ISR3: Communication and Data Storage for an Unmanned Ground Vehicle*

Bruce A. Draper Gökhan Kutlu Edward M. Riseman Allen R. Hanson
Computer Vision Laboratory, Dept. of Computer Science
University of Massachusetts, Amherst, MA 01003

Abstract

Computer vision researchers working in mobile robotics and other real-time domains are forced to confront issues not normally addressed in the computer vision literature. Among these are communication, or how to get data from one process to another, data storage and retrieval, primarily for transient, image-based data, and database management, for maps, object models and other permanent (typically 3D) data. This paper reviews efforts at CMU, SRI and UMass to build real-time computer vision systems for mobile robotics, and presents a new tool, called ISR3, for communication, data storage/retrieval and database management on the UMass Mobile Perception Laboratory (MPL), a NAVLAB-like autonomous vehicle.

1 Introduction

As computer vision technology matures, researchers are forced to divert some of their attention from subproblems (such as line extraction, model matching and shape from shading) to complete systems. Although many open subproblems remain, the need to demonstrate success on real-world applications and the growing belief that vision is inherently task-oriented [1, 2, 5] are spurring research into the practical issues of building working systems.

Much of this work has focused on supporting algorithm development and integration. In particular, both university researchers and commercial vendors have produced software environments that combine vision-oriented development tools with libraries of standard algorithms and representations.

None of these environments, however, are designed for real-time applications. Researchers who apply computer vision to applications such as mobile robotics need support for inter-process communication, short-term data storage and retrieval, and long-term data management. This paper reviews some of the approaches that have been tried for solving these problems, such as ISR [4], CODGER [9] and the Core Knowledge System (CKS; [10]). We then present a further development of ISR, called ISR3, for managing these issues on board an unmanned vehicle (MPL).

2 Real-time Integration

2.1 Issues

When algorithms developed off-line are to be combined for a real-time application such as an unmanned vehicle, several basic problems need to be solved. Among them are:

Communication. Data must be passed from one process—and often one processor—to the next. The latency of the communication channel can be critical. When data is shared among processes, concurrency control becomes important to protect the integrity of the data.

Data Storage/Retrieval. Vision algorithms often produce large amount of temporary data. A line extraction algorithm, for example, may produce hundreds (or thousands) of line segments from an image. Although not persistent, such data may be repeatedly accessed by other modules (e.g. line grouping or model matching algorithms), so the efficiency of data retrieval is critical.

Database Management. Maps, models and other forms of a-priori knowledge make up a permanent data base of information that visual algorithms repeatedly access and occasionally alter. Although less voluminous than the temporary data mentioned

*This work was supported by ARPA through U.S. Army TACOM under contract DAAE07-91-C-RO35, by ARPA through U.S. Army TEC under contract DACA76-92-C-0041, and by ARPA through Rome Labs under contract F30602-94-C-0042.

above, this data is persistent and must be managed over time by efficient storage and access mechanisms which are geared to the nature (e.g. spatial, temporal, 3D) of the data.

2.2 Current Technology

One of the first tools to support real-time image understanding was CODGER, a blackboard-style data store developed for CMU's NAVLAB [9]. The idea behind CODGER was that its "whiteboard" would serve as the central data store, with all other processes reading data from the white board and posting results to it. CODGER was also an information fusion mechanism that merged data by computing coordinate transformations and adjusting for time delays. Its authors called it a whiteboard, rather than a blackboard, because it had fixed-length message buffers, so that new messages forced old messaged to be purged. In effect, this was a primitive memory management system. Since CODGER's "whiteboard" was managed by a single process, it was an example of a (client-server model) data store.

Unfortunately, the central blackboard of CODGER became a bottleneck since all data messages passed from one process to another had to go through CODGER. This problem was exacerbated by CODGER's role as a data fusion mechanism: Kluge reports that CODGER spent most of its time computing coordinate transformations [6], slowing the central communication process down even further.

After CODGER was abandoned, a point-to-point communication package called EDDIE [12] was developed for NAVLAB. EDDIE was built on standard interprocess communication (IPC) protocols, and allowed processes to exchange instances of predefined message types. EDDIE also included simple control mechanisms, such as the ability to "wake up" a process when a message arrives, and a small geometric modeling and map system. EDDIE has since been replaced on board NAVLAB by TCX [11], a point-to-point communication package, and the Annotated Map System [12] for controlling vehicles behaviors based on location.

At about the same time as CODGER, researchers at SRI proposed a blackboard system called the Core Knowledge System (CKS; [10]). CKS, like CODGER, was intended as a blackboard-style data store for an autonomous vehicle. Unlike CODGER, however, it was not intended for data fusion. The philosophy behind CKS was that all messages (data) are merely hypotheses produced by knowledge sources, and that rather than fuse data, the blackboard should record which process proposed it and how much confidence it had. Memory management was handled by giving each knowledge source a memory budget, which it could allocate to messages it wrote or intended to read.

Another type of data store, called ISR [4], was developed at the University of Massachusetts. Unlike CODGER and CKS, ISR was designed as a tool for off-line algorithm development. As ISR's developers, we were interested in the knowledge-directed interpretation of outdoor scenes, and the control issues thereof. ISR was built to integrate many different visual procedures into a single system, and therefore to develop common data structures and mechanisms for passing data from one procedure to the next.

3 ISR3

Under the ARPA Unmanned Ground Vehicle (UGV) program, the University of Massachusetts was developing the Mobile Perception Laboratory (MPL), an autonomous vehicle similar to CMU's NAVLAB [11]. The goal for MPL was a system that integrates low-level behaviors, such as road following and obstacle detection, with high-level behaviors such as landmark-based navigation and map building. To support this project, we designed a prototype data store for real-time vision called ISR3, which was implemented on board MPL.

ISR (an acronym for intermediate symbolic representation) has been the name of a series of symbolic databases for vision developed at the University of Massachusetts [4]. The ISR databases reflect a belief that computer vision requires more than image-like arrays of numerical data; computer vision depends on symbolic representations of abstract image events such as regions, lines, and surfaces, and on mechanisms for efficiently accessing data objects under various types of constraints (such as spatial proximity). One version, ISR1.5, is now commercially available as part of KBVision [13], while the most recent version, ISR3, was used on-board MPL. Although each version of ISR is a refinement of its predecessor, they all assume that visual procedures operate on symbolic records, called *tokens*, or on groups of tokens, and that visual procedures manipulate tokens both for internal computations and for exchanging data with other procedures.

In particular, vision applications typically need temporary storage for large numbers of tokens, which they then access by name, feature value or spatial location. Most of these tokens exist for only a short duration, such as the time required to process one image or a short sequence of images, and should then be deallocated, although a few correspond to significant results (critical features, updated maps, etc.) and should persist over time. ISR3 is an in-memory database whose tokens are C++ class instances (unlike previous versions of the ISR) and that provides a library of functions for storing and retrieving tokens, and for token I/O.

As a rule, most visual algorithms operate on sets of data rather than on individual data instances. Matching algorithms, for example, compare a set of model data instances to image data instances. Therefore most of ISR3's storage and retrieval commands are in the form of set operations, such as a request to access all long, straight lines in the upper corner of an image. Special facilities for optimizing spatial retrieval over arbitrary data sets are also provided, as are macros for iterating over the instances of a set, and functions for taking the union, intersection and differences of sets.

4 System Demonstration

ISR3 was the data management system used during demonstrations of the Mobile Perception Laboratory (MPL) in the fall of 1993. These demonstrations showed the integration of low-level, reactive behaviors such as road following and obstacle avoidance with high-level perceptual capabilities such as landmark recognition, as described in [8]. ISR3 was therefore forced to support a wide variety of image processing needs.

At one extreme were groups of processes that only used ISR3 as a low-latency communications channel. For example, ALVINN [7] (a neural network road follower), its video preprocessor, and the MPL's steering/throttle controller used ISR3 in this way, with the video preprocessor passing reduced (32x32 intensity) images to ALVINN at approximately 10Hz, and ALVINN sending steering commands to the controller at the same rate. These processes did not require any complex data storage or retrieval capabilities, but the speed of the communication channel was critical, since the latency of road following is one of two factors (along with the latency of obstacle detection) that determines a vehicle's speed. (In these experiments, obstacle detection was slower than road following and limited MPL to 5 mph, so slower communication could have been tolerated.) The stereo correspondence and reflexive avoidance modules also used ISR3 as a simple communications channel, this time for passing 255×240 disparity images at about 2.5Hz.

At the other extreme were processes that relied primarily on ISR3's data storage and retrieval capabilities. The landmark recognition behavior, for example, was composed of three processes, one of which retrieved models of landmarks from the model base (or "map") based on the vehicle's estimated position, one which extracted 2D line segments from images (after using color and texture to focus attention), and one of which matched image lines to model lines, determining the vehicle's position and orientation relative to the map [3]. In this case, the speed of communication between these processes was less critical; the vehicle was stopped during landmark recognition, so delays were not a hazard. Moreover, the model matching algorithm took on the order of a minute to execute, so any communication delays were comparatively small. Rapid spatial access to the data was critical, however; the model matcher only ran as quickly as it did because it used the ISR3's spatial access routines to quickly retrieve sets of data lines near the projections of model lines. Data storage and retrieval efficiency was therefore the critical item for these processes.

5 Future Work

In general, ISR3 has been very successful when used as a data store for the structured storage and retrieval of temporary image data, such as regions of interest, points and line segments. We underestimated, however, how often it would be used as a simple communications channel and how often we would need to port it to other machines, and are making improvements in these areas, as discussed below.

5.1 Point-to-Point Communications

The version of ISR3 used on MPL in the fall of 1993 used the physically shared memory of a Silicon Graphics Iris 4D to implement interprocess communication. This had very low overhead, but meant that ISR3 could not be ported to machines (or networks of machines) without shared memory.

We are currently replacing the shared memory model with an explicit virtual point-to-point communications layer developed on top of TCX, to make ISR3 portable to machines without shared memory and to provide the simplest possible communication mechanism for processes which do not require anything more. Processes will pass data between themselves by opening a communications channel and writing to it. The channel will effectively copy data during transmission, resulting in more data copying but no synchronization overhead (note that the hardest timing constraints are on low-level processes like road following that pass small amounts of data and therefore will require very little data copying). This virtual communications layer will be designed to use the fastest means of communication available between any two processes, whether through shared memory or across a network. In all cases, however, it will provide a simple point-to-point interface to the system designer.

5.2 Long-term Data Management

Although most visual data is temporary, some types of data such as maps and object models are persistent and need to be managed. Currently, long-term and short-term data are not distinguished from each other in ISR3; long-term data is simply short-term data that is never purged, and all data is potentially shared between processes. With the change to a point-to-point system, this approach will no longer be possible. Instead, there will be a special database process that manages long-term data. Processes that need to access this data open a communications channel to the database, which acts as a server. The database will manage simultaneous updates, record the history of how maps and models are changed, and be able to recover data in the event of a systems failure. The database will also provide 3D spatial organization and access mechanisms, as opposed to the current 2D ones, since long-term data is generally 3D.

The reason we believe we can provide a central client-server database without it becoming a bottleneck (as happened with CODGER) is that 1) most data is temporary and will not go through the central database, and 2) the processes that do use permanent data are typically high-level processes such as landmark navigation that do not have the most demanding timing constraints. Also, although there is some overhead involved in opening a communication channel to the central database, no process has to do this more than once, usually during start-up.

6 Conclusion

Real-time computer vision applications require tools to support communication, short-term data storage and retrieval, and long-term database management. ISR3 is a tool that supports communication and data storage (and will soon be extended to manage persistent data) that allows computer vision algorithms developed off-line to be integrated into practical real-time applications.

References

[1] J. Aloimonos. Purposive and Qualitative Active Vision. In *IUW*, Pittsburgh, PA, pages 816–828, Sept. 1990.

[2] D. Ballard. Animate Vision. *Artificial Intelligence*, 48:57–86, 1991.

[3] R. Beveridge. *Local Search Algorithms for Geometric Object Recognition: Optimal Correspondence and Pose.* PhD thesis, Univ. of Massachusetts, 1993.

[4] J. Brolio, B. Draper, R. Beveridge, and A. Hanson. The ISR: an Intermediate Symbolic Representation for Computer Vision. *IEEE Computer*, 22(12):22–30, 1989.

[5] K. Ikeuchi and M. Hebert. Task Oriented Vision. *Conf. on Intelligent Robots and Systems*, Raleigh, NC, pages 2187–2194, July 1992.

[6] K. Kluge. Multisensor System Integration for Autonomous Navigation Tasks. To appear in *Intelligent Vehicles.*

[7] D. Pomerleau. *Neural Network Perception for Mobile Robot Guidance.* PhD thesis, Carnegie Mellon University, 1992.

[8] B. Rochwerger, C. Fennema, B. Draper, A. Hanson and E. Riseman. An Architecture for Reactive Behavior. *ICPR* (this volume).

[9] A. Stentz. The CODGER System for Mobile Robot Navigation. In *[11]*, 1990.

[10] T. Strat and G. Smith. Core Knowledge System: Storage and Retrieval of Inconsistent Information. *IUW*, pages 660-665, 1988.

[11] C. Thorpe. *Vision and Navigation: The Carnegie Mellon Navlab.* Kluwer Academic Publishers, 1990.

[12] C. Thorpe and J. Gowdy. Annotated Maps for Autonomous Land Vehicles. In *IUW*, Pittsburgh, PA, pages 765–771, Sept. 1990.

[13] T. Williams. Image Understanding Tools. In *ICPR*, Atlantic City, N.J., pages 606–610, June 1990.