the devices (input and tracker). The Pipeline follows the simple algorithm of input, processing and output. The GUI layer, controller layer and Pipeline are all part of a .NET (executable) package. The device layer is

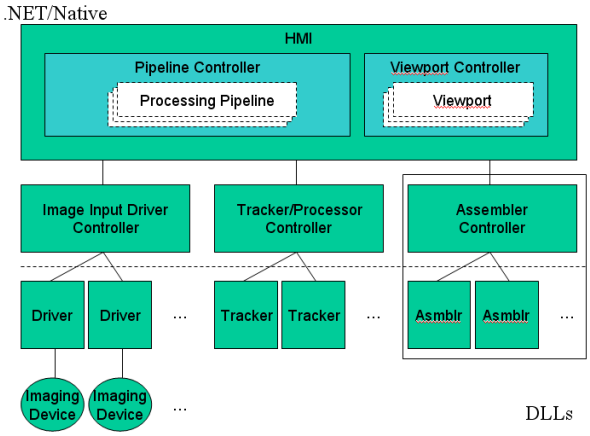

Figure 3. Software Layers

implemented through a DLL interface to allow the dynamic addition of devices (that implement the required interface) without having to edit and recompile the original executable module. The DLL layer also provides a loose coupling between the devices (input and trackers) and the main module (GUI and Pipeline). The DLLs are dynamically linked to the application at load time. Most of the application is native to ensure performance is not degraded to a large degree. Most of the GUI is .NET with the exception of the viewport which is implemented in .NET, native C and inline assembly.

The image buffer is a simple FIFO (First In First Out) queue containing four elements which directly map to the four worker threads. After four images are read into the buffer it pauses for 100ms to force the tracker to run at a constant 10fps. The buffer does not grow but rather it overwrites the existing image data thus ensuring that both the newest images are processed as well as preventing a backlog of images occurring.

From Figure 2, it can be seen that the pipeline is also connected to a Web Server. This is used to transmit automated alerts to a hand-held PDA (personal digital assistant) held by attending security personnel. Transmission occurs via a wireless (wi-fi) network and the generated alerts include information such as a picture of the offending vehicle, parking location and parking times. Other events of interest are also communicated in this fashion.

3 Tracker System Overview

The system is required to cover a four camera network. Each camera (or view) is monitored by a single camera tracker. The four independent trackers are tied together by a management module, responsible for aggregating the tracks and determining when tracks are moving from one view to the next, or have left the system entirely.

The system has five threads to take full advantage of the

Proceedings of the IEEE International Conference
on Video and Signal Based Surveillance (AVSS'06)
0-7695-2688-8/06 $20.00 © 2006 IEEE

parallelism in the system. Each tracker resides in its own thread. The fifth thread contains the management module. The management module feeds images to the trackers, and reconciles the tracks at the end of each frame.

Vehicles are represented in the system as 'Tracked Objects', each of which consists of one or more 'Object Views'. An object view represents the object as seen by a single camera, and stores position information in image co-ordinates as well as other information such as colour. Views are added or removed as the vehicle moves between cameras. The 'Tracked Object' also contains the position of the vehicle in world co-ordinates.

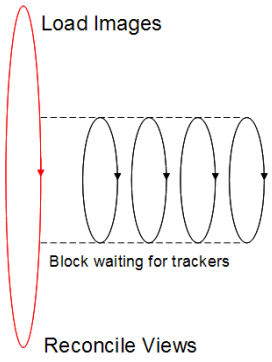

Figure 4. Thread Control

The system has a set of defined zones which specify where tracked objects can exist, be created, destroyed, and move between views. This helps to reduce errors, as objects are only added at known entry points.

3.1 Tracking Process

Each tracker is based upon an adaptive motion detection/optical flow technique [3], with some modifications to allow it to better handle complex scenes and stationary foreground objects. This is applied to the incoming images, and the resultant motion and optical flow images are used to detect vehicles. Motion detection/optical flow is only calculated over the road regions of the frame, as we are not concerned with pedestrian movement. This tracking process is illustrated in Figure 5.

The vehicle detection process uses a simple perspective transform to compensate for the angle the camera is mounted at. This is used to compute the appropriate thresholds for the detection method, given the position of the object in the scene. Minimum and maximum thresholds are specified by the system, and from these and the perspective transform we determine the appropriate threshold. Vehicles can be detected using either motion detection or optical flow.

To use optical flow for detection, we need to be able to effectively estimate the velocity of the vehicle, meaning we need to have tracked the target object for a period to time to be able to make an accurate estimation. As a result, the initial detection and early tracking are done using motion detection.

Motion estimation is achieved by using two motion models for each vehicle. Input for the motion models is taken from the observed position and average optical flow for the vehicle, obtained by averaging the optical flow for the region where the person was detected. The output of the two models is averaged to obtain an estimate of the motion for the next frame.

The system processes the optical flow images first. Any vehicles that have been detected and tracked for sufficient time to predict velocity are detected using the optical flow. Vehicle detection using motion follows.

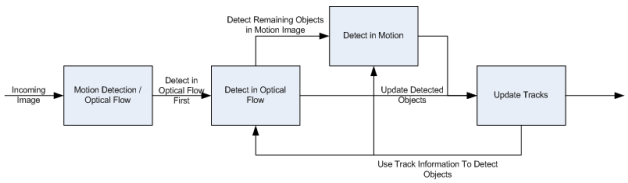

Figure 5. System Flow Chart

All tracks have a state associated with them, which defines how the system handles the track. There are five possible states within the system:

1. Preliminary - Entered into when a track is first created. Tracks in this state must be continually detected, a failure to detect will result in the track being deleted (moved to the 'dead' state)

2. Transfered - Tracks that are moved from another view are created in the transfered state. This is similar to the Preliminary state, but allows for more leeway when detecting the object.

3. Active - The track has been observed for several frames. Tracks spend most of their time in this state. It indicates that the track has been located in the last frame and its position is known.

4. Occluded - Indicates that the track has not been located in the last frame, either due to occlusion or system error. Tracks in this state will switch back to the active state once relocated, or move to dead if they remain occluded for the timeout period.

5. Dead - The track is to be removed from the system. Tracks in this state are deleted when the current frames processing ends.

The state transitions are shows in Figure 6.

Proceedings of the IEEE International Conference
on Video and Signal Based Surveillance (AVSS'06)
0-7695-2688-8/06 $20.00 © 2006 IEEE

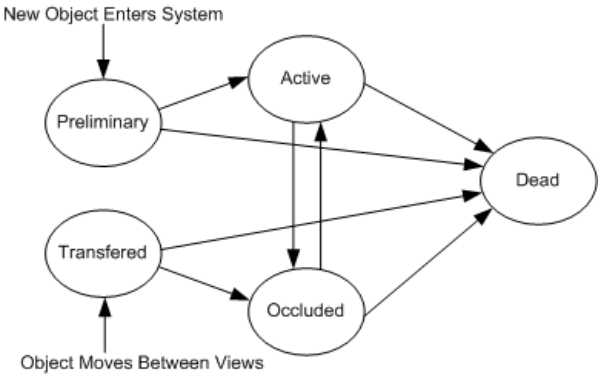

Figure 6. State Diagram for Tracks

3.2 Camera Network and Management

The camera network consists of four calibrated cameras, arranged as shown in Figure 7. Vehicles enter within camera 1's field of view, and exit in camera 4's. Based on this constraint, we only allow the tracker that is handling camera 1 to add new tracks to the system, and tracks can only be destroyed from camera 4. This constraint aids in preventing errors.

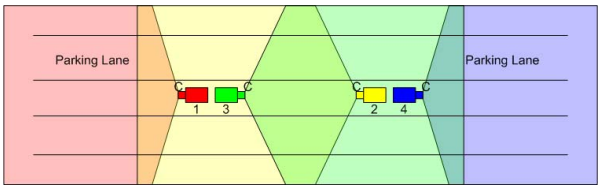

Figure 7. Camera Network

Each camera boundary has a small overlap (a transition area) with the next camera in the network. When a track enters the transition area, the manager creates a dummy track for the object in the next camera view. When the next frame is processed, the tracker responsible for that view will attempt to match the dummy track to one of the detected objects, starting the tracking of the object in the next view.

This provides a safer method of switching tracks between views, as we do not need to create new objects in all views, and attempt to reconcile unassociated tracks at the end of each frame.

The transition zones are wide enough such that there are approximately 10 frames of video (the exact number depends on the vehicles speed) in which this track can be stated, giving the system sufficient time to match the new track.

3.3 Adaptive Thresholding and Feedback

Our system uses a hybrid motion detection-optical flow algorithm [3], which itself is based upon the motion detection algorithm proposed by Butler et al.[1]. Butler[1] proposed an adaptive background segmentation algorithm where each pixel is modeled as a group of clusters (a cluster consists of a centroid, describing the pixels colour; and a weight, denoting the frequency of its occurrence), providing a multi-modal distribution for each pixel.

The motion detection uses colour images in Y'CbCr 4:2:2 format as input. Pixels are grouped into pairs, (2 wide, 1 high) from which four values are used to form a cluster consisting of two pairs (a luminance pair and a chrominance pair). Clusters are matched to incoming pixel pairs (from now on referred to as pixels) by calculating the Manhattan distance between the chrominance and luminance pairs of the incoming pixel and the pairs of the cluster. Thresholds are applied to the distances, and if both are satisfied, then the pixel is suitably close to the cluster to be a match. Once a match is made, the matching clusters centroid and the weights of all clusters in the pixels group are adapted to incorporate the information in the matching pixel. The weight of the matching cluster determines the likelihood of there being motion at that pixel.

If there is no match, then the lowest weighted cluster is replaced with a new cluster representing the incoming pixel, and the pixel is classified as being in motion.

Clusters and weights are gradually adjusted over time as more frames are processed, allowing the system to adapt to changes in the background model so that new objects can be added to the scene (i.e. a box may be placed on the floor), and over time these objects will be incorporated into the background model.

We have added an adaptive threshold to the process to allow the system to handle different lighting conditions within the same scene, and to allow the system to better respond to lighting fluctuations. Regions that are in shadow have a different contrast to regions that are in full sun (or are lit by artificial lights) thus having different ideal thresholds for motion detection.

The adaptive threshold is dependent on the probability of the most likely cluster. The learning rate of the motion detection is such that the more frequently a colour occurs, the higher its corresponding clusters weight (representative of its probability) becomes [1],

$$w_k = w_k + \frac{1}{L}\left(M_k - w_k\right), \tag{1}$$

where w_k is the weight of the cluster being adjusted; L is the inverse of the traditional learning rate, α; and M_k is 1 for the matching cluster and 0 for all others. We assume that the more likely the probability, the more consistent and stable the background colour is, allowing a tighter threshold to be applied,

$$T = T_{max} - \left(w_{max} \times \left(T_{max} - Th_{min}\right)\right), \tag{2}$$

where T is the threshold to be used for matching the pixel; Th_{max} is the maximum threshold; w_{max} is the weight of

the highest weighted cluster; and Th_{min} is the minimum threshold. The learning rate is such that lower weighted clusters will increase in weight faster than higher weighted clusters, so that if the threshold is pulled too low resulting in motion being detected, the weight of the large cluster will be lowered substantially, returning the pixel to a state of no motion. This results in the thresholds for each pixel being able to reach, and approximately remain at, a natural equilibrium.

We have also added a feedback mechanism, that allows the weights of the clusters to be adjusted by feedback from the tracking system. Feedback can be used to reinforce motion detection in regions of interest.

Matching clusters have their weight reduced to prevent the cluster from being incorporated into the background model, while non-matching clusters have their weight increased, to tighten the threshold and increase their sensitivity to motion, as a pixel next to a motion pixel is more likely to be in motion itself. This can be used to prevent a slow moving or stationary foreground object being incorporated into the background model (i.e. a parked car),

$$w_k = w_k \times A^{M_k}, \tag{3}$$

where w_k is the weight of the cluster; A is the adaption rate and M_k is 1 if the pixel is in motion, or -1 if it is not.

4 Results

Table 1 shows the tracking performance of the system. Tracking performance for each camera is shown as well as for the whole system. 3500 frames were hand marked to obtain a ground truth. This ground truth was compared against the actual system output, to obtain the tracking error. A portion of this sequence is shown in Figure 8. The overall euclidean tracking error of 11.41 pixels compares well with other tracking systems ([4] has a tracking error of 3.6 pixels) given the large size of the objects being tracked, and the nature of the camera handover.

Camera	X Error	Y Error	Tracked Instances
1	4.71	5.78	1032
2	9.17	11.30	1714
3	6.64	6.17	205
4	5.04	8.16	205
Total Network	7.28	8.79	3156

Table 1. Tracking Performance

The tracking performance is poorest in camera two, due largely to the inadequacies of the camera network (there is a significant blind spot in transition from camera one to camera two). As the sequence shows however, this is only a problem for vehicles in the outside lanes (the bus), and the inside lane (where the taxi travels) is fully covered. Despite this, the system performs well, and is able to handover tracks across camera boundaries, without loosing the identity of the tracks (it should be noted however that any objects that travel from camera one to camera two in the outside lane will not enter the field of view of camera two at all). The transition zones and level of overlap can be seen in Figure 8, as the bus and taxi moves between cameras.

As the image sequence shows, the only objects tracked are those that enter in camera one. Whilst the system can detect the other cars seen parked in the sequence, as they have not entered through camera one (rather they were there at the start of the sequence), they are not tracked.

5 Conclusions and Future Work

This paper has presented a multi-view surveillance system that can automatically track vehicles across a four-camera network. The system can operate in real-time and results have shown it can robustly track vehicles within each camera and can successfully hand-off tracked vehicles to neighboring cameras in the network. Future work will include: adjusting the camera network to remove the blind spot currently present in the transition from camera one to two; adding additional detection modes (such as colour segmentation and haar detection algorithms) to the system; and developing methods to fuse the detectors. Further testing will also be carried out with the adjusted camera network, and with more complex datasets.

References

[1] D. Butler, S. Sridharan, and V. M. Bove Jr. Real-time adaptive background segmentation. In *ICASSP '03*, 2003.

[2] 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: Emerging Technologies*, 6(4):271–288, August 1998.

[3] S. Denman, V. Chandran, and S. Sridharan. Adaptive optical flow for person tracking. In *Digital Image Computing: Techniques and Applications*, Cairns, Australia, 2005.

[4] S. Denman, V. Chandran, and S. Sridharan. Person tracking using motion detection and optical flow. In *The 4rd Workshop on the Internet, Telecommunications and Signal Processing*, Noosa, Australia, 2005.

[5] K. Jien, T. Watanabe, S. Joga, L. Ying, and H. Hase. An hmm/mrf-based stochastic framework for robust vehicle tracking. *IEEE Transactions on Intelligent Transportation Systems*, 5(3):142–154, September 2004.

[6] S. Kamijyo, Y. Matsushita, and M. Sakauchi. Traffic monitoring and accident detection at intersections. *IEEE Transactions on Intelligent Transportation Systems*, 1:108–119, June 2000.

[7] L. Zhu, J. Song, Q. Huang, M. Zhang, and H. Liu. A novel module of tracking vehicles with occlusion. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 894–899, June 2005.

(a) C1 1 (b) C1 400 (c) C1 660 (d) C1 720

(e) C2 740 (f) C2 780 (g) C3 780 (h) C3 816

(i) C3 880 (j) C3 960 (k) C4 960 (l) C4 1016

(m) C4 1120 (n) C1 1800 (o) C1 1860 (p) C1 1920

(q) C2 1940 (r) C1 1952 (s) C2 1980 (t) C2 2040

Figure 8. Tracking Results - The colour of the bounding box around the track indicates the objects identity. The subtitle of each frame indicates the camera number and the frame number.

Proceedings of the IEEE International Conference
on Video and Signal Based Surveillance (AVSS'06)
0-7695-2688-8/06 $20.00 © 2006 IEEE