Distributed Real-Time Embedded Video Processing

Tiehan Lv
Wayne Wolf
Dept. of EE, Princeton University
Phone: (609) 258-1424
Fax: (609) 258-3745
Email: wolf@princeton.edu
Burak Ozer
Verificon Corp.

Abstract: The embedded systems group at Princeton University is building a distributed system for real-time analysis of video from multiple cameras. Most work in multiple-camera video systems relies on centralized processing. However, performing video computations at a central server has several disadvantages: it introduces latency that reduces the response time of the video system; it increases the amount of buffer memory required; and it consumes network bandwidth. These problems cause centralized video processing systems to not only provide lower performance but to use excess power as well. A deployable multi-camera video system must perform distributed computation, including computation near the camera as well as remote computations, in order to meet performance and power requirements.

Smart cameras combine sensing and computation to perform real-time image and video analysis. A smart camera can be used for many applications, including face recognition and tracking. We have developed a smart camera system [Wol02] that performs real-time gesture recognition. This system, which currently runs on a Trimedia TM-100 VLIW processor, classifies gestures such as *walking*, *standing*, *waving arms*. It currently runs at 25 frames/sec on the Trimedia processor. The application uses a number of standard vision algorithms as well as some improvements of our own; the details of the algorithms are not critical to the distributed system research we propose here.

However, real-time vision is very well suited to distributed system implementation. Using multiple cameras simplifies some important problems in video analysis. Occlusion causes many problems in vision; for example, when the subject turns such that only one arm can be seen from a single camera, the algorithms must infer that the arm exists in order to confirm that the subject in front of the camera is a person and not something else. When views are available from multiple cameras, the data can be fused to provide a global view of the subject that provides more complete information for higher-level analysis. Multiple cameras also allow us to replace mechanical panning and zooming with electronic panning and zooming. Electronically panned/zoomed cameras do not have inertia that affect tracking; they are also more reliable under harsh environmental conditions.

Processing in the distributed smart camera is inherently distributed. Sending raw frames to a central node for processing would consume enormous amounts of bandwidth; that bandwidth not only increases the cost of the network, it consumes considerable energy as well. Performing low-level processing at each camera, then sending abstractions of the frames to other nodes that can combine the results from several frames saves considerable power. Distributed processing is also probably the most effective approach to meeting real-time deadlines and low latency. The smart camera system will often be used as input to a decision-making system or person, so low latency is an important criteria. Using multiple nodes to process the data should considerably reduce the latency in obtaining the result.

We believe that a combination of design-time and run-time decisions are required for the successful deployment of video processing networks. Some run-time decisions are necessary because some characteristics of the system will not be known until run time, and those characteristics may change during operation of the network. Furthermore, the very configuration of the network may not be known at design time, as it may depend on the size of the area to be monitored, the physical constraints of installing nodes, etc. The overall hardware and software architecture of the network must be designed to operate well not just at a single design point, but across a range of possible configurations and operating decisions.

We are developing a middleware-based system for video analysis. The DVM (distributed video middleware) runs on top of the operating system (Linux) on each node. The DVM layer deals with video objects. The video objects may be described in any of several ways: regions of an image, curves that represent sections of the image, etc. The video objects represent the current state of analysis. The DVM layer manages the location of the video object data as it goes through the various processing stages. The DVM may be guided by design-time information provided by tools, such as the relative amounts of data and processing time required for various operations; it also makes use of run-time information that helps it make the best use of available resources.

A DVM-based architecture leverages COTS operating systems. It also helps to minimize the amount of work that is required to port a batch-oriented video application to a distributed platform. Relatively few video processing algorithm experts are also adept at distributed computing. The video object model allows them to map their video algorithms onto data structures and let the DVM layer manage the allocation and scheduling tasks required to run in distributed mode.

References

[Wol02] Wayne Wolf, Burak Ozer, and Tiehan Lv, “Smart cameras as embedded systems,” IEEE Computer, 35(9) September 2002, pp. 48-53.