A Wide Area Tracking System for Vision Sensor Networks

Greg Kogut, Mohan Trivedi

Submitted to the 9th World Congress on Intelligent Transport Systems, Chicago, Illinois, October, 2002

Computer Vision and Robotics Research Laboratory (CVRR)
University of California, San Diego (UCSD)
La Jolla, CA 92093-040, USA
Email: {gkogut, trivedi}@ece.ucsd.edu
Phone: (858) 822-0002
Fax: (848) 822-5336

ABSTRACT

With the increasing availability of cheap imaging sensors and ubiquitous computer networking, large networks of cameras are being installed in many transportation environments, such as freeway systems, business structures and, and metropolitan roadways. However, most applications of computer vision to Intelligent Transport Systems have dealt with data from single sensors. Even systems with multiple cameras typically process the data from each sensor independently. Data fusion among sensors in a network allows for applications impossible with standalone sensors and processing systems. These include such applications as wide-area tracking and the calculation of trip times for individual vehicles.

This paper describes a distributed computing system capable of managing an arbitrarily large sensor network using common computing and networking platforms. The architecture is capable of handling most common computer vision tasks, as well as the inter-sensor communication necessary for developing new algorithms which employ data from multiple sensors.

This system is tested with a wide-area tracking algorithm which tracks moving objects through a wide-area camera network with non-overlapping fields of view. The sensor network covers a large area of a college campus, and an adjacent interstate freeway. This algorithm allows the system to maintain the identity of tracked objects as they leave and enter the fields of view of individual sensors. Such an algorithm is necessary for applications which require tracking objects over large distances or over long periods of time in an environment without complete sensor coverage.

RELATED WORK

A recent trend in computer vision tracking research has been towards distributed, multi-camera systems, and toward the extraction of rich, high-level information from multiple streams of images. However, research in systems using multiple cameras and processing nodes is less developed. This is at least partly due to the expensive, both monetary and computationally, of vision processing. However, this is changing rapidly with the introduction of powerful multi-purpose processors and digital imaging sensors. These changes have led to the development of camera networks, which have expanded the scope of research in vision-based tracking. In 1996 Rander, Narayanan, and Kanade developed a system for the recovery of scene structure from multiple image sequences (1). In 1999 Boyd, Hunter, Kelly, et al. developed a camera network and processing architecture for 3D tracking and rendering of intermediate views in outdoor and indoor scenes (2). However, both of these systems required some specialized digital signal processing equipment, and elaborate set-up procedures, and neither works in real-time. In 2001 Simone Santini introduced a system which calculates real-time freeway traffic

flow statistics from the input of simple, common traffic cameras commonly used for Web sites (3). In 2001 Trivedi, Mikic, and Kogut presented an architecture allowing for an arbitrary number of cameras to be connected via standard Internet to a network of standard computers (4). This system uses only open standards for image and network transfer, and is platform independent in both sensor and computer hardware. This is a flexible, expandable research platform capable of supporting a wide range or research. This system forms the basis for the architecture used in this paper.

This paper also includes ideas motivated by the work in vehicle signature analysis using inductive loop sensors (5).

OVERVIEW

This paper is paper is divided into two primary sections, one describing the development of the software and hardware infrastructure necessary for the development of a wide-area tracking algorithm, and the other describing the algorithm itself. The development of the software and hardware infrastructure is given separate consideration as research since its development was both necessary to acquire the data used in the research presented in this paper, and because such infrastructures are necessary for producing the described algorithm.

DISTRIBUTED VISION PROCESSING SYSTEM

While there are many sensor networks currently deployed to cover major freeway systems and roadways, most of these networks lack the properties necessary for their use in distributed computer vision applications that might prove useful in ITS. Such properties include:

- Processing at the sensor site
- Broadband data transmission
- Transmission to an arbitrary number of users using standard network protocols
- Use of open compression and network standards (Ethernet, MPEG-2, MJPEG)
- Use of variable levels of compression, depending on application
- Capability for transmitting at variable data rates, frame rates, resolutions, etc.
- Use of ODVS sensors (Omnidirectional Vision Sensor)
- Autonomous control of pan-tilt-zoom motors
- Processing power capable of performing complex image processing simultaneously at each video stream
- Software infrastructure capable of distributing the image processing load to an arbitrary number of processors

This section describes the development of a sensor network and processing test bed at UCSD which provides all the above properties, and allowed for the development of the following wide-area tracking algorithm.

The sensor network, though undergoing continual expansion and improvement, currently covers a large area of the University of California, San Diego campus. This area includes several intra-campus roadways, as well as a section of the adjacent Interstate-5 freeway.

Technical details about the sensor network and diagrams of the area covered by the network may be found at (6).

SENSORS

The sensor network is composed of two types of sensors: pan-tilt-zoom capable rectilinear cameras, and omnidirectional vision sensors which produce a 360-degree view of the surrounding environment. There are currently six rectilinear and four ODVS sensor nodes deployed in fixed positions. In addition to the fixed nodes, there is the capability for additional wireless nodes to be located at arbitrary positions. Each sensor is connected to a small, self-contained server capable of serving data to an arbitrary number of users at a variety of data rates and compression levels, using the open MJPEG standard. The server is also capable of accepting commands for pan, tilt, or zoom movement of the rectilinear cameras.

SENSOR NETWORK

All the fixed-position servers are connected to a local Ethernet system, using standard coaxial and fiber optic cables. Within the CVRR lab, 100 Mbps of data is available simultaneously from each sensor. For outside Internet connections, the servers are capable of producing varying compression and data rates to compensate for lower data bandwidth connections.

Data from the wireless nodes is transmitted either via a High Data Rate (HDR) CDMA modem, or via 802.11b.

The use of standard, open standards for data compression and transmission allows for processing on a wide variety of platforms, and with commonly available development tools.

Figure 1 Image from Freeway Node

SOFTWARE INFRASTRUCTURE FOR DISTRIBUTED IMAGE PROCESSING

While the increase in the power of general purpose computer processing power has greatly increased the feasibility of computer vision processing on common computing platforms, computer vision is still intrinsically very computationally intensive. For most real time vision applications, a single processor is limited to the processing of 1-4 video streams, depending on the application.

Simultaneously processing data from an arbitrary number of vision sensors then requires some method of distributing the processing load. The network and software infrastructures in this paper employ two levels of distributed computing: distribution of basic image processing to the sensor site, and distribution of high level tasks to multiple general-purpose machines.

In the first level of distribution, the tasks of digitizing and compressing video streams are performed at each sensor site. In addition to digitization and compression, the sensors are capable of scaling the data rate by adjusting frame rate, resolution, or color depth depending on the available transmission bandwidth, or the requirements of the application. This initial stage of video processing at the sensor node reduces the transmission bandwidth requirement, and reduces the processing load on the processors performing higher level image processing or computer vision.

The second level of distributed computing is governed by a software load manager, and is used to distribute the processing load necessary for computer vision tasks which require the use of a large number of video streams simultaneously, such as the one described later in this paper. The load manager is able to dynamically distribute tasks to an arbitrary number of machines, depending on the nature of the application being processed. The CORBA architecture is used for inter-machine communication, providing a high degree of language, machine, network, and location independence. The distribution of tasks is performed at a per-video-stream level, meaning that the processing of a single video stream cannot be distributed to multiple machines. However, an arbitrary video streams may be processed by a single machine, and the total number of streams that can be handled is limited only by the available processing power. Each distributed task sends its calculated data back to a central machine. This data tends to be high-level data

extracted form video streams rather than modified video streams. This avoids the transmission of large volumes of data among distributed machines. Figure 1 outlines the load distribution hierarchy.

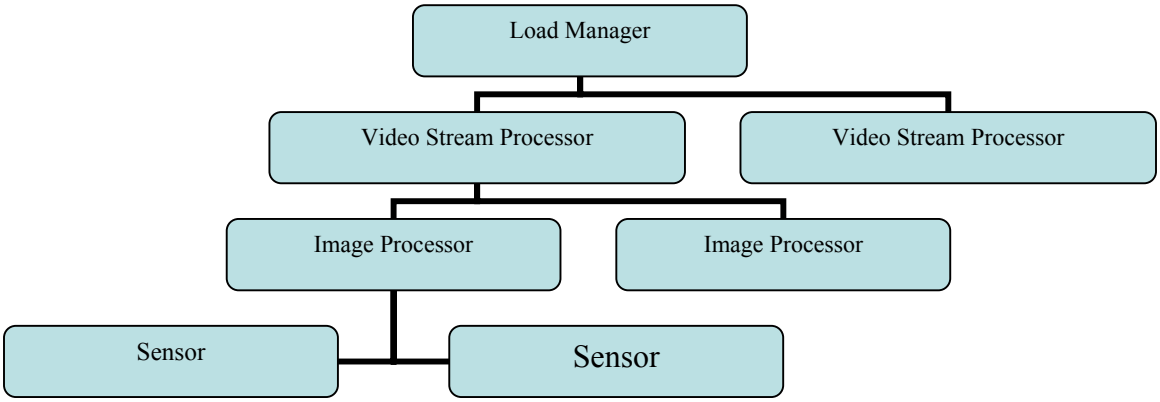

Figure 2: A demonstration hierarchy of processing distribution

WIDE AREA TRACKING ALGORITHM

The single-scene tracking problem has been extensively researched by the computer vision and ITS communities. This problem deals with tracking vehicles as they travel through the field-of-view of a single sensor. However, single-scene tracking tends to only provide a snapshot date of the vehicles in a scene, while many vehicle behaviors occur over greater distances and times. These include such things as trip route, trip time, or lane changing at high speed. Such behaviors are typically not captured within the field-of-view of a single sensor. Data from multiple sensors must be fused to extract the necessary information.

This algorithm described here attempts to track vehicles as they travel along the roadways on the University of California, San Diego campus, including a section of the adjacent Interstate-5 freeway. The sensor coverage is described and shown in the previous section of this paper. It is important to note that the cameras in this network have largely non-overlapping fields-of-view, as is typical of camera networks in most ITS environments due to the prohibitive expense of installing enough sensors for ubiquitous coverage. This incomplete coverage makes the wide-area tracking problem different and more difficult than the camera hand-off problem, which has been studied

extensively. In the camera hand-off problem there is usually a period when the tracked object is within the field-of-view of two cameras, and the object is continuously tracked as the object is "handed over." In the wide-area tracking problem there are periods when the object may be lost to any sensor's field-of-view for significant amounts of time. The problem of the re-identification of tracked objects is therefore a key problem in wide-area tracking. Re-identification solves the problem of maintaining the identity of a vehicle as it travels in and out of multiple fields-of-view. The remainder of this section describes the processing steps involved in wide-area tracking.

The first four steps of processing: segmentation, feature selection, tracking, and platoon detection are processed for each video stream in the network. The final step, graph matching fuses the information from each video stream, and occurs only on one central machine.

The details of several of the processing steps are omitted from this paper since they have been previously published, and are common components of any visual tracking algorithm.

FEATURE EXTRACTION

One of the most critical aspects of vehicle tracking is the choice of which vehicle features to use to uniquely identify vehicles. Previous research has used a wide variety of identifying features such as color, shape, edge features, or pattern-matching against a database of vehicle types (8). These features all have tradeoff associated with their ability to characterize vehicles. There is also a dependency for them to be dependent on the nature of the sensor setup and environment. For example, color features work best in scenes with a close-up view of the vehicle, and without widely varying lighting conditions, while shape features tend to be relatively independent of lighting conditions, but are more susceptible to noisy segmentation algorithms.

This algorithm employs a variety of vehicle features in a feature vector: color, shape, and velocity. The color features are calculated by using partial implementation of the AutoColor Matching System (9). The AutoColor Matching System compensates for differences between illumination at cameras sites and between cameras. The system uses mean and variance values for the R, G, and B channels of a vehicle's color as the feature model. The matching and scoring modules of the AutoColor system are not used in this algorithm. The size of the vehicle is simply described by the length of the major and minor axes of the blob extracted from the segmentation algorithm. This simple metric is usually enough to classify vehicles separately from pedestrians or other moving objects and to classify objects as either cars, or longer buses or trucks. Finally, the velocity of each object is tracked, as described in the following section.

The output of the feature extraction step is a vector of feature data associated with each of the blobs detected in the segmentation stage.

TRACKING

A simple vehicle-tracking scheme identifies identical vehicles in successive video frames from the same camera site. The tracking algorithm uses the color model described above and the blob centroids from the segmentation module, to help solve the data association problem. Full rate video, and predictable target motion in road scenes makes the tracking module accurate in this application, however, problems such as vehicle occlusion introduce some error within this module. A more robust tracking algorithm should be used in future implementations, such as that shown in (8). Vehicles are assigned vehicle IDs (valid only within one camera site, at this stage), and their locations and velocities are tracked continuously while the vehicle travels within the field-of-view of each camera node.

The output of the tracker is unique vehicle identification numbers for each vehicle currently within view of a sensor node.

PLATOON DETECTION

There is information contained in the relative spatial orientation and order of vehicles traveling in close proximity. This information can be employed to greatly reduce the ambiguity in matching cars that travel between two camera sites that are relatively close, and view the same roadway. This wide-area tracker uses a hybrid method, using color and spatial information to construct labeled graphs of small groups, or platoons, of cars, as they travel through a camera site. Then, the task of matching the cars at following cameras sites can be cast as a problem of probabilistic graph matching. In this implementation, a probabilistic hill-climbing algorithm is used to perform the graph matching. Once matching has been preformed, vehicles and platoons of vehicles may be matched between camera sites.

A platoon is a vehicle, or group of vehicles, traveling in close proximity. A platoon in most of ITS literature often refers to groups of cars traveling in the same lane, while in this system the vehicles may be traveling in any lane, and need not maintain strict ordering or relative positioning.

The platoon detection algorithm recognizes groups of cars that are entirely within a pre-defined region of the road scene. The pre-defined area serves to avoid including vehicles that are near the horizon, or that are only partially within the image plane. This avoids including vehicles that tend to have less reliable feature data, such as erroneous size estimates, or missing occluded pixels. New platoons are created only when one or more vehicles either leave or enter the defined region. This avoids the creation of redundant platoons. However, a vehicle may be assigned to multiple different platoons during its trip through the defined region, and every vehicle is assigned to at least one platoon. The graph matching module consists of the algorithms that construct graphs based on the input from the low-level vision module, and the algorithm which matches graph between the two cameras sites.

GRAPH CREATION AND MATCHING

The graph creation module models groups, or platoons, of cars in a road scene as a labeled, undirected graph. Labeled graphs are graphs in which each node and edge has a label. In this paper, nodes vehicle models, and edges encode information about the spatial arrangement of vehicles in the platoon. In the current implementation, the nodes contain the color information calculated in the feature calculation model, and the edges are the Euclidean distance between vehicles.

The module constructs a connected graph from the platoon and spatial information, so that edges exist between a vehicle, and all other vehicles in the platoon. While this increases the computational load on the matching module, described, below, the total complexity is relatively low if platoon sizes are restricted to 1-6 vehicles.

The graph-matching module then matches graphs between the two camera sites. A close match between two graphs suggests a strong likelihood that the graphs correspond to the same, or roughly the same, platoon of cars. A probability threshold is used to determine the matches.

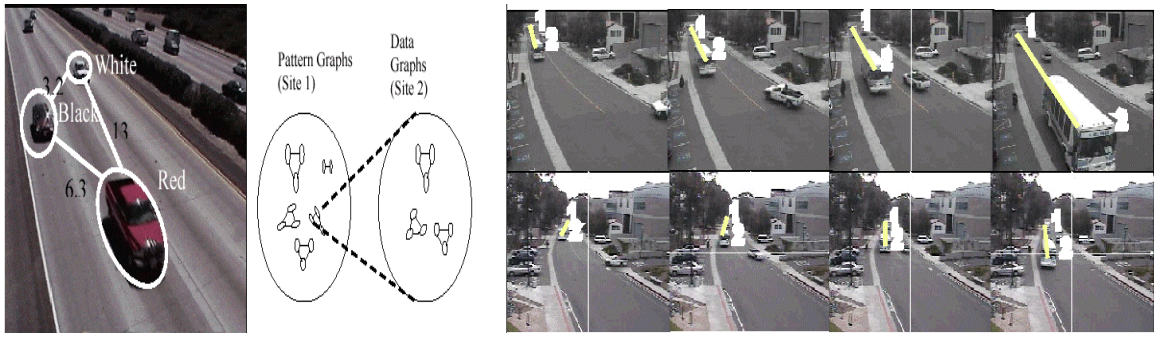

Figure 3: A simplified graph structure, and outline of graph matching process

The first step in the matching algorithm is to create sets of data graphs from Site 2 for each pattern graph from Site 1. Both the distance between the sites, and the platoon velocities are known, so a predicted travel time may be calculated for platoons traveling from Site 1 to Site 2. All platoons at Site 2 that fall within a reasonable time of their predicted arrival time are used as data graphs to match with the pattern graph.

Once a set of data graphs has been assembled, they are matched individually with the pattern graph from Site 1. Because of the noisy nature of computer vision data, and the unpredictable nature of freeway traffic an approximate graph matching algorithm is used, which allows for disparities between the graphs, such as missing or extra nodes. These might result from such common vehicle tracking problems such as occlusion.

The graph-matching module is based on the approximate graph matching algorithms presented in (10). The algorithm constructs a mapping between the pattern graph and

each data graph by finding a sequence of edit operations that would convert the data graph to the pattern graph. The edit operations used are: *node relabeling, node deletion, node insertion, edge relabeling, edge deletion, and edge insertio*n. A cost function, λ, is associated with each edit operation. The cost functions used in this implementation are shown below. *a* and *b* are either edges or nodes in the following notation.

- Node relabeling: $\lambda(a \to b) = (x_a - x_b)' C^{-1} (x_a - x_b)$, the Mahalanobis distance between the color features of the two nodes.
- Node deletion: $\lambda(a \to \Lambda) = |x_a|$, the magnitude of the vector of the node being deleted.

- Node insertion is the same as the above, substituting the inserted node for the deleted node.

Edge cost functions are analogous to node cost functions, with the arithmetic differences and magnitudes substituted for the vector functions.

The total cost function for a graph match is there the sum of all cost functions for a mapping between the data and pattern graphs. It can be shown that the minimum-cost mapping corresponds to the least cost sequence of edit operations. Calculating the minimum-cost mapping can be cast as a state-space search problem. The details of these operations are shown in (10).

Minimum-cost mappings are calculated for all graph-matching combinations. The double threshold evaluation method of was used to determine true matches. In this method, the two least-cost scores are used (9). The best score must pass an absolute cost threshold, and also the difference between the two scores must pass another threshold. This method serves to reduce the number of false positive matches.

Once the best-fitting graph for the pattern graph has been calculated, and has passed the threshold, then the site-to-site tracking data can be calculated for both the platoon, and for individual vehicles within a platoon. This effectively solves the wide-area tracking problem, and can provide such data as point-to-point travel times, changes in velocity, or lane-changing behaviors for the region between the two camera sites.

DATA AND EXPERIMENTS

The algorithm was tested comparing the performance of the algorithm versus a manually-tracked ground truth. The trial data consisted of a several data sets: the deliberate running of an electric cart along various paths through the sensor network, the recording of full-sized vehicles traveling through the network, and the recording and tracking of pedestrians in the areas which adjacent vehicle and pedestrian traffic. The accuracy of the algorithm is calculated as the percentage of the time that a vehicle was correctly re-identified as it entered a new field-of-view after having left the field of view where it first

entered the sensor network. The experiments involved simultaneous calculations on five video streams, and ran on three separate computers.

RESULTS

Data Set	*# Nodes Entered*	*Mean Platoon Size*	*# Successful Reident.*	*Success Rate%*
Electric Cart	18	1.0	11	61%
Full-size Vehicles	27	1.3	14	52%
Pedestrians	33	2.8	19	58%

DISCUSSION

The above test results are preliminary. More test data must be run to explore the weaknesses in the algorithm, and to determine its potential for further development. More trials are being run at the time of writing.

ACKNOWLEDGEMENTS

Our research is supported in part by the California Digital Media Innovation Program in partnership with The California Department of Transportation (Caltrans). We wish to thank our colleagues from the Computer Vision and Robotics Research Laboratory who are also involved in related research activities.

BIBLIOGRAPHY

1. Rander, P.W.; Narayanan, P.J.; Kanade, T. "Recovery of dynamic scene structure from multiple image sequences**,"** 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems, Washington, DC, USA, 8-11 Dec. 1996.) New York, NY, USA: IEEE, 1996. p.305-12.
2. Boyd, J; Hunter, E, Kelly, P. "MPI-Video infrastructure for dynamic environments." Proceedings. IEEE International Conference on Multimedia Computing and Systems, Austin, TX, USA, 28 June-1 July 1998.) Los Alamitos, CA, USA: IEEE Comput. Soc, 1998. p.249-54.
3. S. Santini, "Very Low Rate Video Processing," *Proceedings of SPIE Vol. 4311, Internet Imaging II,* San Jose, Jan. 2001.
4. Trivedi, I. Mikic, and G. Kogut, "Distributed video networks for incident detection and management", *Proceedings of the IEEE Conference on Intelligent Transportation Systems,* Dearborn, Michigan, October 2000.
5. MacCarley, C.A., Videobased Vehicle Signature Analysis and Tracking Phase 1: Verification of Concept and Preliminary Testing. UCB-ITS-PWP-98-10, PATH, University of California, Berkeley, CA.
6. http://cvrr.ucsd.edu.

7. I. Mikic, P. Cosman, G. Kogut, M. Trivedi, "Moving shadow and object detection in traffic scenes," International Conference on Pattern Recognition, Barcelona, Spain, September 2000.
8. D. Koller, K. Daniildis, and H. Nagel, "Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes," *Int. Jour. Of Computer Visio*n, vol. 3, 257-281, 1993.
9. Zeng, N.; Crisman, J.D., "Vehicle matching using color," Proc. ITSC '97, p. 206-11
10. Wang, J.T.L.; Kaihong, Z; Gung-Wei, Ch, "Approximate graph matching using probabilistic hill climbing algorithms," *Proc. Sixth International Conference on Tools with Artificial Intelligenc*e, 1994. p.390-6.