REAL-TIME DISTRIBUTED TRACKING

Wayne Wolf[1], Senem Velipasalar[2], Jason Schlessman[1], Cheng-Yao Chen[1], Chang-Hong Lin[1]
[1]*Department of Electrical Engineering*
Princeton University
[2]*Department of Electrical Engineering*
University of Nebraska—Lincoln

Abstract

Distributed smart cameras use distributed computing architectures to analyze imagery from physically distributed cameras. Performing real-time distributed analysis of video introduces substantial new challenges, but also provides substantial benefits over server-based approaches. This work describes some of the algorithms and architectures we have developed for tracking using distributed smart camera systems, including fault-tolerance, synchronization, and multi-band fusion.

index terms: distributed computing, tracking

1 Introduction

Computer vision uses the term **distributed camera** to describe a set of physically distributed cameras. Occlusion is a key problem in tracking. Multiple camera systems were introduced to reduce occlusion problems. We can distinguish between **static occlusion** caused by a fixed object and **dynamic occlusion** caused by several moving objects. Static occlusion can be tackled by some amount of pre-processing, but dynamic occlusion requires low-latency analysis.

The term *distributed camera* says nothing about the computer architecture used to process the imagery, but most research in distributed cameras has relied on a central server. But server-based video analysis does not scale to the large number of cameras and large volumes of data required in realistic installations; servers also do not scale well under real-time requirements [Wol07]. Realistic multi-camera installations may require hundreds of cameras; these large camera installations cannot be analyzed by a centralized server.

Bove's group at the MIT Media Lab [Bov03] conducted early research on distributed computing architectures for multiple cameras. More recent work on distributed smart cameras includes Rinner's group at TU Graz [Bra06], Radke's group at RPI [Dev06], and Aghajan's group at Stanford [Lee06]. Our own work in smart cameras [Oze02] that perform video analysis in real time led us to develop **distributed smart camera** systems.

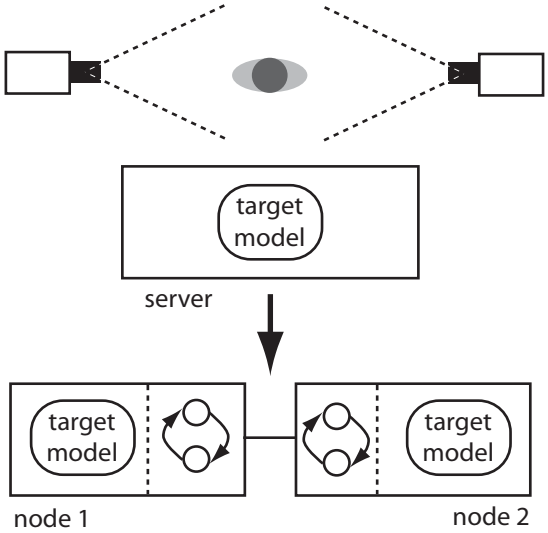

FIGURE 1. A server-based distributed camera algorithm can be transformed into a distributed system.

Our first distributed smart camera [Lin04] performed gesture recognition using a peer-to-peer architecture. Each smart camera performs some processing locally. Smart cameras then send or receiver an abstract intermediate representation of the subject. Those intermediate representations are combined to recognize the gestures. As the subject moves about the scene, smart cameras always perform their local processing, but the node that performs the final gesture recognition may move about the network, as represented by a token.

Energy and power consumption are critical metrics for distributed smart camera systems. Even if smart camera nodes will not be powered by batteries, power consumption determines heat dissipation as well as system cost.

Using distributed algorithms causes us to both be concerned with and find opportunities for fault-tolerant computing. By

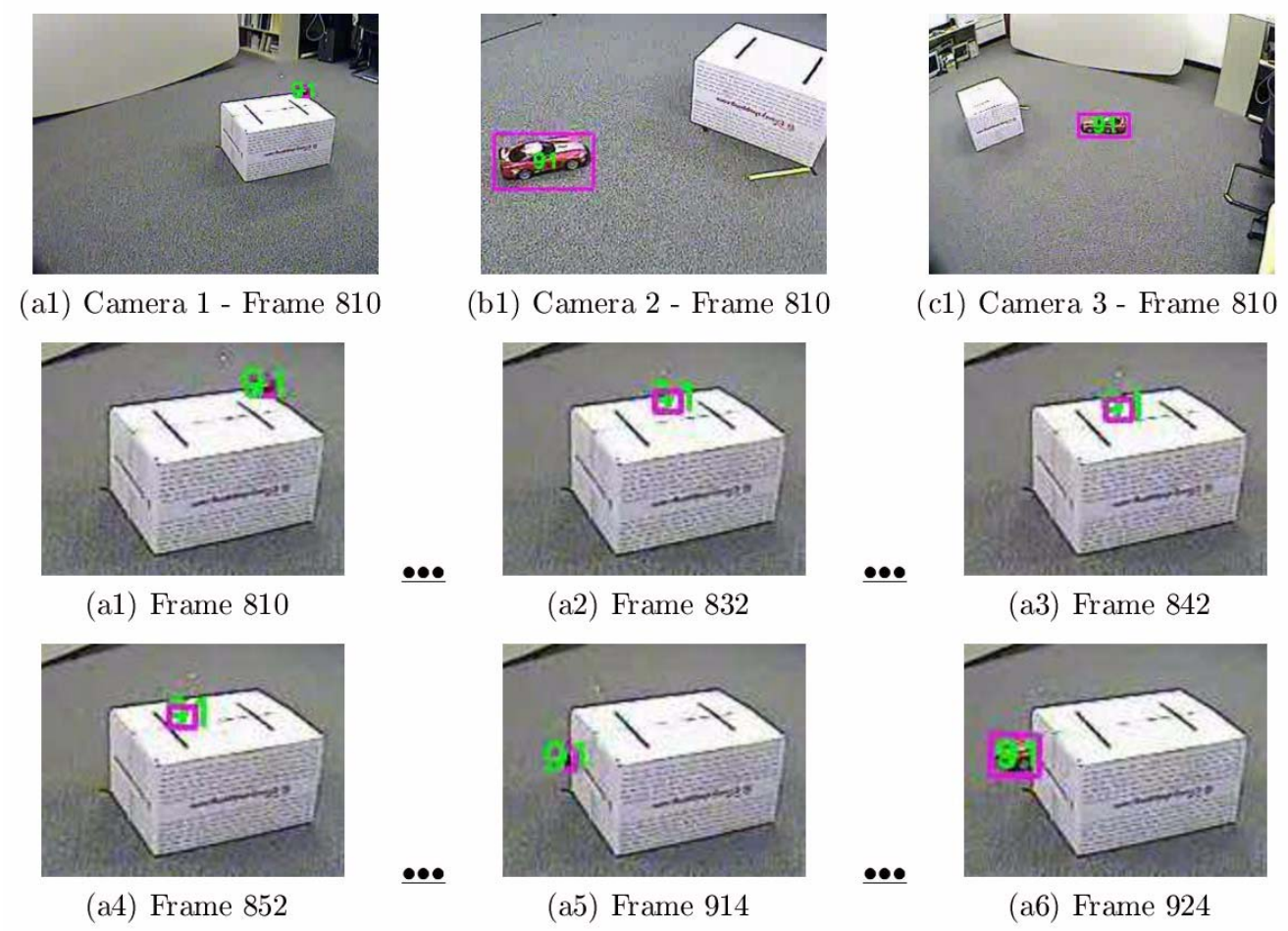

FIGURE 2. Camera 1 tracks an occluded object by communicating with other cameras.

distributing information about tracking operations across the network, we can improve the system's resistance to failures, whether inadvertent or malicious.

This paper summarizes our approach to tracking using peer-to-peer distributed smart cameras. We developed SCCS, the Scalable Clustered Camera System, to demonstrate our ideas on distributed smart cameras for tracking [Vel06A]. SCCS allows us to demonstrate an interdisciplinary approach to multi-camera systems.

This paper will describe our approach to tracking in several steps. Because protocols are central to distributed systems, we will start by describing our protocol for tracking. Next, we will describe the middleware that dynamically controls the operation of the distributed system. We will then consider image processing challenges in real-time distributed tracking.

2 Protocols for Real-Time Tracking

Distributed algorithms are controlled by protocols that govern the communication between the computing nodes. As illustrated in Figure 1, we need to break our vision algorithm into steps that can be performed on distributed processors. This process requires inserting protocols between steps and allocating data to the nodes. When distributing a vision algorithm, data allocation and their associated computations are particularly important because of the large volumes of data used in the early stages of analysis.

A particular goal for SCCS was to make the system more fault-tolerant. As illustrated in Figure 1, we replicated data about targets around the network—each node keeps track of all of the targets in its field of view, even if the target is occluded at any particular time. If each target were tracked by a single node, a node failure would cause the system to lose all information about that node's targets. Failures at

single nodes do not necessarily cause us to lose all information about its targets.

The intersections between the fields-of-view of the cameras are determined before normal operations begin. These intersecting fields-of-view determine the possible required interactions between smart camera nodes, since two cameras that have non-overlapping fields of view cannot track the same subject at the same time. But the actual interactions are determined at run time based upon the positions of the targets.

For each target, a smart camera node communicates with all other nodes whose fields-of-view include that target. The nodes trade information on the position of the target every *n* frames, where *n* is determined by the protocol designer or user. If the target is occluded at one camera, it receives an update on the target's position during this trading step.

Figure 2 shows several frames from SCCS tracking a car in a scene with a static occlusion opportunity. The top part of the figure shows the view of the scene from each of the three cameras; the lower frames are all from camera 1. As the target moves behind the box and becomes occluded, camera 1 continues to know the target's location thanks to its communication with the other camera nodes.

3 Software Architectures for Distributed Smart Cameras

Distributed computing protocols may be implemented *ad hoc* using specialized code but most modern distributed systems make use of **middleware** to implement basic distributed computing functions. Middleware is used because many resource allocation decision cannot be made statically. Middleware dynamically manages requests based upon activity and resource management policies.

Tracking systems present rapidly changing situations and resource management decisions. As targets move about the scene, they move in and out of fields-of-view of the cameras. Each change in the visibility state of the target must cause the network to change its patterns of computation and communication. The smart cameras that need to communicate may not be nearby either spatially (they may be across the scene from each other) or in network distance.

Our SCCS protocol made use of MPI (http://www-unix.mcs.anl.gov/mpi/), a widely-used middleware system for scientific computation. MPI provided basic facilities for sharing data across nodes in the network; the specific protocol was implemented in a layer that made use of MPI primitives. MPI was not designed for real-time, embedded applications. However, our experience with MPI has helped us better understand the requriements on real-time middleware in general and vision-oriented middleware in particular. We are developing a new, more specialized middleware architecture.

Many research groups have studied spatial calibration of distributed cameras, but only a few groups have studied synchronization or temporal calibration. The oscillators used to generate frames in camers are not accurate enough to remain synchronized for any useful period. Analog synchronization methods, such as genlock, are very poorly suited to large, real-world installation. We studied traditional distributed algorithms for timekeeping [Lin05]. We also developed an image-processing-based algorithm to synchronize cameras [Vel05]. Practical systems will probably make use of both types of algorithms.

Sensor network research has tackled many issues of interest to distributed smart cameras: synchronization, network membership, routing, etc. SCCS is not built upon a sensor network middleware system, but such systems could provide a range of useful services. Because vision systems can transmit large volumes of data, some specialized services may be required.

4 Distributed Image Processing

Distributed smart cameras present several interesting image processing challenges. Perhaps the most basic issue is the hierarchy of representations used to describe the image. Many image processing algorithms use intermediate representation, but because they are used only internally, the characteristics of those representations often receive only minimal attention. However, intermediate representations in distributed systems have profound consequences on performance, real-time behavior, and energy/power consumption.

SCCS uses a two-part target model: a color histogram and Bhattacharya distance. This model is small enough to fit into a single 256-byte packet. Other groups have developed sophisticated multi-view target models to handle, which could be incorporated here. An interesting topic for future research is to study how partially-processed imagery could be used as a distributed target model.

We use the target model as our intermediate representation. This very abstract representation is reasonable given our simple model for data replication. However, a more pixel-oriented model could be useful for some types of vision algorithms.

Tracking can also benefit from information from several bands. Figure 3 shows some results of our target model that combines information from the thermal and visible bands [Che06]. This background elimination algorithm combines information from the two bands to create an improved estimate of the foreground. It uses a particle filter with a mix-

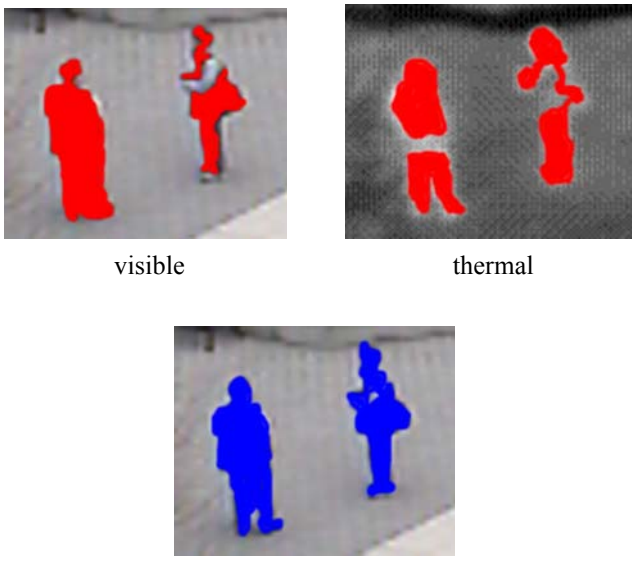

FIGURE 3. Multi-spectral foreground fusion.

ture-of-Gaussians distribution for each foreground object. The V channel is generated by first adjusting for lighting, then fusing visible and thermal channel information with confidence weightings. Multi-band image analysis helps us estimate lighting sources and eliminate shadows; it also helps us handle target fusing and separation during dynamic occlusion.

5 Summary

Realistic multi-camera tracking systems are embedded computing systems that must operate in real time, at low power levels, and within cost constraints. These constraints inevitably lead us toward distributed computing architectures. We cannot completely divorce the consideration of image processing algorithms from the hardware and soft-awre architetures of the distributed platform used for tracking. We must use algorithms that provide compact yet powerful abstract representations of the images and that can be processed with low latencies. However, by viewing the algorithm design process as the insertion of protocols into a server-based algorithm, we can make the process more manageable. We believe that this approach is widely applicable and that distributed smart camera systems will be developed for many applications.

Acknowledgments

This work was sponsored by the National Science Foundation under award 0325119 and by the Army Research Office under contract W911NF-05-12-0480.

References

[Bov03] V. Michael Bove, “Media processing ecologies,” in *Proceedings, ITRE 2003*.

[Bra06] Michael Bramberger, Andreas Doblander, Arnold Maier, Bernhard Rinner, Helmut Schwabach. *Distributed Embedded Smart Cameras for Surveillance Applications*. IEEE Computer 39(2) pages 68-75, February 2006.

[Dev06] Dhanya Devarajan, Richard J. Radke, and Haeyong Chung, *Distributed Metric Calibration of Ad-Hoc Camera Networks. ACM Transactions on Sensor Networks*, Vol. 2, No. 3, pp. 380-403, August 2006.

[Che06] C. -Y. Chen, and W. Wolf, “Background Modeling and Object Tracking Using Multi-Spectral Sensors,” *ACM Workshop on Video Surveillance and Sensor Networks*, 2006.

[Lee06] H. Lee and H. Aghajan, “Vision-Enabled Node Localization in Wireless Sensor Networks,” *COGnitive systems with Interactive Sensors (COGIS)*, March 2006.

[Lin04] Chang Hong Lin, Tiehan Lv, Wayne Wolf, and I. Burak Ozer, “A peer-to-peer architecture for distributed real-time gesture recognition,” in *Proceedings, International Conference on Multimedia and Exhibition*, IEEE, 2004, vol. 1, pp. 27-30.

[Lin05] C. H. Lin and W. Wolf, “A Case Study in Clock Synchronization for Distributed Camera Systems”, *Proceedings of SPIE*, Vol. 5683, January 2005.

[Oze02] Wayne Wolf, Burak Ozer, and Tiehan Lv, “Smart cameras as embedded systems,” *IEEE Computer*, 35(9), September 2002, pp. 48-53.

[Vel05] Senem Velipasalar, Wayne Wolf, "Frame-Level Temporal Calibration of Video Sequences from Unsynchronized Cameras by Using Projective Invariants", *IEEE International Conference on Advanced Video and Signal Based Surveillance*, Como, Italy, September 15-16, 2005.

[Vel06A] Senem Velipasalar, Jason Schlessman, Cheng-Yao Chen, Wayne Wolf, and Jaswinder Pal Singh, “SCCS: a scalable clustered camera system for multiple object tracking communicating via message passing interface,” in *Proceedings, ICME 2006*, IEEE, 2006.

[Vel06B] Senem Velipasalar, Changhong Lin, Jason Schlessman, and Wayne Wolf, “Design and verification of communication protocols for peer-to-peer multimedia systems,” in *Proceedings, ICME 2006*, IEEE, 2006.

[Wol07] Wayne Wolf, *High Performance Embedded Computing*, Elsevier, 2007.