The IEEE 5th International Conference on Intelligent Transportation Systems
3 – 6 September 2002, Singapore.

Real-time Wide Area Tracking: Hardware and Software Infrastructure

Greg T. Kogut, Mohan M. Trivedi, *Member, IEEE*

In recent years, large number of cameras have been installed in freeway and road environments. While the use of some of these cameras is being automated through computer vision, few computer vision systems allow for true wide-area large scaled automation. This paper describes a distributed computing system capable of managing an arbitrarily large sensor network using only common computing and networking platforms. The architecture is capable of handling many 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 an algorithm which tracks moving objects through a prototype camera network with non-overlapping fields of view in a college campus environment. This algorithm allows the system to maintain the identity of a tracked objects as it leaves and enters 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.

Index Terms—**Sensor fusion, computer vision, software infrastructure, sensor infrastructure**

I. Introduction

TRANSPORTATION infrastructures are becoming increasingly sensorized. Most major urban freeway systems currently employ large networks of permanently installed cameras for a variety of purposes. Most of these sensors are used simply for their raw data – a snapshot of current traffic conditions. Computer vision, however, allows for applications which provide processed data, rather than raw video, to users. Such data includes statistical information such as traffic densities, mean velocity vs. time of day, or traffic volume per lane. Some of these applications are currently being explored [1].

Manuscript received July 18, 2002 This work was supported in part by the California Digital Media Innovation Program (DiMi) in partnership with the California Department of Transportation (Caltrans). We also wish to thank our colleagues from the lab who are also involved in related research activities, including Ofer Achler, Andrew Cosand, Brett Hall, and Natalie Roselli.

Greg T Kogut and Mohan M Trivedi are with the Computer Vision And Robotics Research Laboratory, University of California, San Diego, La Jolla, CA 92093-0407 (email: mtrivedi@ucsd.edu; lab web site: http://cvrr.ucsd.edu)

However, most of these systems employ only single video streams. There are few hardware of software infrastructure capable of fusing data from multiple sensors covering a large area, despite the existence of many large scale sensor networks in transportation environments [2]. This paper proposes a hardware and software infrastructure capable of fusing data from multiple sensors in real time. The system is tested with an example application which tracks moving objects in a college campus which is sensorized with multiple cameras with non-overlapping fields of few. The example application is based on a vehicle-matching algorithm which re-identifies vehicles which leave one scene, disappear from any view, then enter another scene at a later time. This ability enables wide-area tracking to take place.

The presentation of this system is divided into five sections: Hardware Infrastructure, Intra-Scene Vision, Inter-Scene Vision, Implementation and Software Infrastructure, and Results

The section on Hardware Infrastructure describes the goals, requirements, and implementation issues of a large-scale, outdoor vision infrastructure capable of supporting high quality, distributed computer vision applications. The Intra-Scene Tracking section describes the methods used to implement common computer vision processing in a modular and scalable framework. And, finally, the section on Inter-Scene Vision describes the implementation of an infrastructure which allows for data fusion among an arbitrary number of sensors. Implementation and Software Infrastructure discusses some of the issues in large-scale vision system design. The paper concludes with a presentation of data and results from an example application with corresponding discussion.

II. Related Work

Research in multi-sensor computer vision is increasing with the introduction of powerful multi-purpose processors and digital imaging sensors. These changes have led to the development of camera networks, which have introduced and 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 [3]. In 1999 Boyd, Hunter, Kelly, et al. developed a camera network and processing architecture for 3D tracking and

0-7803-7389-8/02/$17.00 © 2002 IEEE

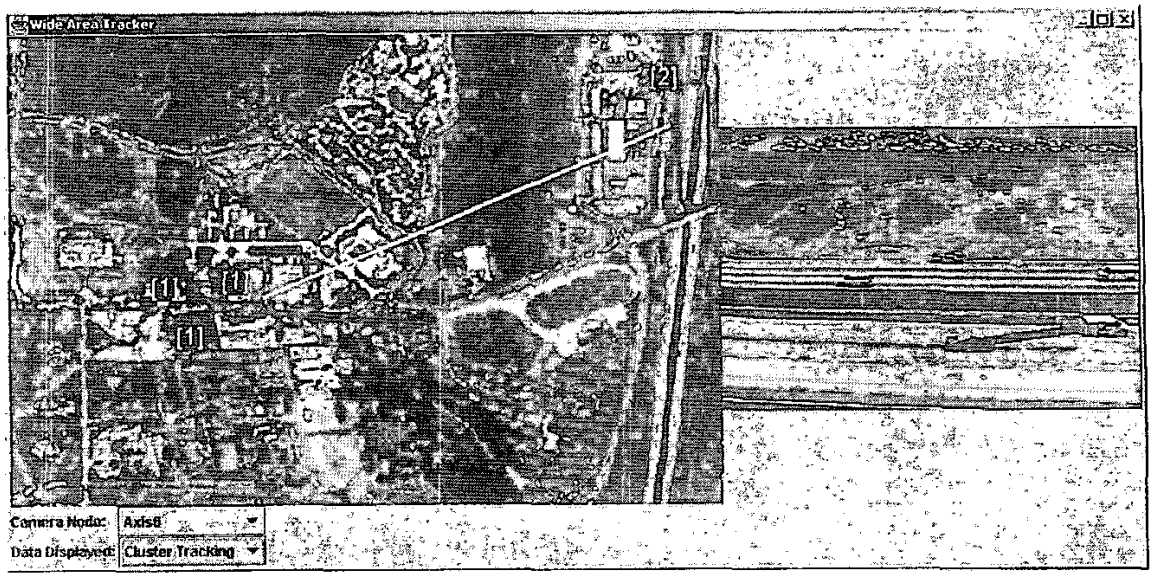

Fig. 1. Application showing 2D Visualization of the wide-area environment being tracked. Live video of the freeway scene is one the right. The black lines over the live video are a visualization of real-time platoon detection, as described in the paper. The yellow numbers superimposed over the map show the number of vehicles currently being tracked at each location (only the near four lanes of traffic in the freeway image are being tracked). The yellow line superimposed over the map indicates a vehicle has just been tracked between the freeway node and the corresponding interior campus node.

rendering of intermediate views in outdoor and indoor scenes [4]. 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 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 [5]. Also, Trivedi, Prati, and Kogut presented Distributed Interactive Video Arrays for Event-based Scene Analysis (DIVA), which is a sister-project to this one, and uses the same hardware infrastructure [6]. This system uses only open standards for image and network transfer, and is platform independent in both sensor and computer hardware.

III. Hardware Infrastructure

Computer vision algorithms depend highly on the quality of calculated vehicle features, such as color, shape, and location. The quality of these features depends, in turns, on the quality of the raw video being used as input data. Most publicly available live traffic video is of relatively poor quality compared to most data used in computer vision research, including inferior dynamic range, resolution, and frame rate. Many also use proprietary network transmission or compression protocols, making the data unavailable for most types of image or video processing.

An Intelligent Transportation vision infrastructure should possess the following characteristics:

- High-resolution video (NTSC or greater)
- High video rate (10fps or higher)
- Data compression for transmission over standard IP-based networks
- High dynamic range or applications which depend on color features
- Scalability, for the construction of networks of arbitrary size
- Modularity, allowing for the addition of new sensor types, protocols, algorithms, or applications

The sensor network installed by the CVRR lab at UCSD during 2000-2002 fulfills all of these characteristics, and was instrumental in the development, implementation, and testing of the algorithms described in this paper. There are several published descriptions of the UCSD ITS infrastructure [5].

IV. Intra-scene Vision

Intra-scene tracking encompasses the algorithms and architecture which deal solely with data acquired from a single camera node. This section does not cover the communication between nodes, nor the algorithms that require data fusion among multiple nodes. The processing steps are performed in the order they are executed in a layered, modular vision architecture: image acquisition, segmentation, connected components and morphology, tracking, and feature calculation.

The goal of intra-scene tracking is to identify each moving object in a raw video stream, and to maintain the object's identity while it persists within the image view.

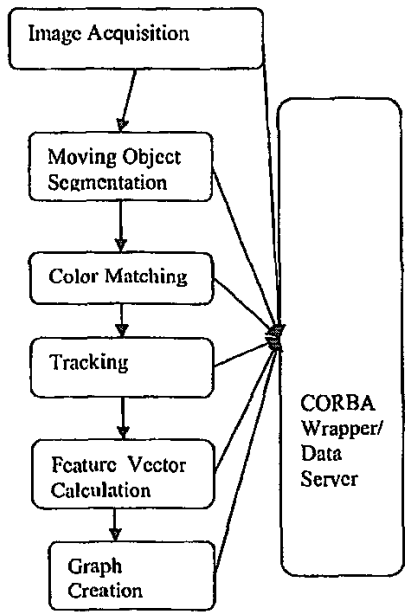

Fig. 2. Data flow of the software infrastructure. The details of implementation are hidden by the CORBA wrapper, allowing for language and platform independence. Modularity allows for easy expandability or change to system design.

A. Image Acquisition

The implemented image acquisition module The input to each single-sensor node is a stream of raw images. There are no requirements of the properties of the images (size, resolution, color depth, etc). Any conventional image or video capture method may be used, such as digital capture cards, USB connections, images streamed over IP networks, or most common video file formats. However, the quality of the acquired data may affect the overall performance and quality of some of the processing steps, such as segmentation. The rate of image acquisition may also affect processing results, depending on the velocity of the moving objects being tracked, and the perspective by which the objects are being viewed. Camera location tends to be a contexts-dependent problem, determined largely by the requirements of the application.

This flexibility in the image acquisition module is achieved through the use of the OpenCV library and the Java Media Framework library in the implementation of the infrastructure [7]-[8]. Methods of implementation are further discussed later in this paper.

B. Segmentation

Segmentation, in the context of this paper, is the identification of moving objects within a dynamic scene. One goal of the infrastructure is a segmentation algorithm capable of working effectively and reliably in a wide range of environments, and for a wide range of applications. Quality segmentation results are critical to any tracking application since errors tend to be propagated to subsequent processing steps. Segmentation is also sensitive to application context and environment, requiring ad-hoc tuning of algorithm parameters.

While the segmentation algorithm in this paper certainly doesn't completely solve all these problems, it is relatively robust with respect to many of the aforementioned problems.

The algorithm is based on an algorithm developed by Ivana Mikic [9]. The algorithm employs local, spatial, and temporal information in segmenting moving objects. The local information is the use of pixel information in determining the segmentation. The spatial information is based on the assumption that moving objects will be represented in the image by compact, connected clusters of pixels. The temporal information is the based on the assumption that the state of a pixel in the current image can be predicted from its state in a previous image.

C. Morphology and Connected-Components

Image morphology and a connected components algorithm are used to extract quality "blobs" from the raw segmentation data. Blobs, in this case, are groups of foreground pixels, as classified by the above segmentation algorithm, which potentially make up a moving object in the environment.

The segmentation algorithm classifies each pixel in an image as one of three classes: foreground, background, or shadow. Segmentation algorithms tend to introduce a significant amount of noise in their output. This noise manifests itself as small, scattered clusters of pixels. These clusters are usually false positive classifications. Therefore it is desirable to remove them before the errors propagate to further processing steps. Common image morphology steps are used to remove these clusters.

The resulting clusters are then input into a connected-components algorithm. The connected component algorithm identifies and labels groups of connected pixels. The connected-components algorithm also uses a 4-neighbor connectivity. During the connected-components processing, the centroid of each blob in image coordinates is also calculated and stored.

D. Color Matching

The perspectives and lighting conditions in each field of view of a wide-area tracking system are likely to vary quite a bit. This variation presents a problem to feature-matching systems which use identifying features such as color and shape in tracking vehicles through non-overlapping fields of view. Also, if multiple types of sensors are used, the color properties of the input data may very significantly. The color matching system used in this paper is the AutoColor Matching System, as

described by Zeng and Crisman. [11]. This system assumes that the color of asphalt is reasonably constant among camera views, and uses the color obtained from the asphalt in each scene to match other colors among the scenes. This details of this matching are omitted since they are outside the scope of this paper.

E. Intra-Scene Tracking

The tracking problem, in the context of this paper, is effectively the data association problem. Specifically, in frame i, there is a set of blobs, B_i, produced by the connected components algorithm. Assuming an error-free segmentation, each of these blobs represents some unique moving object in the scene, and there is a one-to-one mapping between the blobs and the real objects. In the following frame, $i+1$, there is a new set of blobs, B_{i+1}. The goal of the data association algorithm is to find a mapping between the members of B_i and B_{i+1} such that there is a one-to-one mapping between the blobs in each set such that the pairs of mapped blobs both represent the same real object.

A simple, intuitive, solution to this problem was selected for use in this project because of the high quality segmentation results possible, and the relatively low number of tracked objects (<15) present at any given time, avoiding the computational explosion that sometimes results in the data association problem. A limited nearest-neighbor search is used to map blobs in consecutive frame.

Each track is stored within the system as an ordered list of blobs. The state of each track is also stored at each time step. The state includes the time, the object velocity, and the object size. The velocity is calculated using Euclidean distance, along with the time, t, between frame captures. While a motion model, such as those used in Kalman filter-based tracking, could improve the accuracy of the tracking, the results obtained by this method proved sufficient.

F. Feature Calculation

The feature calculation module calculates and stores identifying features of each object being tracked. The determination of which object features best identify classes of moving objects (people, vehicles, etc.) is one of the key research problems in intra-scene tracking. The optimal feature or features to use in uniquely identifying objects is largely application and environmentally dependent. Also, too many features cannot be used, as "data saturation" occurs, which dilutes the power of the feature vector to uniquely characterize an object. The best route is to pick a small set of good features. While a robust technique such as principal component analysis can be used to select those features which best describe the moving objects, for this project, the most obviously visibly salient features were chosen: size, aspect ration, hue, saturation, and intensity. This feature data is calculated and stored for each detected blob in each frame of video. Past data is discarded or recorded to disk on an application-dependent basis to avoid excessive memory requirements. The data structure which describes each blob contains the following information: time stamp of data capture, centroid location, size, aspect ratio, average blob hue, average blob saturation, average blob intensity, and platoon ID (see below).

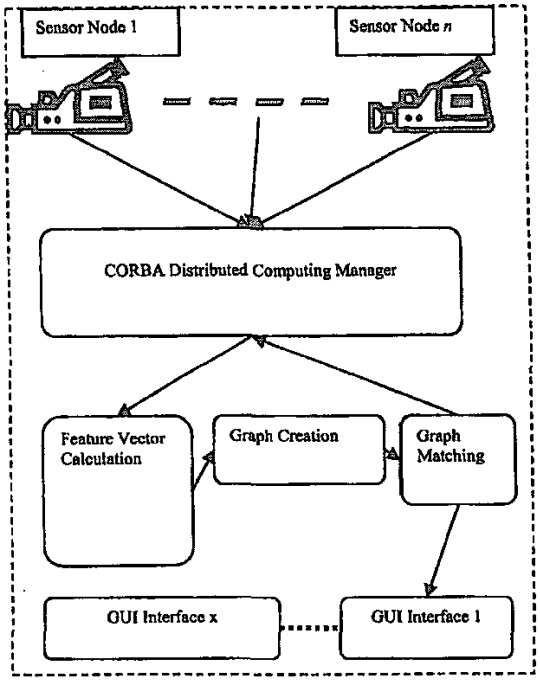

Fig. 3. Data flow between sensor nodes, illustrating the method of distributed computing used. An arbitrary number of sensor nodes can be supported, and can arbitrary number of GUI applications clients can present independent views of the data.

G. Platoon Detection

The idea of a platoon is derived from Intelligent Transportation literature where a platoon is defined as a sequence of two or more vehicles traveling in close proximity in the same direction, and in the same lane [11]. In this application, the definition of the term is relaxed to include any group of one or more objects traveling in the same direction, but in any spatial configuration such that all the moving objects simultaneously fit in one camera frame. The objects need not maintain strict ordering or relative positioning.

For each frame, the feature calculation determines that a group of blobs traveling within a certain proximity are a platoon, and assigns each platoon an ID for the duration of the existence of the platoon. This ID is added to the data structure associated with each detected blob.

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 high-level vision 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.

V. Inter-Scene Vision

Inter-scene tracking is the fusion of tracking data received from multiple individual sensors to allow the tracking of vehicles as they move among nodes without overlapping fields of view. This type of tracking is impossible with data from singles nodes working independently. The input of the inter-scene tracking data is the calculated output from the intra-scene tracking modules. No raw video data is used at the inter-scene level. The output is the history of movement and a unique ID of every vehicle which has moved among multiple field-of-view with a certainty that passes a predetermined threshold. The steps in inter-scene vision include: feature vector calculation, graph creation, and graph matching.

A. Feature Vector Calculation

A feature vector is constructed for the set of feature data describing each blob. This feature data is reported to the inter-scene tracking module by multiple, independent intra-scene tracking modules in an asynchronous manner. That is, the data rate from each intra-scene node may vary depending on scene complexity, processing power, and network transmission bottlenecks. The feature vectors are used in subsequent processing steps, primarily in the formation of "platoon graphs" and the matching of moving objects which have traveled among two or more camera views. The vectors are easily modifiable to adapt to various applications or environments. In the demonstration implementation, each vector includes the following information: blob size, blob color (hue, saturation, and intensity), blob aspect ratio. This vector of features appears sufficient to discriminate moving objects from each other with sufficient accuracy to demonstrate effective inter-scene tracking in a variety of scenarios. The method of actually calculating the likelihood of a match is explained in the following sections.

B. Graph Creation

A graph data structure is used to model the moving objects in a scene, at any given moment. The aspects being modeled are the visual characteristics of all the moving objects, as well as their spatial orientation.

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, as is typical in the demonstration implementation associated with this paper.

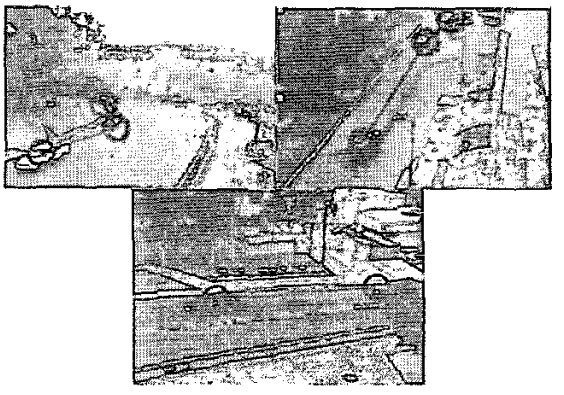

Fig. 4. Snapshots from the three interior-campus video cameras used in the application. The top two images show the same three-vehicle platoon at two different times in two different locations. The bottom image shows the trailing two vehicles from the platoon at a third time and location. Re-identification was successful in these cases. The approximate graph-matching technique used allows the two-vehicle platoon to be matched with the above three-car platoons.

C. Graph Matching

Platoon matching is the core of the inter-scene tracking algorithm. It performs the "matching" of moving objects which travel among camera sites, and allows for a unique, environment-wide ID to be assigned to a vehicle which is successfully identified via the platoon-matching system.

The platoon-matcher 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. 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 results from common intra-scene 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 insertion.* 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: $\lambda(a \rightarrow b) = (x_a - x_b)'C^{-1}(x_a - x_b)$ the Mahalanobis distance between the color features of the two nodes.
- Node deletion: $\lambda(a \rightarrow \Lambda) = |x_a|$, 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 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 [12]. Calculating the minimum-cost mapping can be cast as a state-space search problem, the details of which are shown in [12].

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 [11]. 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.

If the mapping with the lowest cost passes a pre-defined matching threshold value, the data graph associated with the pattern graph is assumed to be a true match to the pattern graph. However, the graphs are not necessarily identical, due to the node and edge operations allowed in the matching algorithm.

VI. Implementation and Software Infrastructure

The software implementation used to demonstrate the above algorithms was coded in Java, with some use of the OpenCV library to perform computationally expensive computer vision tasks. While Java is not, typically, an ideal language for computer vision tasks, it was used in this case because of its strength in easily producing distributed code, and its platform independence.

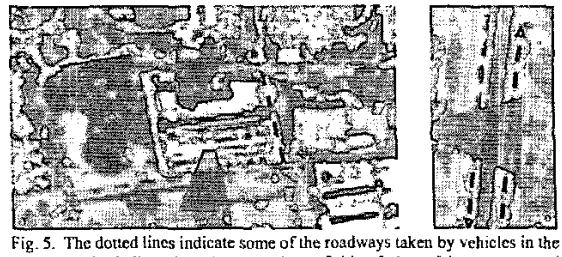

Fig. 5. The dotted lines indicate some of the roadways taken by vehicles in the test data. Also indicated are the approximate fields-of-view of the cameras used to capture the test data . Both images are to the same scale. The freeway is eight lanes wide, four in each direction, and the roadways in the image on the left are one lane in each direction.

The software infrastructure allows distributed computing, on two levels: sensor site processing, and high-level vision processing. A small self-contained video server at each sensor site performs data compression, and serves video to an arbitrary number of clients. This minimizes the bandwidth necessary for the transmission of video streams.

The next level of distributed computing occurs in a high-level vision processing manager. This manager employs the resources of an arbitrary number of general-purpose computers in order to successfully maintain real-time video processing. Java IDL, Sun Java's implementation of CORBA, is used to distribute vision processing among multiple physical machines. Figure 4 depicts an outline of the organization of the software infrastructure.

VII. Data and Results

Figure 2 shows some snapshots from the four cameras nodes used in the demonstration application. These are four scenes from the sensorized college campus used as a test-bed.

Figure 4, below, shows the application interface, and includes a 2D visualization of the campus area, with the locations of the camera nodes with approximate fields of view.

Data collected includes 5-minute segments of video simultaneously recorded from all four views. Three sets of video were recorded, each at a different time of day, to collect a wide range of conditions and traffic densities. Although the data is "canned," the application is run as if the data were real-time – the data does not "wait" for the application.

There are three performance metrics calculated. The first metric tests the performance of the software infrastructure's ability to distribute computational load to maintain effective data processing rates as additional camera nodes are added. In this example, only three physical computers are used, however the data suggests that distributed computing significantly improves the data processing rate over what a single machine could perform. The frames-per-second ratings below are in average of the frame processing rates of all processed video streams. The performance results are:

- 1 node: 14 frames per second (fps)
- 2 nodes: 14 fps
- 3 nodes: 13 fps
- 4 nodes: 8 fps

As expected, the frame rate drops significantly when multiple video streams have to processed by one machine, which is the case when the number processed video streams exceeds the number of available machines.

The second metric test the re-identification accuracy using a data set created by recording staged driving maneuvers in very light traffic. The results are shown as percentages of successful re-identification of moving objects traveling between sensor nodes vs. a hand-counted ground truth. The results show considerable success in re-identifying the platoons. The failure cases are largely due to false positives in one case where two cars have very similar color. This problem could be solved by using additional feature, such as vehicle classification, in discriminating between vehicles.

The results for this data set are presented in Table 1.

Table 1 – Inter-Scene Tracking Results for Test Data

	1 (TURNAROUND)	2 (SHUTTLE STOP)	3 (SERF)	4 (FREEWAY)
1	100% (7)	100% (7)	86% (7)	100% (1)
2		86% (7)	71% (7)	N/A
3			71% (7)	N/A
4				33% (4)

Table 1, "Laboratory" re-identification accuracy rates. The numbers in parentheses are the numbers of test cases used in calculating the success rate. The results are bidirectional, meaning that there is no distinction between a vehicle traveling from node 2 to node 3, or node 3 to node 2. The numbers are vs. a set hand-counted ground truth test cases combined from several video capture sessions. The freeway node in table location 4,4 indicate re-identification at the same location.

The third metric tests the performance of the inter-scene tracking algorithm in real-world testing., with the system being run in real-world conditions. These results are shown in Table 2. While the tracking results may not appear to be "good," it should be noted that they are comparable to results obtained by other modalities, such as inductive loops [10], that wide-area tracking is a difficult problem. Problems introduced by real-world operation are problems typical to real-world computer vision, and include the introduction of noise from high traffic density, occlusions, and erratic vehicle behavior in the on-campus environment. An example of erratic behavior including stopping near the campus shuttle turnaround point to pick up passengers, a scenario which the system is not yet able to handle. Results in the freeway data are generally lower because the freeway scene contains large numbers of vehicles moving at high speed, and is a difficult scene for computer vision, particularly for re-identifying vehicles since many vehicles look the same even when trying to manually re-identify vehicles in recorded video.

Table 2 – "Real World" Inter-Scene Tracking Results

	1 (TURNAROUND)	2 (SHUTTLE STOP)	3 (SERF)	4 (FREEWAY))
1		76% (36)	54% (43)	20% (12)
2			67% (28)	13% (10)
3				23% (13)
4				

Table 2, Live re-identification accuracy rates. The reduced accuracy rates are largely caused by much higher traffic density, which causes occlusion, and increases the likelihood of false positive graph matches. In these results, re-identification at the same location was not considered.

VIII. Discussion

One of the primary problems with the recent introduction of large numbers of video cameras to transportation environments is the inability of human operators, or classical computer vision systems, to handle the enormous quantities of data available. Typically, cameras are viewed one-at-a-time in a serial manner, with only instantaneous information being available to the user. Distributed real-time vision, together with effective data visualization, allows new applications to be developed which enables the display of new types of real-time information which may give a more complete, comprehensive view of the state of the traffic environment. Such information includes site-to-site trip time, system-wide traffic flow statistics, and propagation effects of traffic incidents.

References

[1] S. Santini, "Very Low Rate Video Processing," *Proceedings of SPIE Vol. 4311 Internet Imagine II.*, San Jose, Jan 2001

[2] http://video.dot.ca.gov/

[3] 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*, 1996, Washington, DC, USA, 8-11 Dec. 1996 p.305-12.

[4] Boyd, J.E.; Hunter, E.; Kelly, P.H.; Li-Cheng Tai; Phillips, C.B.; Jain, R.C. "MPI-Video infrastructure for dynamic environments, " *Proceedings. IEEE International Conference on Multimedia Computing and Systems*, Los Alamitos, CA, USA, 1998. p.249-54.

[5] M. Trivedi, I. Mikic, G. Kogut, "Distributed Video Networks for Incident Detection and Management," *IEEE Conference on Intelligent Transportation Systems*, Dearborn, Michigan, October 2000.

[6] M. Trivedi, A. Prati, G. Kogut, "Distributed Interactive Video Arrays for Event Based Analysis of Incidents," Submitted to the 5th International IEEE Conference on Intelligent Transportation Systems, Singapore, September 3-6, 2002.

[7] http://www.intel.com/research/mrl/research/opencv/

[8] http://java.sun.com/products/java-media/jmf/

[9] 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

[10] Zeng, N.; Crisman, J.D., "Vehicle matching using color," *Proc. ITSC'97*, p. 206-11.

[11] Coifman, B., Cassidy, M., Vehicle Reidentification and Travel Time Measurement on Congested Freeways, Transportation Research-B (submitted)

[12] 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 Intelligence*, 1994. p.390-6.