Flexible Vision

Mobile Agent Approach to Distributed Vision Sensor Fusion

The use of vision as a sensor source has been gaining popularity because of the vast amount of information obtainable from visual data. Past research efforts have focused on the industrial uses of vision systems for machine vision [1] in large-scale industrial manufacturing with safety systems, product inspection, inventory control, and security monitoring. With an increased interest in ubiquitous computing and the advancement of technology, low-cost vision systems that can be geographically distributed for vision sensing are now readily available. These vision systems can be easily integrated into sensor networks or robot networks as separate mobile or stationary units or fused into the sensor or robot nodes for target tracking, environmental monitoring, or intrusion detection.

In the world of robotics, vision systems have been fruitful in mobile robot systems. The wealth of information obtained from integrating vision sensors can greatly enhance the ability of a system to interact with or model the world around it. General vision research with mobile robots focuses on machine vision in terms of navigation [2], simultaneous localization and mapping (SLAM), interaction [3], and modeling. With multirobot systems (MRSs), the vision fusion technique becomes important, as the number of agents in a system grows because of potential information overloading.

The integration of strategically placed vision sensors can dramatically reduce the uncertainty of a scene. Vision sensors can be manually placed by soldiers in a war zone or integrated into the nodes of an MRS. Distributed vision sensors are generally afflicted with the same constraints as any self-sustaining distributed sensor system, with power being the most scarce and valuable resource. Communication is far more energy expensive than computation. Therefore, continuously sending images over the network is not very efficient. Hence, the efficient use of distributed vision sensors is often one of the key aspects to a successful mission. Also, with a myriad of vision sensors distributed over a given region, the continuous streaming of visual information to a central location would congest the network and limit the productivity of the overall system. In general, visual information may not be needed at all times, and instead, the vision systems should only be accessed on demand. This is especially true in situations where the interfacing

Digital Object Identifier 10.1109/MRA.2010.937857

BY STEPHEN S. NESTINGER AND HARRY H. CHENG

1070-9932/10/$26.00©2010 IEEE

system only needs to acquire visual data from specific vision systems and fuse them into usable information. To fully utilize the vast amount of potential visual information from geographically distributed systems, a new visual fusion technique is required that provides on-demand access to the vision systems while reducing the communication overhead.

Distributed vision systems have been previously studied in different scenarios [4]. Most distributed vision systems rely on hard-coded algorithms. When dealing with multiple vision systems that may be geographically distributed, it is impractical to manually update hard-coded firmware whenever a modification to the system is required. A novel solution to distributed vision systems is required to provide the necessary flexibility and maintainability needed for unstructured environments. The use of mobile agent technology in distributed scenarios has gained popularity and can enhance the design and analysis of problem domains that are geographically distributed, exist in dynamic environments, and require subsystems to interact with each other more flexibly.

Mobile agent technology has been proven to be an efficient and effective tool for sensor networks and distributed systems [5], [6]. Mobile agents can be dynamically created during runtime and dispatched to remote systems to perform tasks with the most up-to-date code. In contrast to hard-coded systems, a mobile agent that carries the required image-processing algorithm can travel between different execution environments in a network [7]. The innate mobility property of mobile agents along with their ability to carry the required data-processing algorithms allows for distributed vision sensor fusion. Therefore, the mobility of mobile agents provides large-scale distributed applications with significant flexibility and adaptability to deal with unforeseen system perturbations.

This article presents a flexible architecture for distributed vision fusion using mobile agent-based technology. Our previous work, verified through a simple feasibility case study involving part localization in the manufacturing domain, has shown that the mobile agent technology is a salient solution for remote vision sensor fusion [8]. However, the scope of the previous work was limited. This article provides more detailed information concerning the components of the architecture and their implementation, a more extensive description of the manufacturing experiment, and a new verification experimental apparatus for planetary reconnaissance.

The Architecture for Mobile Agent-Based Distributed Vision Fusion

A geographically distributed vision fusion system may comprise different types of mechatronic devices and span multiple network domains. A general architecture for distributed vision fusion using mobile agent technology is shown in Figure 1. The architecture demonstrates the possibility of having multiple geographically distributed mechatronic systems. The main components include mobile systems, stationary systems, and knowledge bases. Specialized distributed vision systems may contain all or a subset of the discussed subsystems. The communication infrastructure may include a myriad of communication methods, and its implementation is highly application specific.

Mobile Systems

Mobile mechatronic devices encompass a vast array of devices from mobile robot platforms and unmanned aerial vehicles to portable personal devices such as cell phones and personal digital assistants. Each device provides a new perspective of the environment and can be used to further enhance the virtual perception of the system. Some of the mobile mechatronic devices may not have an attached camera system and would greatly benefit from collaborative information fusion with nearby devices containing vision systems. Each mechatronic device is assumed to utilize some type of communication device, allowing it to communicate with nearby systems or directly integrate itself into a system-wide communication network. The mechatronic devices should also contain an embedded computer system, running an operating system (OS) capable of supporting a mobile agent platform.

Stationary Vision Systems

Stationary vision systems are devices deployed by personnel, manipulator systems or mobile systems. They provide a consistent viewpoint that has been localized by some desired means, including multipoint communication signal triangulation, local global positioning system (GPS), high-altitude visual systems, inertial measurement unit (IMU), and differential odometry. As with the mobile systems, stationary vision systems are assumed to have some means of communication with other nearby systems and contain an embedded system capable of supporting a mobile agent platform.

Knowledge Base Systems

Knowledge bases may be present in one or all of the mechatronic systems or reachable by network access. The knowledge bases may provide multiple functionalities and form the means for the organization, collection, and retrieval of knowledge. The type of information and ontology used for storage is application specific. A knowledge base may be implemented with a procedural reasoning system (PRS) [9], [10] for dynamic task decomposition or may contain multiple visual-processing and manipulation algorithms fetchable by mobile agents for on-demand dynamic in situ processing of visual information. Mobile agents can migrate to a knowledge base and acquire multiple algorithms suited for the current task or operation of the requesting mechatronic device or provide new visual tasks to the mechatronic device as prescribed by the PRS. A PRS-enabled knowledge base or expert system may provide solutions to encountered problems or correlate the visual tasks of the system as a whole. New visual-processing and manipulation algorithms or visual tasks can be uploaded to the knowledge base by users.

Communication

The communication infrastructure of the system is highly application specific. It is assumed that all mechatronic systems contain some type of communication device, allowing for the sharing of mobile agents. The simplest network solution is to have a system-wide area network, allowing all managed devices to communicate with each other via the same protocol. The network may be managed by a single-server system or

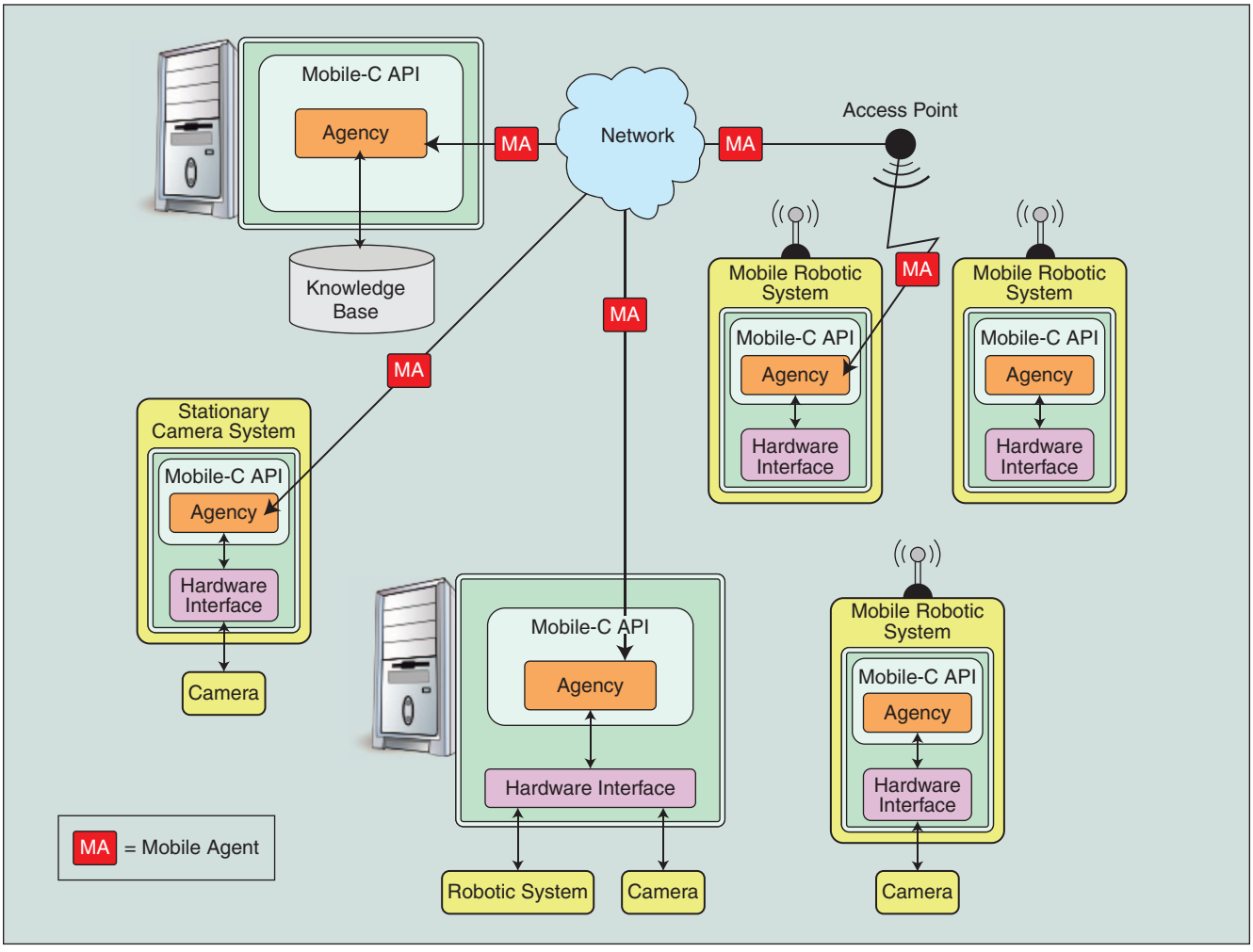

Figure 1. *The architecture for mobile agent-based distributed vision fusion.*

router, as in normal office network deployments, use an ad hoc network infrastructure, or both.

Not all devices need to be connected to the same network. Some distributed systems may detract from or be strategically deployed out of communication range of the main network. A mobile robot may be instructed to survey an area out of communication range to deploy stationary vision systems. Also, some devices may use different forms of communication methods. Stationary systems may use radio-based communication, where the mobile systems may contain both radio and standard wireless devices, thereby allowing them to interact with both the stationary systems and overall network.

Implementation

The following sections describe in detail the implementation of the architectural components, including the mobile agent systems, vision systems, and knowledge bases mentioned in the previous section.

Mobile Agent Systems

Among all of the mobile agent systems that have been developed over the past decade, JADE [11] is a notable representative with regard to active maintenance and research use. It is IEEE Foundation for Intelligent Physical Agents (FIPA) standard compliant [14]. However, developed in Java with a very large footprint, JADE has been focused toward IT and would require developing specialized modules for integrating with low-level hardware. To facilitate the implementation of the mobile agent-based distributed vision fusion architecture, a mobile agent platform geared toward interfacing with low-level mechatronic hardware is needed. Therefore, the mobile agencies discussed in the architecture are implemented using Mobile-C [12], [13]. Mobile-C is an IEEE FIPA standard compliant multiagent platform for supporting C/C++ mobile agents in networked intelligent mechatronic and embedded systems. It was originally developed as a stand-alone mobile agent system. To provide distributed applications with significant code mobility, a Mobile-C library [15] was developed, which allows Mobile-C to be embedded into applications to support C/C++ mobile agents. The Mobile-C library is an implementation of the IEEE FIPA standard-compliant mobile agent system using Ch, an embeddable C/C++ interpreter, as its agent execution engine [16], [17]. Mobile-C agents containing mobile C/C++ code are structured in XML for portability and flexibility [18]. The Mobile-C library facilitates code mobility in C/C++ programs and the development of multiagent systems that can

easily interface with a variety of hardware devices. The library is designed to allow software agents written in C/C++ to access low-level hardware as well as to provide a degree of mobility across heterogeneous computer systems. C/C++ was chosen as the library and agent language because of its wide availability, portability, and flexibility.

Mobile-C also provides an interface between agent and binary space. Using the provided application programming interface (API) functions, agent space scripts are able to access data, synchronize, and communicate with code running in binary space, and vice versa. Agents are also able to communicate and synchronize with each other to perform complex tasks that require interagent coordination. The comparative study between Mobile-C and JADE indicates that the agent migration in Mobile-C is approximately 50% faster than that in JADE [15]. The study also indicates that both Mobile-C and JADE are scalable with respect to the number of migrating agents.

Vision Systems

The vision systems discussed in the architecture comprise a camera and an embedded microcomputer system used to interface with the camera hardware and apply either image or video processing to the camera data stream. The embedded hardware runs an embedded OS with a Mobile-C agency on top of it, allowing for the migration of agents into the vision system. Therefore, systems requiring video or image access can send an agent to the vision system and acquire the desired vision data. This method provides the most flexibility, since the incoming agent can apply its own specific image- or video-processing algorithms to the data stream instead of only using what is available on the distributed vision system. Safeguards may be implemented to protect the vision system from mobile agent saturation. One method is to limit resource allocation to a certain number of agents, either using mutual exclusions or queuing the agents. Another method is to use FIPA communication agreements, where the systems must first request the use of a resource. Once permission has been granted, the requesting system can then send an agent containing the desired vision-processing algorithm to the system providing the vision resource. Mobile-C provides many methods for implementing resource restriction, including condition and mutual exclusion variables, FIPA communication, or queuing agents.

Image Processing and Manipulation

To facilitate the acquisition of video or image data, vision systems utilize OpenCV [19] to provide incoming agents with onboard sophisticated vision algorithms and ImageMagick [20] to provide simple image manipulation and processing. OpenCV contains an optimized collection of C libraries spanning a wide range of computer vision algorithms. These algorithms include motion segmentation and pose recognition [21], multiprojector display system [22], object and face recognition, and three-dimensional (3-D) reconstruction. ImageMagick is a widely used set of image manipulation utilities that provide a robust collection of tools and libraries for writing, reading, and manipulating images. Use of these visual processing and manipulation libraries by the agents are facilitated by the inclusion of the Ch OpenCV package [23] and the Ch ImageMagick package [24]. The Ch OpenCV and Ch ImageMagick packages are Ch bindings to the OpenCV and ImageMagick C libraries, respectively. With the Ch OpenCV and Ch ImageMagick packages, all C (or C++) programs using function from the OpenCV and ImageMagick C libraries can readily run in Ch interpretively without compilation.

In the world of robotics, vision systems have been fruitful in mobile robot systems.

Knowledge Base

The knowledge base is stored in a database and implemented using either open database connectivity (ODBC) or Structured Query Language (SQL). Access to the knowledge base by mobile agents is facilitated through the use of Ch ODBC [25] or Ch SQLite [26] packages, which are the Ch bindings to the ODBC and SQLite C libraries, respectively. Upon arriving at the knowledge base, mobile agents may query or modify the database through available Mobile-C services or provide their own mechanism for accessing the database.

Experimental Case Study

The following sections describe the two experimental case studies in which distributed vision systems benefit from the integration of mobile agents for vision fusion. The first example focuses on the concept of part localization in an automated manufacturing workcell while the second focuses on tier-scalable planetary reconnaissance with vision sensor fusion between multiple camera views. The complete software, including the mobile agency, Mobile-C, the mobile agents in XML, and the mobile code in C used in the following experiments, is available from the Web site for the project [13], [27].

Automated Vision System for Manufacturing

The lack of reprogrammability is a pinnacle drawback to industrial machine vision systems. System designers trying to reduce vision system complexity while under tight operating and budgetary conditions have to choose between creating a generalized vision system that requires a significant amount of reprogramming to get it to perform practical vision tasks or a turnkey vision system that provides a total solution to a single given industrial vision task [28]. Future developments are focused on maintaining automated flexible manufacturing, where flexibility is enhanced by injecting programmability directly into the automation process to ensure vision-assisted automation cost savings [29]. To further cut costs, vision control systems have recently moved to using affordable commercial off-the-shelf vision components because of their good performance-to-low cost ratio [30]. The mobile agent architecture for distributed vision fusion presented in the "The Architecture for Mobile Agent-Based Distributed Vision Fusion" section is a flexible architecture well suited to maintain manufacturing flexibility. The architecture is capable of integrating new

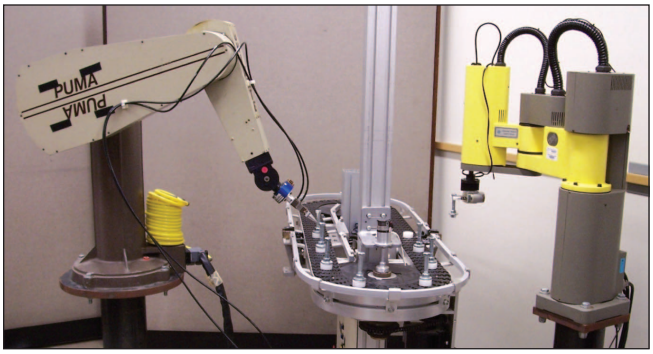

Figure 2. *An automated robotic workcell.*

off-the-shelf vision technology while providing in-process vision reprogrammability of potentially distributed vision systems. Mobile agents carrying new on-demand vision programming can be sent to any required distributed vision system for in situ image acquisition, analysis, and object recognition.

An experimental robotic workcell, shown in Figure 2, was used to verify the feasibility of the mobile agent-based vision fusion architecture. The assembly workcell was developed at the Integration Engineering Laboratory at the University of California (UC) Davis and comprises of a Puma 560 and an IBM 7575 robotic manipulator and a conveyor system [31], [32]. The robotic systems were retrofitted to comply with open-architecture requirements using two Delta Tau Data System's Turbo PMAC2 peripheral component interconnect (PCI) controllers [33]. Control of the assembly system using Ch and mobile agents has been previously verified [34]. In this experiment, the robotic manipulators' base frame would move relative to the world frame under the robot's motion, since they are not bolted to the ground. Therefore, the robots would have to be recalibrated after each run in order for the robots to securely acquire the part. The parts themselves are not always placed in the same initial configuration. To deal with uncertainties in part configuration and disturbances to the robots' reference frame, a vision sensor was integrated into the robotic workcell. The vision sensor provides the system with the ability to accurately identify and locate objects even if the objects were moved since the last calibration. Control of the robot manipulators and vision fusion process is accomplished through the use of mobile agents. The following sections describe the manufacturing experiment used to validate mobile agent-based vision fusion.

Experimental Framework

A National Television System Committee (NTSC) Watec LCL902K camera was in tegrated into the automation workcell and mounted to the IBM 7575 robot, as shown in Figure 3. A custom-machined mount was used to position the camera to have the end effector be centered in the field of view while the robot was in ready position. The camera was directly mounted to the robot as opposed to somewhere in the world frame to avoid unnecessary camera frame-to-robot frame calibration and transformation steps. The camera was mounted approximately 2 ft from the end effector because of a low pixel resolution of 640×480. The simulated part of the assembly automation workcell was manufactured to be easily recognized by the camera. The part, shown in Figure 4, was machined out of aluminum for the base and shaft and delrin for the top. This provides a good weight distribution for part stability during conveyor transport. The entire part was coated with five layers of a spray-on rubber compound to assist gripping. The rubber was removed from the top to provide a naked delrin square for visual part identification.

The hardware architecture for the case study is shown in Figure 5. The automation workcell and the vision system each have their own computer system. The computer systems of the automation workcell and the vision system run a Mobile-C agency that is ready to receive mobile agents.

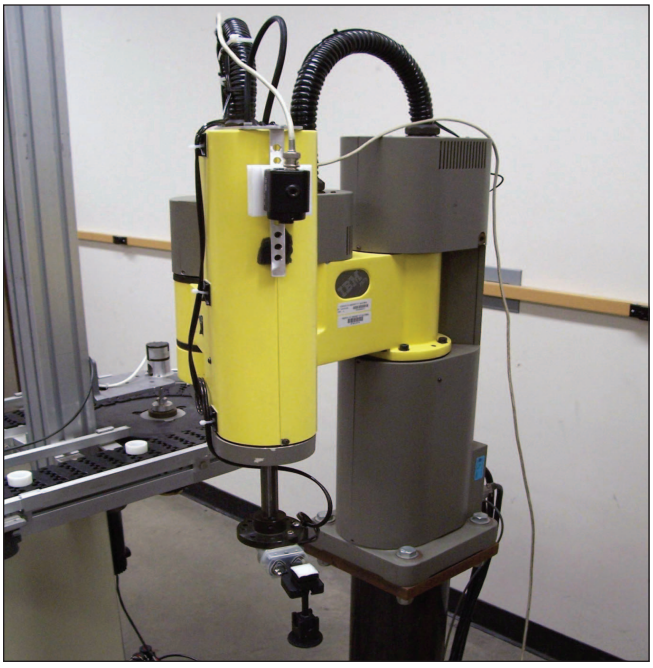

Figure 3. *The IBM 7575 with a mounted camera.*

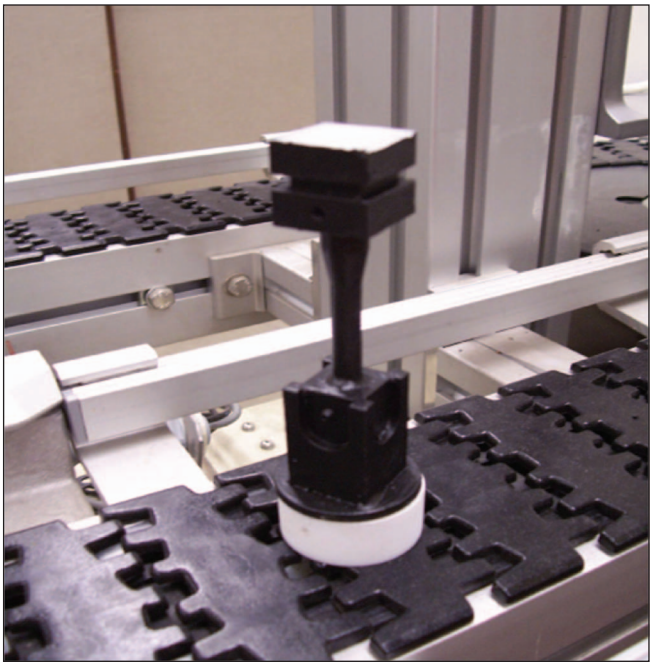

Figure 4. *A simulated assembly part situated in a conveyor carrier.*

Vision-Assisted Automation Experiment

An automation agent containing the desired control functions that make up the first automation task is sent to the automation workcell from Machine 1 to Machine 2. Once the agent has been received, it begins its execution on the automation workcell computer. The mobile agent code simulates the operations of an automation assembly. When the automation workcell is started, the two robots and the conveyor system are moved to their ready positions after initialization and calibration. Then, the IBM 7575 robot is used to place the camera over the area where the parts to be picked up are expected to be. The automation agent running on Machine 2 subsequently generates a new vision mobile agent containing the required object-recognition algorithm and sends it to the vision system on Machine 3. The vision agent accesses the camera hardware and acquires the position of the part relative to the IBM 7575 body reference frame. The vision agent then migrates back to the assembly workcell on Machine 3 and uses FIPA communication to relay part positions to the assembly agent. The IBM 7575 then picks up the first part from the acquired location and moves the camera to the desired drop-off location. Once again, a new vision agent is deployed to the vision system with a drop-off location-recognition algorithm. After the vision agent returns, the drop-off location is relayed back to the assembly agent and the part is lowered into the drop-off container on the conveyor system. Once a part has been placed, the conveyor system will rotate. As the conveyor is rotating into its preset position, the IBM 7575 will then move back to the next acquired pickup location while the Puma 560 moves to pick up a part from the conveyor. After the Puma 560 has picked up a part, it moves to a drop-off location and positions the piece. The assembly cycle simulates the assembly operations of a part. Once a part has gone though its initial stages of manufacturing, it is placed on a conveyor system and brought to either the next stage in fabrication or packaging. The motion of the robots is synchronized to ensure that a robot is fully stopped before proceeding with the next control command.

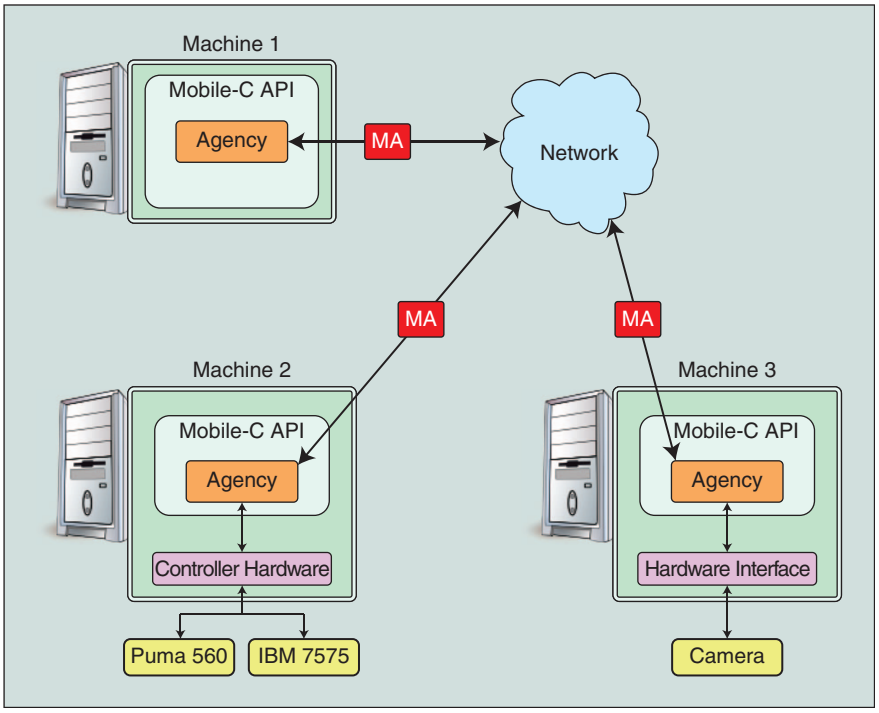

Figure 5. *The architecture for an experimental distributed vision fusion-automated manufacturing workcell.*

Object Identification

Two object-recognition algorithms were used. The first object-recognition algorithm determines the location of the part to be moved. The second object-recognition algorithm determines the position of the drop-off location on the conveyor system. Object recognition is accomplished through the use of Intel's OpenCV library via the Ch OpenCV package [24]. The Mobile-C agent code is capable of calling Intel's OpenCV API directly through the Ch OpenCV package. The result of the object-recognition algorithm is shown in Figure 6. The coordinates for the center of the part were (213, 112), as measured in pixels from the upper left corner of the original image. The agent also finds the orientation of each individual object as well as the orientation of the group (given multiple objects arranged in a linear formation).

Visual Calibration

The robot coordinate frame was aligned with the camera coordinate frame by doing a series of vision tests. The end effector holding the part was rotated at intervals of 10° within the field

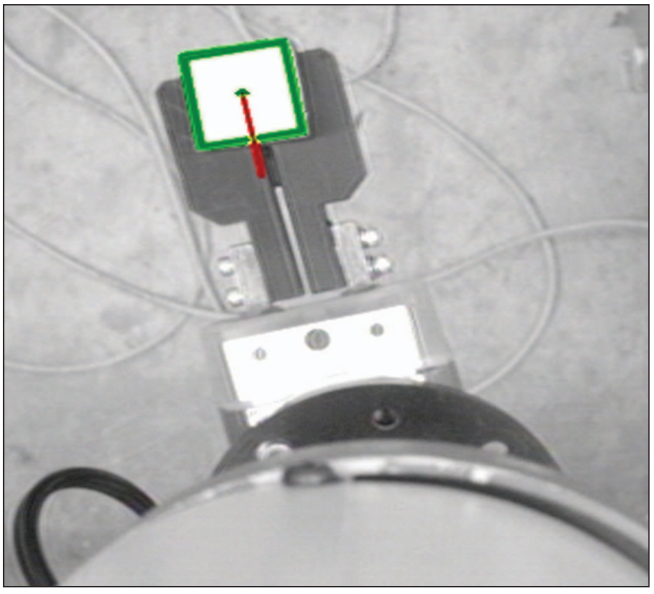

Figure 6. *The application of the object-recognition algorithm to determine the orientation and position of the part.*

C/C++ was chosen as the library and agent language because of its wide availability, portability, and flexibility.

of view of the camera in a circular motion. At each end-effector position, the object's position was recorded using the object-recognition algorithm. The absolute robot position with respect to the robot base frame and the converted vision positions are shown in Figure 7. Using a circular regression fit, an equation for each of the two circles was found from the data points. The equations were used to find a conversion factor between the two coordinate frames.

Tier-Scalable Planetary Reconnaissance

There has been a fundamental shift in remote extraterrestrial planetary reconnaissance from segregated tier reconnaissance methods to an integrated multitier and multiagent hierarchical paradigm [35]. The use of a cooperative multitier paradigm requires a flexible architecture that not only provides a mechanism for hardware access but also an agile vision fusion mechanism for vertical and horizontal integration of all vision sensor components. Multitier planetary reconnaissance systems include an orbital satellite with optical systems capable of taking large surface images, aerial unmanned vehicles capable of taking topographical maps of a desired region for exploration and robot localization, and multiple ground systems, as shown in Figure 8. The system can also include stationary vision systems for long-term visual exploration of a given area. It would be inefficient to have all of the visual systems constantly streaming visual data to all working robots, especially in a dynamic situation where a mobile robot may sporadically enter a given vision sensor's visual range. Instead of having each vision system stream visual data, a mobile agent carrying the desired visual-processing algorithm is sent to the vision system and the algorithm is carried out in situ. The agent can migrate to all nearby vision systems and then return to the sending host with the required data, which reduces the overall network congestion and decreases power consumption. A mobile robot can send out an agent to an aerial vehicle and nearby station or mobile vision systems with optimal path generation and location algorithms.

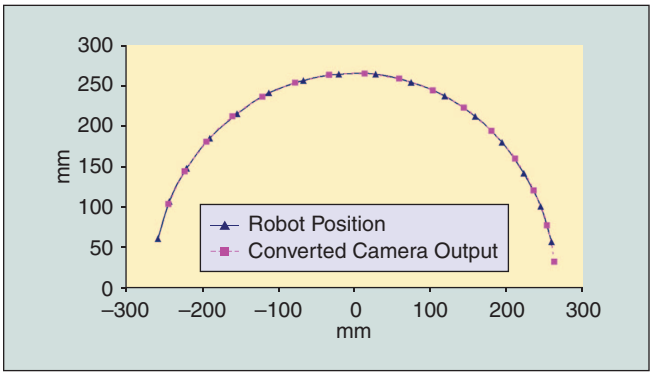

Figure 7. *The vision frame-to-robot frame calibration plots.*

Experimental Framework

A depiction of the experimental scenario for mobile agent-based vision sensor fusion in tier-scalable planetary reconnaissance is shown in Figure 8. The main experimental objective is to have a mobile robot with specialized equipment locate and take mineral samples of desirable rocks. However, the sensor information of the mobile robot is limited, and the mobile robot is incapable of locating a desirable target on its own. The mobile robot will utilize the visual system of a manipulator robot exploring the same area and the visual system of an aerial robot taking topological images to choose and localize acceptable rocks for sampling. The main purpose of this case study is not only on the actual algorithms used to implement object detection or path planning but also to show how mobile agents can be utilized to integrate information obtained from distributed vision systems. The aerial view is too coarse to detect desirable rocks but is ideal for optimal path planning. The view from the manipulator robot, on the other hand, provides enough resolution to deduce which rocks would make superb targets for sampling but is too low to the ground for optimal path planning. Ideal rock targets for sampling can be detected using specific sensors that can detect certain radioactive isotopes, optimum rock geometry, or remotely chosen by human operators.

The laboratory setup to simulate the environment is shown in Figure 9. The visual systems, mobile robots, and field objects are highlighted. A K-Team Khepera III robot is used as a mobile robot, a Puma 560 with an attached Logitech QuickCam Pro 9000 simulates the mobile manipulator, and a Watec LCL-902K B/W camera mounted above with a wide field-of-view

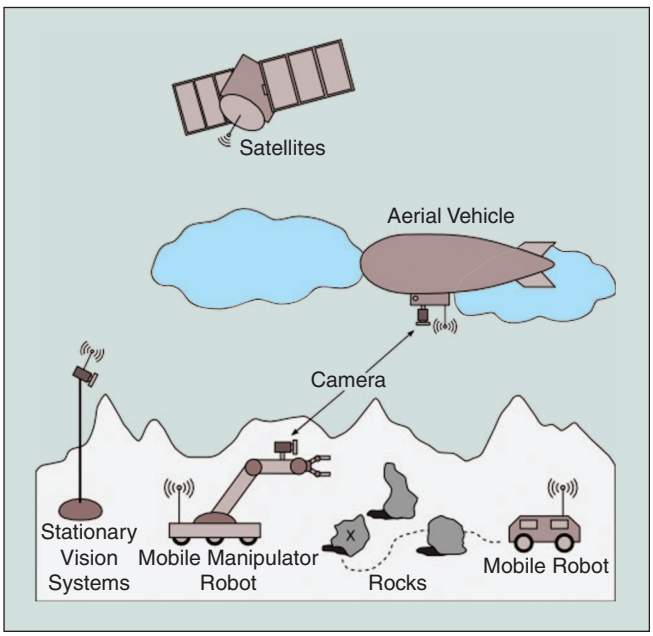

Figure 8. *The distributed network of mobile and stationary intelligent robotic agents exploring a planetary environment.*

lens is used to simulate the visual system of an aerial robot. Small paper drinking cups are used as field objects, with one cup colored red to signify the target.

Field Object-Detection Algorithm

The detection of the field objects is highly dependent on the environmental conditions and the geometrical configuration of the objects that need to be detected. In general, some aspects of the objects are used to distinguish them from the background along with specifying certain assumptions of the scene. The field objects consist of only the cups. Figure 10(a) and (b) shows the aerial and manipulator views of the field objects, respectively. The field object-detection algorithm looks for the bright contrasting regions, differentiating the tops of the cups with the rest of the environment. The algorithm runs a contour-locating algorithm to determine the locations relating to the top of the cups. The rectangles in Figure 10(a) and (b) show the bounded region of the located contours using the field object-detection algorithm, with a dot representing the center of the bounding rectangles.

Target-Detection Algorithm

Target detection can only be accomplished using the visual system of the manipulator, because it is the only available color camera, and the aerial view is assumed to be too high to distinguish desirable target characteristics. The target-detection algo rithm determines the desirable target by taking into account that the target is colored red and subtracting individual color channels, as shown in Figure 11. The found target is shown in Figure 12.

Multiview Target-Fusion Algorithm

To deduce which field object in both the aerial and manipulator views correlate with the target, a target-correlation algorithm is used to determine which field object was the target in the manipulator view, and a fusion algorithm is used to correlate the target manipulator view field object with the field objects of the aerial view. This required the assumption of no possible object occlusion.

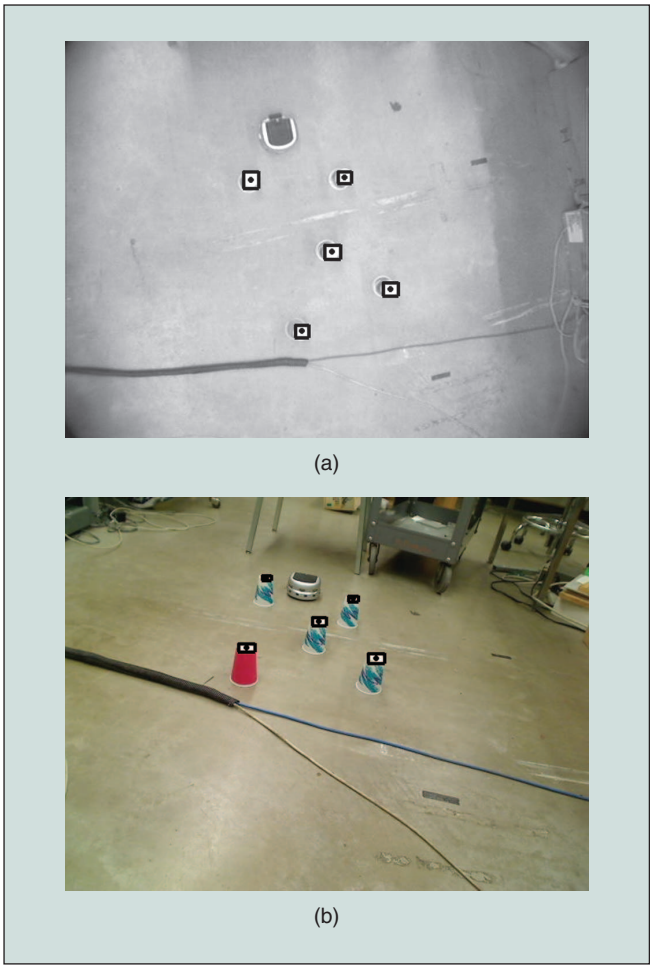

Figure 10. *Detected field objects are shown encompassed in a black rectangular box, with a black dot depicting the center of the rectangular region. (a) An aerial view and (b) mobile manipulator view.*

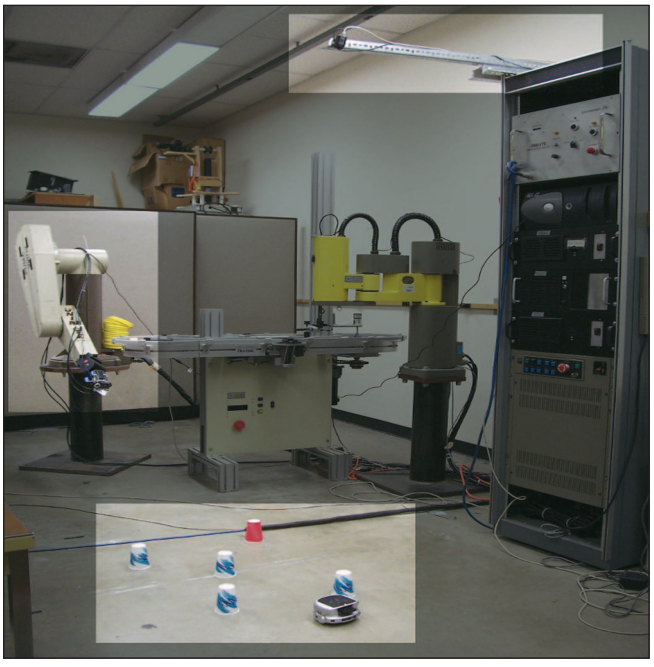

Figure 9. *An experimental setup.*

Figure 11. *The summation of the two subtractions.*

Mobile-C is an IEEE FIPA standard compliant multiagent platform for supporting C/C++ mobile agents in networked intelligent mechatronic and embedded systems.

The target-correlation algorithm utilizes the rectangular regions found from the field object and target-detection algorithm. It is assumed that the center of a target object's rectangular region will lie within the left and right extents of the correlating field object's rectangular region and that the target object's rectangular region center will be below that of the correlating field object's center. The algorithm goes through all of the found targets and compares each one with all of the found field objects. The algorithm first determines whether the target object's rectangular region center lies within the left and right extents of the field object's rectangular center and whether or not the target object is visually below the field object. The final test determines which field object is vertically closest to the target object by calculating their vertical distance. The field object with the smallest vertical distance is set as the target. The index of the field object is stored in an array for later use.

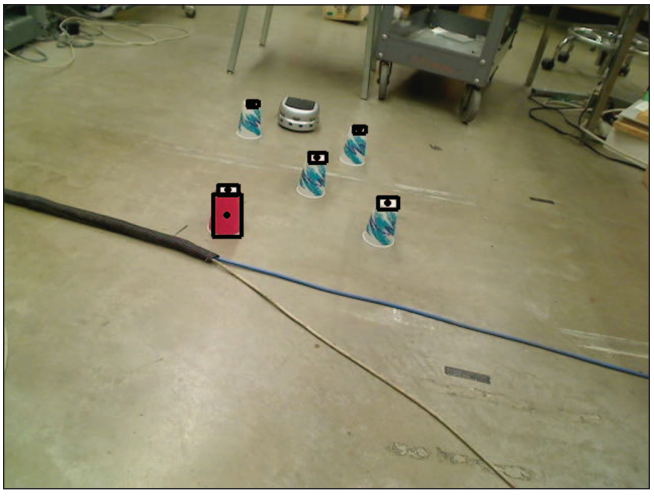

Figure 12. *Finding the target along with detected field objects.*

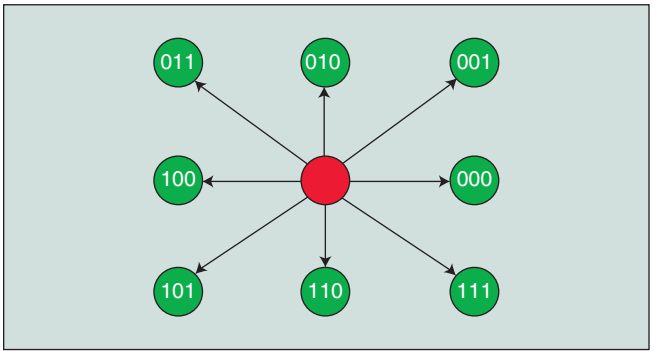

Figure 13. *Direction encoding, as used in [38].*

Hence, a fusion algorithm is used to correlate the manipulator target field object with the field objects of the aerial view. Fusion of the separate views is rather complex because of the skewed nature of the manipulator view. There are multiple methods that can be implemented to handle the skewed data. One method is to locate the robot in both manipulator and aerial views and use it as a pivot point while matching the object positions. The second method requires a localization mechanism, either through the use of dead reckoning, communication channel triangulation, or from a planetary or localized GPS system. Given the orientation, the manipulator view can be rotated to correspond with the view of the aerial robot.

Once the images have been properly aligned, a vertical scan is used to first sort the found field objects from the manipulator view using their respective center pixel points. A second sort is then applied using the horizontal position of their respective center pixel points. The same technique is applied to the aerial view based on the orientation of the manipulator robot. Once completed, the sorted aerial field object array should coincide with the manipulator field object array. It is then a simple task to correlate the targeted object from the manipulator view with the aerial view.

Path Planning Using the Genetic Algorithm

Although any path-planning algorithm can be used, a genetic implementation was chosen because of its innate nature of finding multiple solutions. The genetic algorithm was implemented using the Genetic Algorithm Utility Library (GAUL), a C-based, open-source programming library designed to assist in the development of the code that requires evolutionary algorithms [36]. The Ch GAUL package [37] provides Mobile-C agents the ability to utilize all of the functionality provided in the GAUL C library.

The genetic algorithm follows the same implementation as was done in [38]. The aerial image was subdivided into zones of 10×10 pixels and stored internally as a two-dimensional array. If a field object encompassed part of a zone, that zone was deemed blocked and noted as a one in the two-dimensional array. Field-object locations were found via the field object-detection algorithm discussed in the previous section, which provided the center location and bounding rectangular region of each object. The radius of the robot is added to the field-object bounding rectangular region, which reduces the complexity of the path-planning problem by allowing a singularity approximation of the mobile robot.

Each chromosome contains $2(n + m) = 2(64 + 48) = 224$ genes, as suggested in [38], given that the 640×480 pixelized image was broken into 64×48 zones. The genes contain 3 bits and define the direction the mobile robot should move from its current position. The direction encoding scheme is shown in Figure 13.

The genetic algorithm was run using a population size of 2,000 with 100 evolutionary cycles. A multipoint mutation and allele crossover mixing were used with a crossover rate of 0.9, mutation rate of 0.02, and migration rate of 0. Population entity selection for migration and mutation were accomplished using the selected one best of two approaches. Crossover entity

selection was accomplished using the two best of two approaches. The genetic algorithm path-planning output is shown in Figure 14.

Experiment

The experiment initiates with the Khepera III mobile robot approaching the designated area of rock sampling. The Khepera III mobile robot is running a Mobile-C agency and is under the control of a mobile agent, with the objectives of searching for and sampling desirable rock specimens. The mobile agent migrates itself to the mobile manipulator robot, where it utilizes the available Ch OpenCV commands and runs through the field and target object-detection algorithms. From there, it migrates to the overhead aerial robot with the field and object data obtained from the mobile manipulator robot, runs the field object-detection algorithm on the aerial view, and fuses the data from both views to locate the target. The mobile agent then populates a binary map of the area, where the objects are defined as being one and the other areas as being zero. Afterward, the mobile agent runs the genetic path-planning algorithm using the Ch GAUL package, producing waypoints for the Khepera III mobile robot. Finally, the mobile agent migrates back to the Khepera III mobile robot with the waypoints for the path.

The determination of a satisfactory path was dependent on the gene seed and the population size used. Small population sizes were incapable of producing a path ending at the desired target, which is mainly due to the method of seeding the population. Generally, each chromosome of the population is first seeded with a randomized gene selection. However, some of the populated chromosomes did not provide an appropriate path or a useful population entity. A better approach would be to still utilize random gene selection but only to populate the world with chromosomes that provide a viable path.

Discussion

The following sections discuss the differences between mobile agent and monolithic approaches to distributed vision sensor fusion, scalability of the described system, and challenges and future concerns of mobile agent approaches.

Comparison Study with Monolithic Approaches

Many vision-based sensor systems are implemented using a monolithic client–server approach that generally operates over standard communication infrastructures. Visual data are either provided on demand or continuously streamed to the clients where the image processing and analysis computation are performed. As such, these systems demand a high bandwidth communication infrastructure, especially with high-fidelity images. When dealing with mobile and remote systems, it is impractical to assume that all clients contain the necessary resources and required supplemental libraries to process large volumes of vision data. Since vision systems are capable of capturing high-quality images, they typically also contain the necessary resources to process the images as well. Therefore, it would be beneficial to have the vision system perform the required image processing and manipulation and send the results back to the client. Algorithm reconfigurability can be accomplished using monolithic techniques by preinstalling all necessary image-processing and manipulation algorithms onto the vision sensor and having the client select the desired algorithm. However, it is highly impractical to assume that all the possible scenarios can be accounted for and preinstalled on a vision system. Mobile agents provide an innate capability to handle algorithm variability, because they carry the necessary algorithms with them as they selectively traverse over multiple hosts.

Mobile agent technology has been proven to be an efficient and effective tool for sensor networks and distributed systems.

In our first experimental example, continuously sending the visual data to all robots would congest the network, leaving no room for necessary real-time interrobot communication that would eventually lead to robot collision. Mobile agents are used to perform in situ image processing and manipulation at the vision source and return back to the client with the processed data. The algorithm provided in the mobile agent code is also modified based on the current state of the robotic system, demonstrating the adaptability of the mobile agent approach to system variability. System adaptability may also be implemented in the monolithic systems by using the semidynamic approach of scripting the sharing of binary files over the network. However, this process is rather cumbersome, especially when dealing with heterogeneous systems with different compilation and linking needs. Also, it has been reported that mobile agents provide a faster adaptability response time in comparison with semidynamic monolithic implementations [5], [39].

In our second experimental example, the mobile robots operate on embedded systems under resource constraints and relatively slow processing capabilities. Mobile systems are generally constantly moving relative to vision sensors and experience more communication noise than static systems. Having the vision data to be continuously streamed to the mobile robot may have adverse affects on the robot's ability to safely navigate complex terrain and reduce image-processing functionality because of sporadic loss of video information, jeopardizing the mission. Also, the higher tier-level systems are

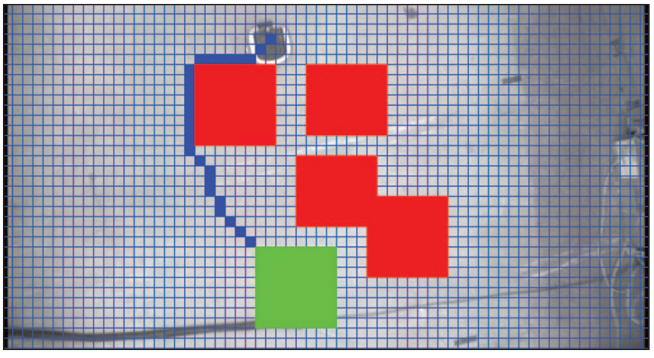

Figure 14. *Finding a path using the genetic algorithm.*

With the Ch OpenCV and Ch ImageMagick packages, all C (or C++) programs using function from the OpenCV and ImageMagick C libraries readily can run in Ch interpretively without compilation.

more computationally rich and provide a better environment for executing image-processing and path-planning algorithms. Mobile agents provide an invariant execution of code over disparate hosts and can independently handle necessary sensor fusion requirements by migrating with previously processed sensor data. Mobile agents can also independently migrate to more resource-rich systems for faster execution to improve overall system response and efficiency. Therefore, the use of mobile agents is highly advantageous in applications dealing with dynamic, geographically unstructured environments.

Scalability

The scalability of a mobile agent-based system is dependent on the application and is due to resource limitations in the system. In our experimental examples, scalability issues may arise when too many mobile agents are sent to a remote sensor system with limited resources. One method of handling mobile agent overloading is to limit the number of mobile agents allowed to access a given resource on the desired remote system. Another method is to restrict the number of the mobile agent migration messages from external sources or cap the number actively running mobile agents on the system. All of the discussed methods for handling scalability concerns can be easily implemented within the mobile agency by using standard resource access restriction methods such as semaphores or mutexes or by simply using a counter. The comparative study between Mobile-C and JADE indicates their ability to handle more than 100 and potentially more mobile agents on a given system [15].

Selecting an Appropriate Mobile Agent System

One major concern when either enhancing the current implementation or designing a new vision system with mobile agents is the selection of an appropriate mobile agent system. Naturally, applications developed in C/C++ can be easily integrated with a C/C++-based mobile agent system like Mobile-C. Using the same language for application and mobile agent code greatly enhances the interoperability of functions and variables in both binary and agent spaces. Mobile agent code can seamlessly call functions and access variables in binary space and vice versa. Applications developed in Java might use a Java-based mobile agent system such as JADE. However, designers should also consider the implementation language of the low-level device drivers for the desired vision sensors that are typically written in C.

Challenges and Future Concerns

Some challenges and future concerns relating to mobile agent-based distributed vision sensor fusion are listed as follows:

- design optimized adaptation methods for collaborative vision processing, especially in dynamic network topologies due to node mobility
- study the effects of the mobile agent approach on real-time deterministic vision sensor fusion applications
- fault-tolerance mechanisms should be considered in case a mobile agent is lost in transit or a vision system goes down.

Conclusions

This article presented a mobile agent-based distributed vision fusion architecture that provides a flexible vision fusion solution to increase power efficiency by reducing excessive communication and enhance sensor fusion capabilities with migratory in situ on-demand algorithms for vision data processing and analysis. The IEEE FIPA standard-compliant mobile agent system, Mobile-C, implemented as a C library, is used as the foundation for the mobile agent-based distributed vision fusion architecture. Mobile agents dynamically migrate from one sensor node to another to fully combine all necessary sensor data in a desired manner specific to the system requesting the data. Dispatching mobile agents to target vision systems on the network is done on demand, reducing network congestion and the required communication bandwidth. The use of mobile agents in a distributed vision system allows for the encapsulation of specific fusion techniques. The differences between monolithic and mobile agent-based approaches along with future considerations were discussed. The validity of the architecture was proven through two separate case studies. The first case study involves the localization of a part in a real experimental setup with a retrofitted robotic workcell composed of a Puma 560, IBM 7575, conveyor system, and vision system. The second case study vertically and horizontally integrates multiple systems as a tier-scalable planetary reconnaissance experimental system involving two vision systems: a Puma 560 manipulator and a K-Team Khepera III mobile robot. All source code including Mobile-C, the mobile agents, and the mobile agent code presented in the article are available at the project Web site [27].

Acknowledgments

This project was partially funded by UC Davis and by a fellowship from the Sandia National Laboratories. The authors would like to acknowledge Alex Rumer and David Ko for their contributions to the project.

Keywords

Mobile agent, distributed vision, sensor fusion.

References

[1] P. C. West, "Machine vision in practice," *IEEE Trans. Ind. Applicat.*, vol. IA-19, no. 5, pp. 794–801, 1983.

[2] D. Murray and C. Jennings, "Stereo vision based mapping and navigation for mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, NM, Apr. 1998, pp. 1694–1699.

[3] C.-H. Kuo, C.-M. Yang, and F.-C. Yang, "Development of intelligent vision fusion based autonomous soccer robot," in *Proc. 2005 IEEE Int. Conf. Mechatronics*, Taipei, Taiwan, July 10–12, 2005, pp. 124–129.

[4] D. R. Karuppiah, Z. Zhu, P. Shenoy, and E. M. Riseman, "A fault-tolerant distributed vision system architecture for object tracking in a smart room," *Lect. Notes Comput. Sci.*, vol. 2095, pp. 201–219, 2001.
[5] A. Boulis, "Programming sensor networks with mobile agents," in *Proc. 6th Int. Conf. Mobile Data Management: MDM '05*. New York, NY: ACM, 2005, pp. 252–256.
[6] D. Georgoulas and K. Blow, "Intelligent mobile agent middleware for wireless sensor networks: A real time application case study," in *Proc. Advanced Int. Conf. Telecommunications*, 2008, pp. 95–100.
[7] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding code mobility," *IEEE Trans. Software Eng.*, vol. 24, no. 5, pp. 342–361, 1998.
[8] S. S. Nestinger, D. Ko, A. Rumer, and H. H. Cheng, "Mobile agent-based remote vision sensor fusion," in *Proc. 2008 IEEE/ASME Int. Conf. Mechatronic and Embedded Systems and Applications*, Beijing, China, Oct. 12–15, 2008, pp. 482–487.
[9] F. F. Ingrand, M. P. Georgeff, and A. S. Rao, "An architecture for real-time reasoning and system control," *IEEE Expert*, vol. 7, no. 6, pp. 34–44, Dec. 1992.
[10] F. F. Ingrand, R. Chatila, R. Alami, and F. Robert, "PRS: A high level supervision and control language for autonomous mobile robots," in *Proc. 1996 IEEE Int. Conf. Robotics and Automation*, Apr.1996, vol. 1, pp. 43–49.
[11] JADE—Java Agent DEvelopment Framework [Online]. Available: http://jade.tilab.com/
[12] B. Chen, H. H. Cheng, and J. Palen, "Mobile-C: A mobile agent platform for mobile C/C++ agents," *Software Pract. Exp.*, vol. 36, no. 15, pp. 1711–1733, Dec. 2006.
[13] Mobile-C: A multi-agent platform for mobile C/C++ code [Online]. Available: http://www.mobilec.org
[14] FIPA: The Foundation for Intelligent Physical Agents [Online]. Available: http://www.fipa.org/
[15] Y.-C. Chou, D. Ko, and H. H. Cheng, "An embeddable mobile agent platform supporting runtime code mobility, interaction and coordination of mobile agents and host systems," *Inform. Softw. Technol.*, vol. 52, no. 2, pp. 185–196, 2010.
[16] H. H. Cheng, *C for Engineers and Scientists: An Interpretive Approach*. New York: McGraw-Hill, 2009.
[17] Embedded Ch, SoftIntegration, Inc. [Online]. Available: http://www.softintegration.com/products/sdk/embedded ch/
[18] B. Chen, D. Linz, and H. H. Cheng, "XML-based agent communication, migration and computation in mobile agent systems," *J. Syst. Softw.*, vol. 81, no. 8, pp. 1364–1376, 2008.
[19] Open source computer vision library [Online]. Available: http://opencvlibrary.sourceforge.net/
[20] ImageMagick [Online]. Available: http://www.imagemagick.org
[21] G. R. Bradski and J. Davis, "Motion segmentation and pose recognition with motion history gradients," in *Proc. IEEE Workshop Applications of Computer Vision*, 2000, p. 238.
[22] R. Yang, D. Gotz, J. Hensley, H. Towles, and M. Brown, "Pixelflex: A reconfigurable multi-projector display system," in *Proc. 2001 IEEE Conf. Visualization*, San Diego, CA, Oct. 21–26, 2001, pp. 68–75.
[23] Q. Yu, H. H. Cheng, W. W. Cheng, and X. Zhou, "Ch OpenCV for interactive open architecture computer vision," *Adv. Eng. Softw.*, vol. 35, no. 7–8, pp. 527–536, 2004.
[24] S. S. Nestinger and H. H. Cheng, "Interactive image processing and manipulation," in *Proc. 2007 ASME/IEEE Int. Conf. Mechatronic and Embedded Systems and Applications*, Las Vegas, NV, Sept. 4–7, 2007, pp. 245–254.
[25] Ch ODBC [Online]. Available: http://www.softintegration.com/products/toolkit/odbc/
[26] Ch SQLite [Online]. Available: http://chsqlite.sourceforge.net
[27] Mobile agent-based vision sensor fusion [Online]. Available: http://www.mobilec.org/apps/vision/
[28] P. F. Whelan, "System engineering issues in industrial inspection," in *Proc. IEE Colloquium Industrial Inspection* (Digest No: 1997/041), Feb. 1997, pp. 1/1–1/6.
[29] H. Golnabi and A. Asadpour, "Design and application of industrial machine vision systems," *Robot. Computer-Integrated Manufact.*, vol. 23, pp. 630–637, 2007.
[30] S. H. Kim and B. K. Kim, "Analysis on time-delay of commercial off-the-shelf vision system considering motion-blur," in *Proc. 2001 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Maui, HI, Oct. 29–Nov. 3, 2001, pp. 2080–2085.
[31] X. Hu, J. Pannu, and H. H. Cheng, "Retrofitting an automatic manufacturing workcell for study of open architecture object-oriented integration of mechatronic systems," *Chin. J. Mech. Eng.*, vol. 15, no. 2, pp. 149–152, 2002.
[32] J. Pannu, X. D. Hu, and H. H. Cheng, "Retrofitting of industrial manipulators for study of open architecture integration of mechatronic systems," in *Proc. ASME 18th Computers in Engineering Conf.*, Atlanta, GA, 1998, Paper DETC98/CIE-5533.
[33] D. T. D. System, *PMAC User's Manual and Software Reference*, Firmware version 1.13, Dec. 1992.
[34] S. S. Nestinger, B. Chen, and H. H. Cheng, "A mobile agent-based framework for flexible automation systems," *IEEE/ASME Trans. Mechatron.*, vol. PP, no. 99, pp. 1–10, 2010.
[35] W. Fink, J. M. Dohm, M. A. Tarbell, T. M. Hare, and V. R. Baker, "Next-generation robotic planetary reconnaissance missions: A paradigm shift," *Planet. Space Sci.*, vol. 53, no. 14–15, pp. 1419–1426, Dec. 2005.
[36] The genetic algorithm utility library [Online]. Available: http://gaul.sourceforge.net
[37] Ch GAUL [Online]. Available: http://iel.ucdavis.edu/projects/chgaul
[38] J. Tu and S. Yang, "Genetic algorithm based path planning for a mobile robot," in *Proc. IEEE Int. Conf. Robotics and Automation, ICRA '03*, vol. 1, pp. 1221–1226, Sept. 2003.
[39] L. Cragg and H. Hu, "Mobile agent approach to networked robots," *Int. J. Adv. Manufact. Technol.*, vol. 30, pp. 979–987, 2006.

Stephen S. Nestinger received his M.S. and Ph.D. degrees in mechanical and aeronautical engineering from UC Davis, in 2005 and 2009, respectively. He is an assistant professor in the Mechanical Engineering Department at Worcester Polytechnic Institute (WPI). He is also the director of the Robotics, Automation, and Mechatronics Laboratory at WPI. He is a Member of the IEEE. His research interests include real-time systems, software and systems integration, multirobot systems, collaborative cooperation, mobile agent systems, intelligent embedded and mechatronic systems, and distributed, evolutionary, and adaptive control systems.

Harry H. Cheng received his M.S. degree in mathematics and Ph.D. degree in mechanical engineering from the University of Illinois at Chicago in 1986 and 1989, respectively. He worked as a senior engineer on robotic automation systems in the Research and Development Division at UPS from 1989 to 1992. He is a professor at the Department of Mechanical and Aerospace Engineering and Graduate Group in Computer Science and the director of the Integration Engineering Laboratory at UC Davis. He is the founder of SoftIntegration, Inc. He has published more than 160 papers in refereed journals and conference proceedings, is the author of a book, and holds one U.S. patent. He is a Senior Member of the IEEE and a fellow of American Society of Mechanical Engineers (ASME). He served as the conference chair and program chair of the IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications. His research focused on computer-aided engineering, mobile agent-based computing, intelligent mechatronic and embedded systems, robotics, and innovative teaching.