

# Real-time Wide Area Tracking: Hardware and Software Infrastructure

Gregory T. Kogut, Mohan M. Trivedi

Computer Vision and Robotics Research Laboratory, University of California, San Diego

## Abstract

*This paper presents a hardware and software infrastructure capable of performing real-time tracking of vehicles moving through a large environment with sparse sensor coverage. The system is designed to be scalable and modular at both the hardware and software level. It supports an arbitrary number of vision sensors, and employs a distributed computing architecture that allows for the simultaneous real-time processing and fusion of data from an arbitrary number of sensors using only general purpose computers linked to each other and the sensors via Ethernet. An example demonstration is shown in an application which fuses the data from four widely-spread cameras on a college campus. The output of the application is a top-down 2D visualization of the vehicular traffic moving among the parts of the campus which have sensor coverage.*

*Index terms—real-time, tracking, sensor fusion, distributed*

## INTRODUCTION

This paper represents the next iteration of research of the subject presented by the same authors at ITS 2001 [1], namely maintaining the identity of vehicles as they travel through an environment with sparse vision sensor coverage. While the previous paper presented an algorithm and example data which was post-processed offline, this paper presents a hardware and software architecture capable of real-time execution of the algorithm for an arbitrarily large environment containing an arbitrary number of sensors.

The tracking of moving objects from video data is generally a computationally expensive task. While there have been significant increases in computational power and the availability of broadband networking in recent years, most computer vision applications which use software to perform image processing still test the limits of general-purpose computers. Few intensive computer vision applications can simultaneously process more than 2-4 full-size video streams, depending on the nature and complexity of the vision tasks.

The transmission of multiple high quality digital streams also pushes the limits of many common IP-based networking systems. A digital video stream, depending on the size, quality and degree and type of compression, can occupy anywhere from 0.5-10 Mbps. General purpose networks can quickly become a bottleneck for applications which require the processing of large numbers of high quality video streams.

These problems are solved with a distributed computing system, which distributes the computational load, and reduces the likelihood of network bottlenecks. Both the software and hardware infrastructures are designed to be as platform and language independent as possible, so that the system can be implemented on a wide variety of general-purpose computing hardware, and can be easily updated, modified, or enlarged.

The infrastructure are tested with an application which performs real-time tracking on two levels, using four full-size video streams as input. The two levels of tracking are:

1. Intra-Scene tracking. This is the tracking of vehicles within a single camera view.
2. Inter-Scene tracking. This is the tracking of vehicles as they move among camera views. The camera views need not overlap, or be in close proximity, distinguishing this from the camera handoff problem

Because the simultaneous viewing of multiple video streams is as difficult for humans as for computers, the output of the applications is displayed with a top-down 2D visualization of the environment, which fuses the output from both levels of tracking into one display.

## RELATED WORK

Research in vision-based tracking systems which use multiple cameras and processing nodes has only recently become active. This is largely due to the expense, 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 large-scale 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 [2]. 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 [3]. 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 [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 of research. This system forms the basis for the architecture used in this thesis.

There have also been significant advances in vision software architecture during the past decade as well. Early computer vision implementations were specialized programs, and required extensive development and maintenance. However, in recent years, ideas from the software architecture community have been introduced into computer vision software implementations. These ideas, such as modularity, reusability, and scalability have been incorporated into computer vision projects and tools, and have served to improve the speed and efficiency of research. In 1993 Godwin and Mitchell presented a modular, object-oriented system for general computer vision architectures [5]. In 2000 Graf and Knoll proposed an agent-based system for distributed processing in vision architecture [6]. These projects introduced the latest ideas in software engineering to computer vision research. A number of powerful software tools have also emerged. Intel's Open CV is an open source project that contains a wide array of vision algorithms, and is a great resource to vision research [7]. Sun's Java platform also offers powerful multimedia and distributed computing capabilities, and is platform and operating system independent. Both of these tools are used in this paper.

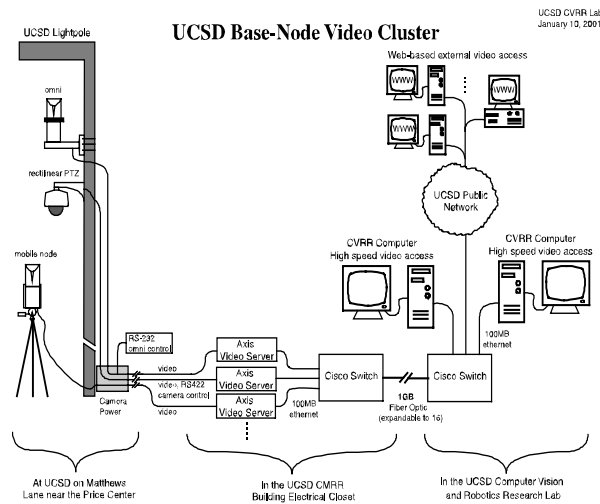
## HARDWARE INFRASTRUCTURE

The hardware infrastructure was designed to embody several properties:

- ❑ Modularity
- ❑ Scalability
- ❑ Use of “off-the-shelf,” general-purpose components
- ❑ Capable of distribution across a wide area
- ❑ Use of a hard-wired or wireless links
- ❑ “openness,” or the use of open standards in software, networking protocols, and data compression

The system manager, which distributes the processing load among multiple machines provides a similar interface which provides information about the environment wide state of moving vehicles, such as the track of a vehicle which has moved among multiple sensor sites.

These properties allow for the infrastructure to be easily upgraded, modified, or enlarged., and to be easily accessible with common general-purpose computers, and largely operating system independent.



**Figure 1: ITS Video Network, showing the connection of a various sensor types to a standard IP network.**

**Networking.** The networking components of the architecture are strictly standard Ethernet-based IP networks. While some sections of the networking architecture are dedicated to the vision architecture, others simply take advantage of the existing campus Internet infrastructure. The dedicated networks are used simply to guarantee a high data rate in areas with a potential for network bottlenecks.

**Sensors.** The architecture support most types of vision sensors, and can potentially handle the addition of other sensor modalities, such as IR or audio. Currently the network includes Pelco Spectra II pan-tilt-zoom cameras domes and omnidirectional vision sensors. There are 9 sensors permanently deployed, plus two mobile sensors with wireless video links which can be brought online in minutes at most locations on campus.

**Video Servers.** Video servers exist at each sensor site for several reasons. They compress video before any network transmission, to reduce network load. They also eliminate the need for a heavy-duty centralized video server. Video can be transmitted point-to-point, or be multi-cast to various users straight form the sensor node. Currently, Axis 2401 video servers are used. But the network design is not dependent on any specific type of server.

**Computing Platforms.** The hardware infrastructure is not dependent on a specific platform. Currently, a variety of standard Intel-based PCs are used to provide computing power. The details of the software infrastructure are included below.

**SOFTWARE INFRASTRUCTURE**

The software infrastructure was designed to embody properties similar to that of the hardware infrastructure:

- ❑ modularity
- ❑ scalability
- ❑ use of open standards
- ❑ platform and language independence
- ❑ distributed computation
- ❑ runs on general purpose computers

While these characteristics exist in such libraries such as Intel’s OpenCV library, the software infrastructure described here refers to a vision architecture at a high level, not to low-level image processing. The infrastructure is also specifically designed to handle the simultaneous computation of a large number of video streams.

The software architecture is based on the CORBA (Common Object Request Broker Architecture). CORBA provides a large degree of platform, network, and language independence.

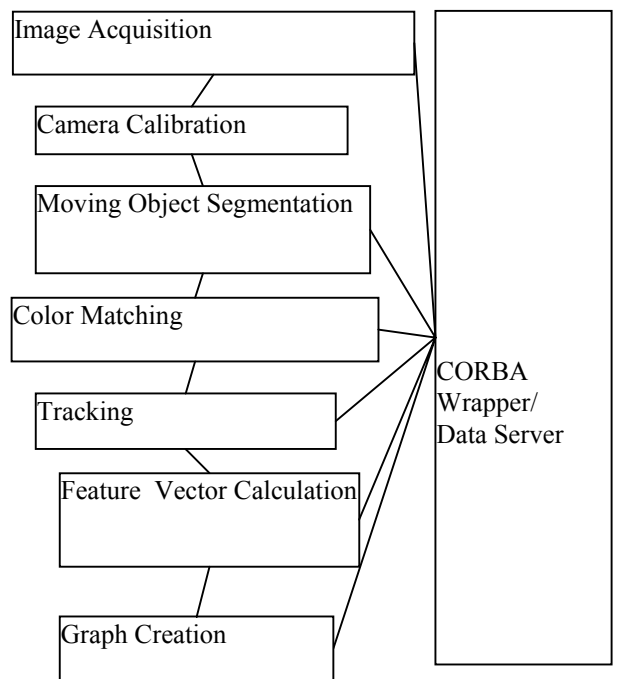
The primary component of the CORBA architecture is the ORB, or Object Request Broker. The ORB locates objects, makes connections, and delivers data. The ORB, in this case, allows all of the image processing and vision routines to be described solely by their interfaces. Their physical location, details of implementation, and the platform they’re running on are transparent to the final application. The interface, in pseudo-code provided by each sensor node in the example implementation give in this paper is:

- ❑ GetRawData()
  - Returns video stream
- ❑ GetNumberofMovingVehicles()
  - Returns number of moving vehicles
- ❑ GetVehicles
  - Returns location and feature information (velocity, color) for each moving vehicle
- ❑ GetPlatoonInfo
  - Returns a data structure which describes the platoon structure, if any, of the vehicles in the scene

The methods for implementing these functions are independent of the interface, so that new algorithms can be implemented and tested without requiring change to the overall design of the software architecture.

It should be noted that most of the code in this project was written in Java, which is not the optimal language for computer vision from a performance perspective. However the fact that real-time performance was achieved without optimized code reflects the underlying power of distributed computing. The OpenCV library was used to implement some of the more intensive image processing routines [7].

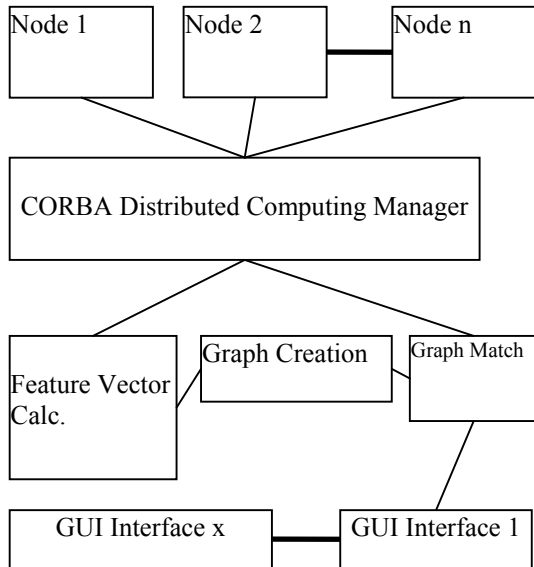
**Intra-scene Tracking.** The term intra-scene tracking, in this paper, refers to the tracking of vehicles within a single camera’s field of view. This type of tracking has been widely researched, with robust solutions to a variety of tracking applications being widely known.



**Figure 2 Dataflow within an intra-scene processing node.**

**Inter-scene Tracking.**

Inter-scene tracking is the tracking of vehicles as they travel among camera nodes. Because the infrastructure design allows for sparse sensor coverage, or non-continuous sensor coverage of the environment, inter-scene tracking is a nontrivial task.. Inter-scene tracking requires the fusion of data processed from multiple individual intra-scene processing nodes.



**Figure 3 Inter-Scene Processing Data Flow**

## VISUALIZATION

The visualization of multiple video streams is a difficult task for humans. Therefore, in a wide-area tracking system with multiple sensors, information may be more readily available with the aid of some form of data visualization, such as the abstraction of fused data into a format that is readily interpreted by users.

The example visualization application shown in this paper provides multiple levels of tracking information in the same interface, avoiding the need to switch between multiple views. The interface is a 2K top-down map of the UCSD campus, with overlaid sensor site and tracking information. The information received from multiple nodes is fused so that the interface need not change with the addition or subtraction of sensors. To the side of the map, real-time video may be seen from any of the sensor sites. Intra-scene information may be overlaid on top of the raw video. Such information includes:

- ❑ Moving object segmentation results
- ❑ Vehicle velocities
- ❑ Visualization of platoon detection
- ❑ Basic vehicle classification (based on aspect ratio)



**Figure 4 A screen snapshot of the tracking application. A 2D top-down map of the UCSD campus is overlaid with the fused tracking information from each sensor node. Note that the there are good tracking results even though this snapshot was taken around midnight. Ambient and head lights provide enough light such that there is little drop in tracking performance from daylight. The yellow arrow indicates vehicle re-identification sensor node. A vehicle has left the campus and taken the freeway onramp.**

**Figure 5 Expanded portion of the interface. The ovals indicate areas of sensor coverage, and the numbers indicate the number of vehicles currently tracked in each area. Nodes of interest can be selected with the mouse to provide details intra-scene tracking information. The data from each sensor site is being processed by a separate physical computer, and being communicated to the visualization via an Object Request Broker**

## COMPUTER VISION ALGORITHMS

The algorithms implemented to test the hardware and software infrastructures are nearly identical to those described by the authors of this paper in 2001 [1]. The details of the algorithm are therefore omitted from this paper, which focuses on the details of large-scale, real-time vision processing.

## RESULTS

There are two types of results calculated to measure the performance of this system: tracking accuracy, and frame rate.

The accuracy of inter-scene tracking reflects the performance of the high-level computer vision after the fusion of data from individual sensors. The results in this case are reported as the accuracy rate of vehicle re-identification between each pair of sensor nodes. While the system works live, the test data was a recorded video for the ease and accuracy of manually calculating ground truth.

	1	2	3	4
1		73% (21)	52% (32)	19% (9)
2			63% (19)	14% (7)
3				25% (12)
4 (freeway)				

**Figure 6 Live re-identification accuracy rates. The number in parenthesis is the number 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.**

It should be noted that these results are comparable to re-identification results from inductive loop sensors, as reported in [1].

The application performance is reported as the average frame rates at which the visualization application displays data. This result is given in an incremental form, to demonstrate that the addition of additional sensors does not appreciably slow the performance of the final application. In this demonstration, the total computational load, including the data visualization, is spread across three machines:

- ❑ 1 sensor: 16 frames/sec.
- ❑ 2 sensors: 16 frames/sec
- ❑ 3 sensors: 16 frames/sec
- ❑ 4 sensors 13 frames/sec

The drop in frame rate with the addition of the fourth sensor is a result of the assignment of two sensors to one machine. With the addition of more machines, an arbitrary number of cameras could be added.

The frame-rates could also greatly increase with the use of optimized code for the image processing.

## DISCUSSION

The results indicate that the hardware and software infrastructure have achieved their goal of scalability and modularity. Additional sensor may be added without modification of the underlying infrastructures, and with little modification to existing code. Such infrastructures will be necessary as computer vision becomes a ubiquitous part of such activities as wide area tracking or surveillance.

The results of the vehicle re-identification could probably be improved with the addition of other sources of information, such as IR sensors, as well as a more rigorous study of the vehicle properties of vehicles which can be extracted from video to aid in uniquely identifying vehicles.

## FUTURE WORK

Future work includes the inclusion of more sensors, and type of sensors. While the example in this paper included the input from four cameras, the infrastructure can support an arbitrary number of cameras. Also, the system need not be restricted to vision sensors. IR sensors, microphones, or ODVS sensor can also be included to increase the accuracy of tracking. Also, applications other than tracking, such as classification of vehicles, or behavior recognition can be developed with little modification to the underlying infrastructures.

## 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.

<sup>1</sup> Gregory T. Kogut and Mohan M. Trivedi are with the Electrical and Computer Engineering Department, University of California-San Diego, 92093-0407, USA.

## References

1. 1. Kogut, G T and Trivedi, M., "Maintaining the Identity of Multiple Vehicles as They Travel Through a Video Network." Proc. IEEE Conference on Intelligent Transportation Systems, 2001
2. 2. 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
3. 3. 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.

- 
4. 4. M. Trivedi, I. Mikic, G. Kogut, "Distributed Video Networks for Incident Detection and Management," IEEE Conference on Intelligent Transportation Systems, Dearborn, Michigan, October 2000
  5. 5. Godwin, L.E.; Mitchell, O.R. "Design and implementation of a modular object-oriented integration architecture for real-time systems." Proceeding of the International Robots and Vision Automation Conference, Ann Arbor, MI, USA: Robotic Ind. Assoc, 1993. p.5/47-59.
  6. 6. Graf, T.; Knoll, A. "A multi-agent system architecture for distributed computer vision." International Journal on Artificial Intelligence Tools, vol.9, (no.2), World Scientific, June 2000. p.305-19.
  7. 7. <http://www.intel.com/research/mrl/research/opencv/overview.html>