An Infrastructure-Free Automated Guided Vehicle Based on Computer Vision

An Effort to Make an Industrial Robot Vehicle that Can Operate without Supporting Infrastructure

Automated guided vehicles (AGVs) have been operating effectively in factories for decades. These vehicles have successfully used strategies of deliberately structuring the environment and adapting the process to the automation. The potential of computer vision technology to increase the intelligence and adaptability of AGVs is largely unexploited in contemporary commercially available vehicles.

We developed an infrastructure-free AGV that uses four distinct vision systems. Three of them exploit naturally occurring visual cues instead of relying on infrastructure. When coupled with a highly capable trajectory generation algorithm, the system produces four visual servo controllers that guide the vehicle continuously in several contexts. These contexts range from gross motion in the facility to precision operations for lifting and mating parts racks and removing them from semi-trailers. To our knowledge, this is the first instance of an AGV that has operated successfully in a relevant environment for an extended period of time without relying on any infrastructure.

Overview

The market for AGVs is the oldest established market for mobile robots. This market is probably valued at over US$900 million today [4], [17]. The largest consumer of AGVs is the automotive industry although many other industries, including warehousing and distribution, paper, printing, textiles, and steel, also use these vehicles. Even large 65 ton material handling vehicles in the shipyards of Rotterdam and Brisbane [2] have been successfully automated.

BY ALONZO KELLY, BRYAN NAGY, DAVID STAGER, AND RANJITH UNNIKRISHNAN

Digital Object Identifier 10.1109/M-RA.2007.901317

1070-9932/07/$25.00©2007 IEEE

While there are many specialized forms of AGVs, three main types of vehicles are in use today: tugs or tractors pull several passive loads placed on wheeled platforms behind them. Unit loads carry a single load placed on a platform on the vehicle. Forked AGVs carry a single load but also pick it up and place it using fork implements.

In part, the historical success of these vehicles has been based on a strategy of exploiting the valid assumptions of structured indoor environments. Such assumptions include mostly flat floors and the assumed availability of infrastructure that is provided to support vehicle guidance. System design elements include reducing speeds to very safe levels, centralizing movement authority, and confining the vehicles to dedicated pathways, known as guidepaths, which are kept clear of obstacles to the highest degree possible.

Of course, such risk reduction comes at the cost of limitations in performance and adaptability. Contemporary AGVs rely heavily on specially installed infrastructure to determine their position in the facility. Such infrastructure is costly to install and modify.

Motivation

AGV guidance systems have been evolving for about 50 years [1]. Three guidance technologies have been dominant over this time. Wire guidance uses wires embedded in the floor that are sensed inductively in order to determine vehicle lateral position with respect to the wire. This is an earlier technology that is not used much today. Inertial guidance uses gyroscopes and wheel odometry (measurements of distance traveled). These are used to implement very accurate dead reckoning. Magnets are placed in the floor at regular intervals to be used to reset the inevitable drift of the dead reckoning system. This technology is available on the market today. Laser guidance uses a spinning laser emitter- receiver that is mounted on the vehicle. It senses the bearings to retro reflective landmarks placed carefully in the facility and then it triangulates an accurate solution. This technology is also available on the market today.

It has long been a goal of the AGV industry to reduce dependence on guidance infrastructure—the wires, magnets, and reflectors mentioned above. The need to preserve visibility of infrastructure limits the capacity of vehicles to deviate from pathways that were specified when the system was installed. Wire-guided vehicles must stay very close to the wire, laser-guided vehicles must avoid interrupting their line of sight to specially mounted retro reflectors, and inertially guided vehicles must drive over floor magnets at regular intervals.

Systems that are able to deviate significantly from their guidepaths are known as free-ranging. When vehicles are not free-ranging, a single failed vehicle can temporarily block a main thoroughfare and shut down all automated traffic.

Infrastructure dependence often prevents AGVs from operating in environments where infrastructure is difficult to employ. For example, weather conditions make outdoor environments more challenging, although radar guidance has been used successfully outdoors.

For applications that involve operations in semi-trailers, it is normally not feasible to place adequate infrastructure in the trailer. The trailers are not usually owned by the facility and they are not dedicated to any particular shipping route or customer.

A second limitation of contemporary AGVs is that they are essentially blind. With some exceptions, contemporary systems rely on precise positioning of loads because the systems cannot determine if the loads are imprecisely positioned. These vehicles may not be able to interface with loads placed by human-driven vehicles because humans do not usually position loads with the required precision.

Problem Addressed

This article summarizes the results of a five-year program that attempted to remove the above limitations by applying one key technology: computer vision. The longer-term goal is to automate all operations moving material from trailer to production line and back in automotive stamping and assembly plants. These kinds of operations include picking up and setting down loads at a number of sites in the facility. These sites include semi-trailers, tug AGVs that cooperate with the forked AGVs, automated storage and retrieval systems, and staging areas near the production line.

Several operating scenarios are typical. In the first, a forked AGV moves a filled parts rack (containing auto parts) from a trailer to one of the sites mentioned above. In the second, a forked AGV moves empty parts racks back to a trailer for removal from the facility. In a third, a tug AGV moves several full loads from storage to the production line, waits for unload, and then returns for another load.

Secondary goals of the program include investigating how costs can be reduced by retrofitting industrial trucks and by using cameras instead of laser detection and ranging (LADAR). Another secondary goal is determining the degree to which AGVs can coexist and interact with human-driven material handling vehicles.

Approach

The results of our efforts can be described in terms of four visual servo controllers, each with a specially designed vision system. One of these servos is always active. The servos coexist with conventional lower level control algorithms and higher level planning algorithms, all of which contribute to a complete solution.

To our knowledge, we have demonstrated the first reliable, infrastructure-free guidance of an AGV in a realistic setting for an extended period of time. We have also demonstrated automated stacking of parts racks based on fiducials that could be removed with additional development work. Our demonstrations of vision-based automated load acquisition and automated unloading of trailers were also infrastructure-free.

This article summarizes the overall effort to develop computer vision-based solutions to our vehicle automation problem. Technical details can be found in the many referenced technical articles [8]–[12] associated with the program.

Figure 1. *Vehicle retrofits. The tug AGV (bottom left) has a single downward-looking camera mounted to the chassis in the center of the vehicle. The fork truck (right) has two stacking cameras (cameras used for automated stacking) mounted to the roll bars and a forward-looking camera that moves with the forks. The lift and side-shift degrees of freedom of the mast are shown in the top left. A wheel encoder is attached to a small wheel that rides on top of the main drive wheel as shown. String encoders measure motions of the mast. LADARs that are normally mounted on the rear of the fork truck and the front of the tugger are not shown.*

Vehicle Retrofits

Two common material handling vehicles were initially retrofitted for autonomy. The retrofitted vehicles are shown in Figure 1. In the bottom left is a Hyster model R30FT tugger vehicle, also known as a tractor, capable of pulling several carts carrying a total weight of up to 4,500 kg. On the right is a 5,000-kg Hyster model 50 counterweight fork lift, capable of carrying loads up to 2,300 kg.

The sales volumes of such vehicles exceed those of AGVs by at least two orders of magnitude, so our sponsor felt there was long-term potential to reduce costs by exploiting a less expensive base vehicle. In the later stages of the program, we also tested our guidance system on commercial AGVs manufactured by FMC Corporation.

Although LADAR systems were used, the number of such devices was minimized. Again, we hoped that eventual volume manufacture would lower costs as volume markets drove down the price of cameras.

Both vehicles shared common computer hardware and software architectures. A central PC running the Windows NT operating system hosted a multithreaded application that controlled vehicle motion and ran vision, planning, and navigation software. The vehicles had primary power batteries at 48 V/36 V. Vicor dc-to-dc converters supplied computing, hydraulic, and control logic power at lower voltages.

Each vehicle was controlled through a custom relay board that actuated the manual controls for the vehicle's standard hydraulics and electrics. Under computer control, the relay board switched the throttle and hydraulic controls to be actuated by an I/O board that was controlled from within the Windows NT program.

The throttle on both vehicles was a 0–5-V analog signal normally controlled by a potentiometer on the accelerator. Braking was accomplished primarily through regeneration built into the vehicles. A parking brake was used for stopping. Steering on the fork lift was controlled using a second hydraulic steering box that was driven by an electric motor. Steering on the tugger was controlled by using the built-in analog steering signals for the manual control joystick.

Fork actuation was controlled through multispeed hydraulic valves. The fork lift was able to tilt the mast and lift and side-shift the forks. Initially, the width between the forks could only be adjusted manually. The original forks were later replaced by forks whose width could be automatically controlled, as described later in this article.

Encoders measured wheel rotation to provide feedback for odometry. String potentiometers measured fork positioning. Later iterations of the design used fork-mounted limit switches to detect proper load engagement. Cameras were installed on the fork lift to detect fork holes. These are the square holes in parts racks into which the forks are inserted for lifting. A NTSC digitizer captured images for the software system.

To provide localization, a downward-looking vision camera was mounted with lights beneath the vehicles. The downward vision system was originally integrated into the central vehicle control computer but evolved later into a stand-alone device with a separate CPU running Linux. The vehicle computers were networked using Ethernet.

The rest of the article focuses on the fork truck because every system present on the fork truck was also present on the tugger. The computer vision solutions discussed also apply to the tugger, except in the contexts of lifting or dropping loads.

Architecture

A central computer coordinates the activities of all of the AGVs by communicating with them regularly over wireless Ethernet. The vehicles do not communicate directly with each other.

As shown in Figure 2, the software has two distinct components: a facility segment and a vehicle segment. One instance of the facility segment runs on the central computer. Each vehicle has its own instance of the vehicle segment.

Facility Segment

The facility segment is divided into on-line and off-line portions. Consider first the on-line portion. The Trailer Planner is used to generate the order in which loads are to be removed and installed. The Task Scheduler is used to allocate vehicles to tasks based on their capabilities and proximity to pick and drop locations. The Path Planner uses the A* algorithm to find the shortest path between any two nodes in the network

of guidepaths. The Execution Monitor tracks task completion and allocates guidepath intersections to vehicles on an exclusive basis in order to avoid collisions.

The off-line portion includes three elements. The Mosaic Editor produces globally consistent mosaics of the factory floor. A mosaic is a large-scale composite picture that is created from many individual pictures, each representing a small section of the floor. The process of creating mosaics (mosaicking) is described later in this article. The Calibration Routines calibrate sensor poses (pose of sensors with respect to the vehicle coordinate frame) and camera lens distortion parameters. The Network Editor produces and modifies the guidepaths to which the robots are nominally confined during motion.

The planning of vehicle motions must respect certain constraints and policies. Tuggers cannot drive backward because they cannot push on their loads. Fork trucks should, however, always drive backward on long hauls. Their LADARs are rear mounted, preventing them from being occluded when the forks carry a load. Fork trucks must address a load in a forward manner, so opportunities to turn around must be built into their guidepaths.

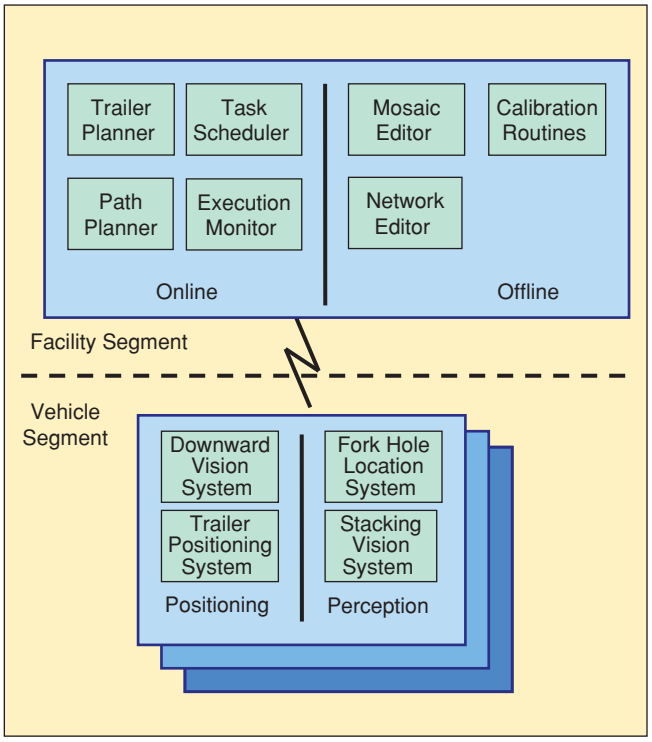

Figure 2. *System architecture. The facility software segment runs on a computer that controls the activities of all AGVs by communicating with them over wireless Ethernet. Each vehicle runs the algorithms in the vehicle segment.*

Vehicle Segment

The vehicle segment is divided into vision-based positioning systems and perception systems.

There are two positioning systems. The Downward Vision System uses floor mosaics as navigation maps for guiding the vehicle through the factory. The Trailer Positioning System uses LADAR to locate the vehicle with respect to the walls of a trailer.

There are also two perception systems. The Fork Hole Location System visually detects fork holes in loads. The vehicle uses this system to find the position of loads relative to the forks and to position the forks appropriately for picking up the loads. The Stacking Vision System enables racks to be stacked. It computes the position of a rack that is currently loaded on the forks with respect to a rack that is on the floor, enabling their legs to be aligned for proper stacking.

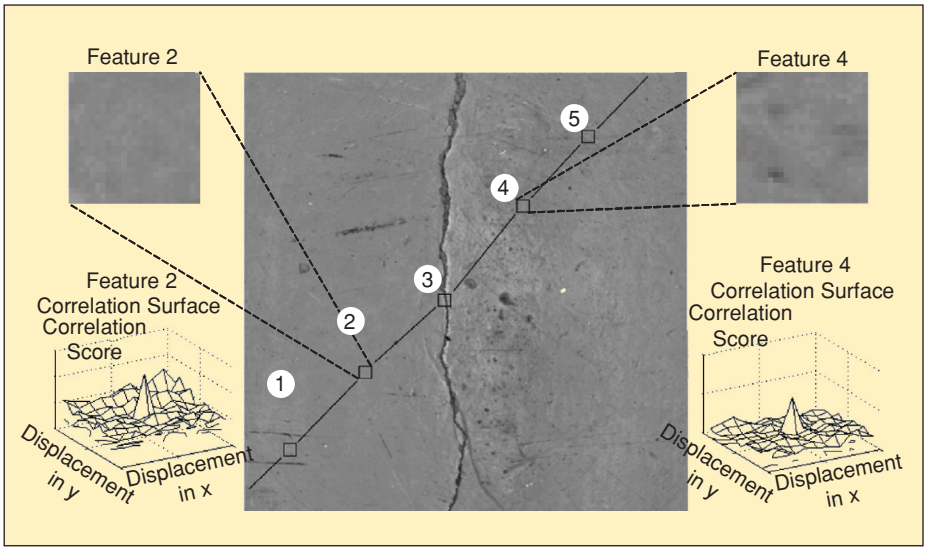

Figure 3. *Concrete floor feature correlation. The concrete floors typically found in factories and warehouses exhibit enough texture that normalized cross correlation can be used to track specific locations on the floor as a camera moves over them. The above image is a typical image of a concrete floor. Five features (25 × 25 pixel windows) are selected randomly on a diagonal line. Of these, the two with the weakest texture scores are selected to show that even these are good features. For each, an autocorrelation surface is computed by displacing a copy of the window with respect to itself in both directions and computing the correlation. A peak in the correlation surface that is strong and unique over the search range means the feature can be distinguished readily from all points nearby, even in the presence of noise. It is not uncommon to find that a window around any point in such an image is a good feature.*

Vision Systems

This section describes the four vision systems mentioned above that comprise the vehicle segment. It also discusses the mosaic creation process.

Downward Vision System

The core capacity that enables AGVs to move autonomously in a factory is

their capacity to localize themselves relative to the building. Our goal was to develop an infrastructure-free, free-ranging guidance system. Our approach for achieving this goal was both highly unconventional and highly successful [9]. We used image mosaicking techniques to generate a large-scale visual record of the appearance of the floor. This composite image served as a navigation map in a visual tracker.

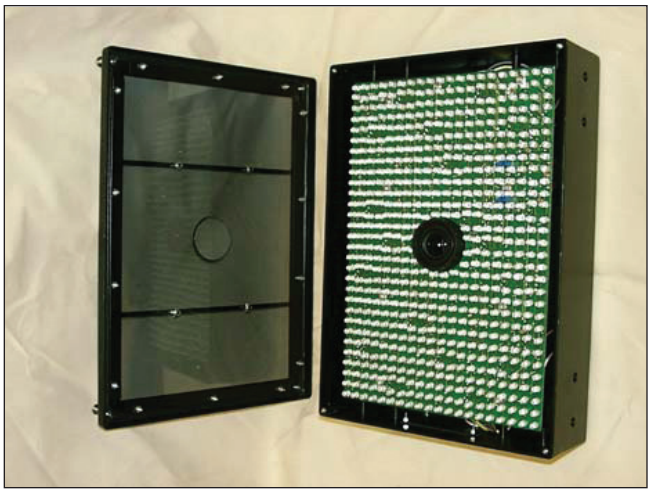

Figure 4. *Lighting and imaging module (smallest size). A standard machine vision camera is combined in this module with custom lighting and polarization filters. Control electronics modulate the LED intensities and synchronize the lighting with the camera shutter.*

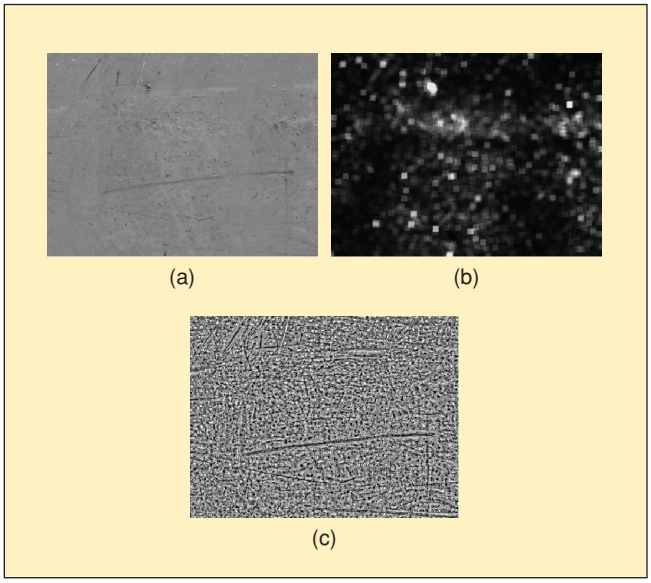

Figure 5. *Image preprocessing. Two important processes that are applied to images are shown. (a) A typical image of a concrete floor. (b) Texture scores for the image. Point discolorations and scratches have high scores but linear features do not. These scores are used to decide which places in the images should be matched to the mosaic. (c) A statistically normalized version of the input image. Statistical normalization is the first step in a normalized cross correlation computation.*

Several fundamental observations motivated our use of this approach. First, we observed that floor texture is rich in landmarks. Most factory floors exhibit visual texture on the millimeter scale that is both persistent and locally unique. This texture may have esthetic or operational purposes, or it may result from normal wear and tear, or both. Bare or transparently coated concrete is the most common floor surface. This type of flooring is generally covered in cracks, scratches, discolorations, and stains, all of which are persistently visible (Figure 3).

Second, vision algorithms are sufficiently mature. Visual tracking can be rendered highly reliable given a good estimate of motion between successive images, simplified scene geometry, and lighting control. All of these factors are achievable for a camera that is mounted beneath an AGV for the purpose of imaging a flat floor.

Third, sufficient storage is affordable. Typical camera resolutions image a 100 cm^2 area at a resolution of 0.2 mm per pixel. After reducing this resolution to 2.5 mm, 1 GB of offline storage (such as flash disk) can store detailed uncompressed imagery of a guidepath that is 6.25 km long and 1 m wide. Such imagery can also be highly compressed for feature tracking by storing only descriptions of the important features.

Fourth, processing requirements are feasible. Odometry can estimate motion between successive images to an accuracy of one pixel. Hence, only a very minimal amount of searching is needed in a visual tracker that maintains a visual lock on the mosaic.

Our concept for the guidance system is based on the idea that a high-resolution image of floors can be used as a navigation map. A map of sufficiently high resolution can be constructed by mosaicking a few hundred thousand images into one smooth, globally consistent image. The position and orientation of the vehicle can then be found by tracking the motion of the camera over this visual map.

Major technical challenges to this approach included constructing the imaging module, developing a localization algorithm, and developing a globally consistent mapping algorithm. These challenges are discussed in this section.

Imaging Module

The darkness underneath a vehicle presents both an opportunity to control lighting and the difficult problem of actually doing so. The floor clearance of an indoor vehicle is typically on the order of 10 cm. We were unable to find a way to diffuse a point light source sufficiently well over the 90° field of view required to illuminate a 20-cm diameter area, so we spread the light source itself into an LED array as shown in Figure 4.

This device incorporates several important features. Modular components that support three sizes of lighting arrays are constructed from different numbers of the same components. A spatially modulated intensity pattern is used to create uniform subject illumination. Cross polarization filtering is used to eliminate saturation caused by specular reflections of the light source from shiny floors. This is a commonly used technique in industrial vision systems. Light exiting the LEDs

passes through a filter polarized at right angles to the filter in front of the camera. Since specular reflections preserve polarization, they will not be sensed, but light that is absorbed and re-emitted (diffuse reflections) will be sensed.

Precise synchronization of the illumination flash to the camera shutter produces a minimal lighting duty cycle that significantly reduces the energy drawn from the vehicle battery. A shutter open signal is generated to tell the position estimation CPU when to save the pose tag for the image. The pose tag is the pose of the vehicle at the instant the shutter was opened. It must be saved until the image arrives later at the main computer.

Localization

The localization algorithm solves the visual tracking [7] and pose refinement problem [6] as it occurs in the limited context of rigid motion of undistorted features in the image plane. Imagery produced on all vehicles is rectified to remove the effects of differences in camera mounting (pose of camera on the vehicle) and distortion in the wide-angle lenses. Once images are rectified, mosaics produced by one vehicle can be tracked by the other vehicles.

Following the method of [15], we examine the eigenvalues of a matrix of image intensity gradients in the input image to identify regions of bidirectional texture. Small rectangular windows around these points of high texture are the features that are used for image matching. Up to 16 well-separated features are used per image (Figure 5).

Small templates are matched, rather than the entire image, in order to limit sensitivity to rotational error. The decorrelation caused by rotational error is never large enough to require a search in orientation. However, orientation error is computed for each image based on the differential position errors of the templates.

Feature matching is a two-step process based on normalized cross correlation [3]. The first step in this process is statistical normalization:

$$x' = \frac{x - \mu_w}{\sigma_w}.$$

This step enhances texture by replacing each pixel intensity value x by its deviation from the mean intensity μ_w of a neighborhood around it. It then normalizes the result by dividing by the standard deviation σ_w of the intensity in the same neighborhood. To save computation, the mosaic is stored in this normalized form but real-time imagery must be normalized as soon as it is read.

The second step of feature matching computes the correlation for all possible displacements in a limited search region. The most recent vehicle pose estimate is used to predict the position of each feature in the mosaic and the search is centered there. A unique peak and a high correlation score corresponds to an unambiguous match (Figure 3). The difference between the true and predicted position in the mosaic of each correctly matched feature is then used in a Kalman filter to refine the position estimate.

Mosaic Editor and Globally Consistent Mapping

Imagine taking a set of overlapping floor images produced while a vehicle drives sufficiently slowly in a straight line. Designate the images with sequentially assigned indices $0, 1, 2 \ldots n$. Each successive pair of images $(i, i+1)$ is registered by matching the features that appear in both. Conceptually, image $i+1$ is moved relative to image i until the features in the region of overlap line up correctly.

The situation described so far produces a linear mosaic, but more generally, there is a need in our application to produce globally consistent mosaics whose final configuration is a network of guidepaths containing cycles that close correctly (Figure 6).

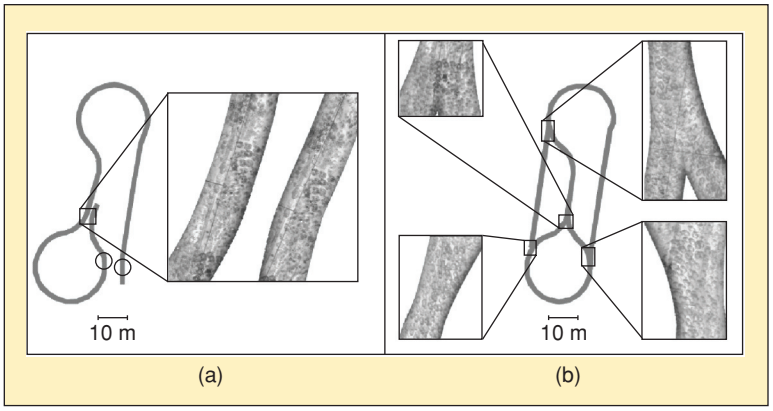

Figure 6. *Cyclic mosaicking. This relatively small 150-m long mosaic was produced from 1,836 floor images. (a) Before consistency enforcement, the two cycles have closure errors on the order of a meter. (b) After consistency enforcement, closure errors are on the order of 2 mm. An extra guidepath has also been added that closes a third cycle.*

The need to develop a globally consistent mosaicking solution arises from two concerns. First, the cumulative effect of small displacement errors (dead reckoning drift) causes large internal discrepancies at the point of closure of large cycles. Such discrepancies cannot be spanned by the largest possible search windows of the real-time visual tracker, so the robot will get lost if the discrepancies are not eliminated. Second, it is often desirable to distort the mosaic to force it to agree with externally derived guidepath descriptions or with factory schematics.

In principle, the imagery used to create mosaics can be produced by any vehicle. For the sake of efficiency, however, we typically used a special mapping vehicle employing a large, 1-m scale imaging module.

To solve the global consistency problem, we developed an algorithm that automatically formulates loop constraints for arbitrarily complex guidance networks. It sets up a system of loop constraints that are optimized and explicitly enforced to make sure that all loops close correctly [11], [12].

To avoid addressing the data association problem of loop closure, we had the human driver of the mapping vehicle use a custom graphical user interface to establish the necessary correspondences. Other researchers have developed automatic solutions for similar instances of this problem [5], [13], [14].

LADAR-Based Guidance from Trailer Walls

While the mosaic-based guidance system can be used in trailers, it requires that a large fraction of the trailer floor be mapped. A much simpler solution was available: LADAR.

The fork truck already used a laser rangefinder for obstacle avoidance. We therefore used this device to track the position of the fork truck inside the trailer by matching the endpoints of LADAR pixels to a rectangular model of the trailer geometry. If the trailer's dimensions were known, they were used to generate the model. If the dimensions were unknown, the vehicle used an initial LADAR scan to find the dimensions and generate the model.

Several occlusion issues arose in this application. If the trailer is full during loading or unloading, very little of its walls is visible. We solved this problem by imaging the walls of the loading area outside the trailer. To transition between the two guidance systems, the vehicle scanned both trailer and loading area before driving off the mosaic.

Fork Hole Finding Vision

The need to locate pallets and racks relative to the fork truck arose in two contexts: picking them up from the floor, and de-stacking stacked loads. Major technical challenges included designing the sensing configuration, developing the robot-load pose refinement algorithms, and developing the visual servo controllers used to position the forks inside the fork holes. The first two challenges are discussed in this section; the third is discussed later in this article.

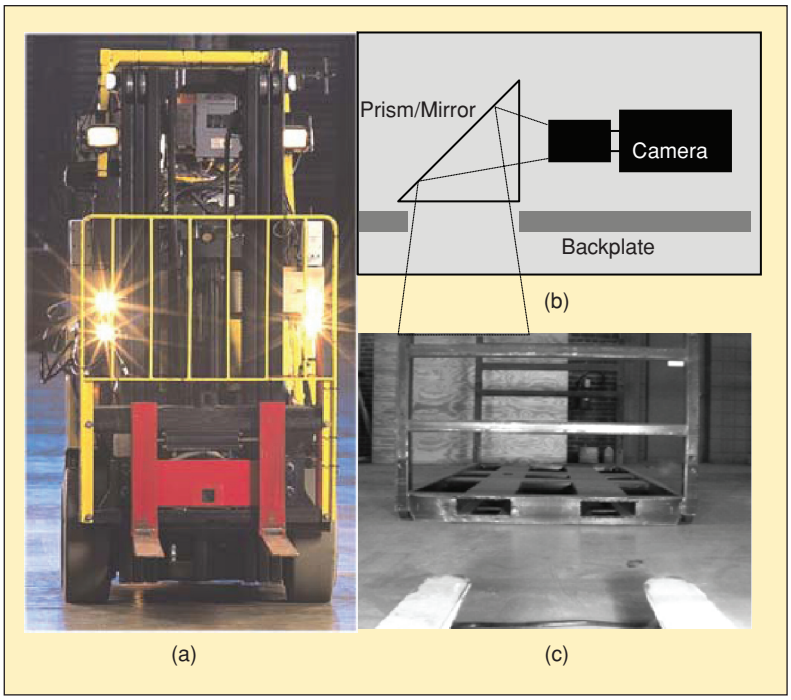

Figure 7. *Forward camera configuration. The fork camera is mounted behind the fork back plate. (a) The aperture in the plate is visible as the small square cut into the red plate. (b) Schematic of top view of the design configuration. (c) A view of an empty rack from this perspective.*

Sensing Configuration

In this problem, the location of the loads relative to the forks is the quantity of interest. The optimal place for the associated sensor would be on the forks themselves, allowing the sensor to move when the forks are actuated vertically or sideways by the hydraulics. Unfortunately, this would put the sensor in the most vulnerable possible position on the fork truck, where almost all forceful interactions with the environment take place.

Our solution (shown in Figure 7) was to place a camera behind the existing steel back plate and bend its field of view through a 90° angle using a small flat mirror. The mirror was located behind a rectangular hole in the back plate and imaging took place through a Plexiglas viewfinder. Both the mirror and the viewfinder were inexpensive and easy to replace.

Pose Refinement

Our approach to finding the position of loads relative to the fork-mounted camera was based on three assumptions. CAD models of the specialized parts racks used in automobile manufacturing are assumed to be available. Rack recognition is assumed to be unnecessary because the robot will know when to look for a rack. The estimate of the robot's position is assumed to be accurate to 20 cm in position and 30° in heading. This reflects the accuracy with which a human truck driver might have originally placed the load.

Our pose refinement algorithm used work performed at JPL for computer vision algorithms for the International Space Station [16]. This approach matches line segments in a CAD model to those detected in imagery by using an edge detector (Figure 8). Sufficient observations are obtained to enable a simultaneous and continuous calibration of the camera model.

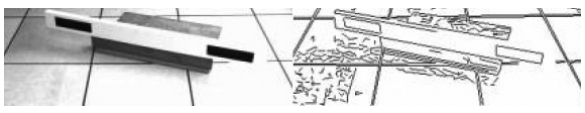

Figure 8. *Finding fork holes. Image edges are matched to a CAD model of the parts rack in order to compute its pose relative to the camera. The figure shows an example processed image. A Canny-Lowe edge detector was used.*

The monocular vision system was easily able to compute the lateral position of a load. Finding the range of the load was more difficult but less important because limit switches would tell the truck when the forks were completely inserted.

The yaw of the load relates to the difference in the ranges of each side. The system required a good estimate of yaw to determine where to position the truck. This challenge is discussed later in this article.

Stacking Vision

The purpose of the stacking vision system is to compute the position of a rack on the forks with respect to another rack on the floor, enabling the two racks to be stacked. The bottoms of the four legs of the top rack must fit into slightly oversized receptacles on top of the legs of the bottom rack. Clearance was on the order of 2 cm. Rack sizes were as large as 2 m deep and 4 m wide.

Analysis suggested we would barely be able to achieve this resolution with our cameras even after formulating the following strategy for maximizing precision [8].

Direct visual feedback was used to avoid exposure to errors in kinematic models relating indirect measurements to the quantity of interest. We were also able to exploit the principle of differential measurement to generate a degree of robustness to parameter calibration errors throughout our models. The error geometry was favorable because the cameras were arranged at an angle of approximately 90° to each other. The four basic observations of high-quality lateral position and low-quality range overdetermined the three degrees of pose freedom relating the two racks in the plane.

The same pose refinement system used for finding fork holes was used here. The camera simultaneously imaged both the top and bottom racks (Figure 9) enabling direct measurement of displacement from a single image. This direct differential measurement was insensitive to many errors in camera calibration. Two cameras were used: one pointing left, the other right. Only the two front pairs of rack legs were visible. However, we were able to align the left and right front legs well enough to cause the two rear legs to be automatically aligned as well.

The prototype system used retro reflective fiducials placed on the legs. A subsequent design iteration intends to replace the reflectors with LED line generators, creating a structured light system. The better illumination that the LED-based lighting system provides will allow the system to position the racks without needing reflectors on the legs.

Trajectory Generation

It soon became clear during the execution of the program that we would need a good solution for controlling the posture of our vehicles to position them precisely enough for this application. The original motivation for solving this problem was that of pallet pickup by the fork truck. Pallets can only be picked up when addressed from a posture that places the fork tips at the fork holes with the right heading and with zero curvature (Figure 10).

When a vision system determines the location of the fork holes, the goal posture may not be known until limited space

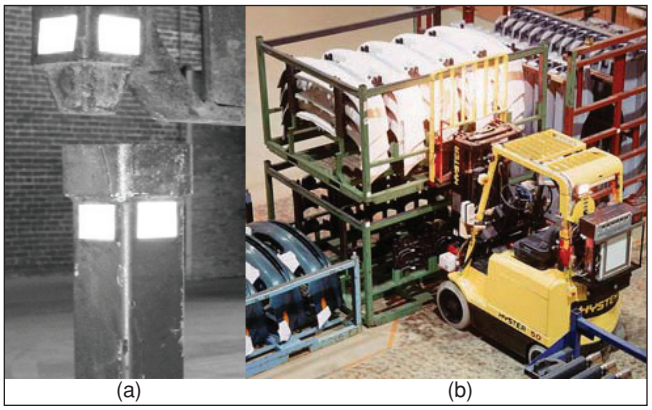

Figure 9. *Stacking two parts racks. (a) A close-up of two legs that must be aligned. (b) Fork truck about to align the load it is carrying with the one on the floor. Four such alignments are needed for the racks to stack correctly.*

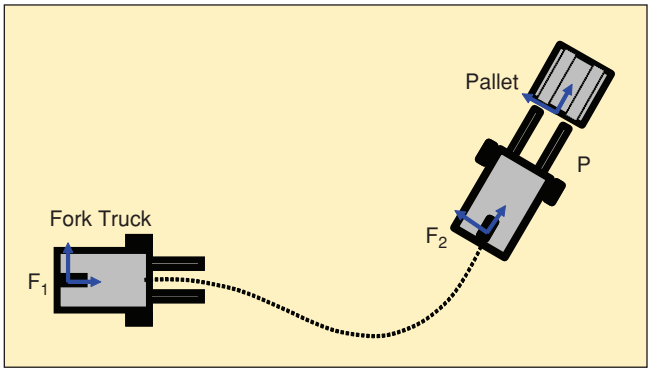

Figure 10. *Pallet pickup. To successfully pick up a pallet, a fork truck must achieve a fairly precise target posture characterized by position, heading, and zero curvature.*

requires an aggressive maneuver to address the load correctly. The problem (as shown in Figure 10) is that of determining a feasible motion to connect frame F_2 to frame F_1, given a measurement of the relationship between frame P and frame F_1.

The trajectory of the vehicle is represented as curve called a polynomial spiral whose curvature is polynomial in distance:

$$\kappa(s) = a + bs + cs^2 + ds^3.$$

The system computes trajectories that satisfy constraints on initial and final vehicle positions, headings, and curvatures [9]. Acceptable performance could only be achieved by using good initial guesses from which to iterate to a solution. Initial guesses for a sampling of all trajectories are stored in a precomputed lookup table that is interpolated when accessed later. In order to generate initial guesses for the construction of the lookup table, it is computed in a manner that reuses the last nearby result to seed the search for the next (Figure 11).

The implemented algorithm was able to generate any feasible motion for the vehicles in under a millisecond of computation. Accuracy in achieving terminal states was a millimeter in position and a milliradian in heading.

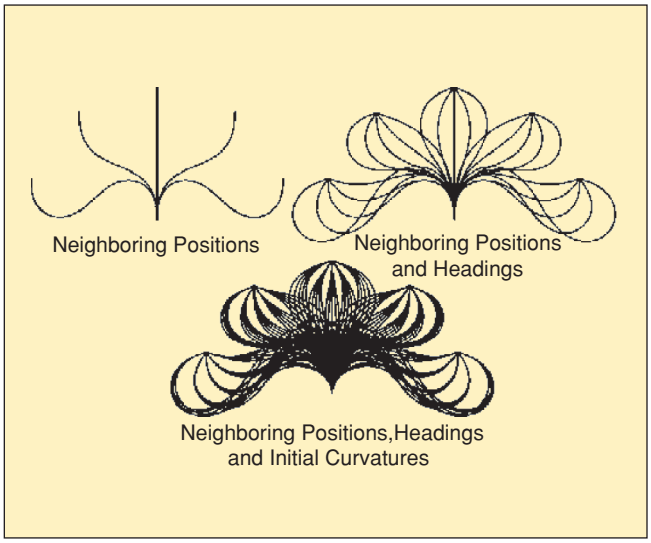

Figure 11. *Computing trajectory lookup tables. A lookup table to be used for initial guesses is generated by slowly scanning the goal posture through state space in such a way that each solution is always very close to the last.*

The basic capacity to drive a vehicle to a goal posture has many uses. Once a solution to the basic problem was in place, other applications for it became clear immediately.

A second use for the algorithm was the generation of the representation of the guidepath network. Typical AGV guidepaths are expressed as combinations of lines and arcs (and, more rarely, linear curvature polynomials known as clothoids). In all these cases, the point where two such primitives join is likely to have a discontinuous curvature that is not feasible for a real vehicle to execute because the steering mechanism cannot change position instantaneously.

Cubic polynomial spirals are the simplest curves that can be continuous in curvature where arbitrary primitives join. We developed a user interface for drawing polynomial spiral guidepaths over the guidance mosaics in order to exploit this continuity property. An added advantage was that of achieving, by construction, consistency between the guidepaths and the mosaic. Otherwise, we would have had to calibrate the two to agree.

Once such paths are specified, a third use of polynomial spiral primitive motions is for corrective trajectories in path following. We developed a path-following algorithm based on cubic spiral trajectories with two desirable properties. First, corrective trajectories reacquire the target path at the correct heading and curvature. Second, the point of reacquisition is generated by searching along the path for the best solution, thereby adapting the effective gain to instantaneous vehicle state and trajectory curvature.

This algorithm typically ran at a rate of 10 Hz and it achieved lateral path following errors under 1 cm.

Visual Servo Controllers

Having discussed both vision systems that measure vehicle pose relative to both the plant and objects of interest as well as a mechanism to generate feasible motions to arbitrary terminal postures, this section discusses how both elements are used together to cause purposeful robot motion.

Since all forms of guidance are based on a form of vision, our vehicles operated continuously in one visual servo controller or another. Four visual servo controllers can be distinguished based on whether the state estimation was derived from floor vision, LADAR scanning of trailer walls, fork hole vision, or stacking vision. Each visual servo controller enables the vehicle to execute a desired motion that is specified as a polynomial spiral. Several challenges had to be addressed to make this approach practical, as described below. Between visual updates (which arrived at a frequency of a few Hz), the system drove based on higher frequency odometry information.

Generating Guidepaths

For gross motions from one place in the plant to another, a predefined guidepath network was used to specify the roads of legal travel for our vehicles. By contrast, local motions to somewhat uncertain load positions could only be defined once the load was in sight. The specification of these motions was performed on-line once vision informed motion planning of the target vehicle posture.

While our vehicles had the capacity to deviate laterally from their guidepaths to avoid obstacles, safety policy prohibits this in many commercial settings. The next section assumes such path deviation is not allowed.

Guidepath Following

We developed a guidepath following algorithm that continuously tries to reacquire the desired path at some slightly forward position on the path. Corrective trajectories are generated on a continuous basis to drive the following error to zero as quickly as possible.

Both the guidepath and the corrective trajectory are polynomial spirals. One distinguishing feature of our approach is that the corrective trajectory matches the guidepath exactly in position, heading, and curvature at the point of path reacquisition.

At some point during execution, the guidepath may come to an end where the vehicle must achieve the state at the endpoint as closely as possible. The mechanism of following a forward point then breaks down since there is no forward point. Our approach here was to simply follow the rest of the last generated corrective trajectory open loop using odometry alone. The magnitude of the cross track error before opening the loop (< 1 cm) and the short length of the open loop section made this strategy practical.

Goal Instability

For visually generated guidepaths, an additional issue is noise in the vision system. The most difficult aspect of this problem proved to be computing the yaw of a load to be picked up relative to the fork truck. The situation is aggravated by the fact that if the yaw of the load changes, the correct pickup posture moves sideways. We typically first estimated load pose when the fork tips were about 3 m from the load.

In rough terms, yaw estimates improve as the vehicle approaches the load. Continuing to use vision to refine the goal posture is advisable but this also means that the controller finds itself trying to achieve a moving target. To make matters worse, the path length available to move to correct for yaw and lateral misalignment also decreases rapidly as the load is approached. Achieving success amounts to a computational race between observability and controllability—that of refining estimates of where to go before running out of time to get there.

Our system was able to meet its specification for load position and orientation errors but was very brittle beyond them. One effective approach for out-of-spec loads was to back off and try again from a new start point based on the previous best rack position.

Trailer Operations

We originally intended to attempt both automated trailer unloading and loading. However, we were only able to demonstrate unloading in a proof-of-principle context. Figure 12 shows the fork truck in the process of unloading a trailer. We addressed the context of using fork trucks to unload parts racks of nominal geometry, arranged predictably up to two wide in trailers (also of nominal geometry).

One major difference between loading and unloading is that during loading, the sensors used to pick up the load are likely to be occluded while driving into the trailer. In unloading, the fork truck has a clear field of view in which it can search for the fork holes using the fork hole finding algorithm described above.

Clamping Forks

The width of our racks was 5 cm less than the internal width of the trailer. Often, parts racks are designed such that the fork holes are oversized with respect to the cross section of the forks. The resulting uncertainty in the pose of the rack relative to the fork truck was unacceptable given operating wall clearances on the order of 2.5 cm on each side of the rack.

Given a choice between measuring where the rack is on the forks or forcing the rack to be in a specified position, we picked the latter approach. A special clamping fork assembly was retrofitted onto our fork truck. It could either squeeze or separate the forks while applying significant force. The parallel rectangular channels of the fork holes become predictably aligned with the vehicle frame within a few millimeters after application of the clamping force.

The fork truck-rack assembly then becomes a single rigid body of known geometry, and the problem of guiding both out of the trailer reduces to one of producing a position estimate of adequate accuracy.

Results, Outlook, and Conclusions

Our efforts to produce an infrastructure-free AGV have pursued multiple directions at once. The various elements have reached differing levels of maturity and performance. Our floor mosaic guidance system has achieved sufficient maturity to be proven in an auto assembly plant. In the final qualification test, we demonstrated reliable operation on a guidepath network exceeding 1 km in total length, over a total time of just less than 100 hours after traveling a total distance of 110 miles. During this test, 900,000 floor images were processed and the system was unable to establish a visual lock on only three of them. During such events, for a mere one-tenth of a second, the system navigated solely and reliably based on odometry. Then it reacquired visual lock in the next iteration of the tracker. Repeatability of 1 mm and speeds up to 10 mph were routinely achieved in our own 50,000 ft^2 facility.

In our facility, the system has operated on four different vehicles over a five-year period. These vehicles successfully shared a common mosaic, which at times was several years old. The floor was often far dirtier than a manufacturing facility would be allowed to become.

The globally consistent mapping work has been adapted in straightforward manner from camera imagery to LADAR scans. It has been applied on scales as large as hockey arenas and grocery stores to produce LADAR-based guidance maps [10].

Fork hole and stacking vision systems were demonstrated on a regular basis both in our facility and at Ford Motor Company. These elements were not placed in a production setting for testing based only on decisions of how to best prioritize our efforts. Trailer unloading was demonstrated several times at our test facility.

The trajectory generation algorithm has been continuously improved since its original development to adapt it to arbitrary vehicles, complex dynamic models including wheel slip, and even arbitrary rough terrain. The algorithm is currently in use at Carnegie Mellon University on multiple programs as the basis for many efforts in path following, obstacle avoidance, and search space generation for discrete motion planning in extremely cluttered environments. It is the basis of many elements of our off-road robotics programs including DARPA-funded UGVs and NASA-funded planetary rovers.

Figure 12. *End unloading. The robot is removing the rack of auto parts from the trailer through the loading door. Note the lights of the visual guidance system under the robot.*

Our goal was to explore the potential of vision-enabled, automated guided vehicles. While it is no surprise to the robotics research community that vision enables environmental and situational awareness for robots, it probably is significant to know that mobile robot vision can be deployed in a factory for several weeks without experiencing any failures. The AGV industry has been slowly adopting vision of its own accord for some time. For example, a LADAR-based pallet finding system appeared on the market during our execution of the program. Hopefully, our efforts provide an example of what a fully vision-guided AGV might be able to do in a more highly automated facility in the future.

Acknowledgements

This work was conducted at the Robotics Institute of Carnegie Mellon University under contract to NASA and Ford Motor Company as part of the National Robotics Engineering Center.

Keywords

AGV, factory automation, material handling, computer vision, motion control.

References

[1] *A Personal Guide to Automated Guided Vehicle Systems.* Charlotte, NC: Material Handling Industry of America, 2004.

[2] G. Bekey, R. Ambrose, V. Kumar, A. Sanderson, B. Wilcox, and Y. Zheng, "International Assessment of Research and Development in Robotics," World Technology Evaluation Center, Jan. 2006.

[3] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

[4] C. Dejong (July 1999), "Material handling pop quiz," *Automot. Design Product*. [Online]. Available: www.autofieldguide.com/articles/079905.html.

[5] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 229–241, June 2001.

[6] D.B. Gennery, "Visual tracking of known three-dimensional objects," *Intl. J. Comput. Vision*, vol. 7, no. 3, pp. 243–270, 1992.

[7] S. Hutchinson, G.D. Hager, and P.I. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 651–670, Oct. 1996.

[8] A. Kelly, W. Kim, and D. Helmick, "Model-based object pose refinement for terrestrial and space autonomy," in *Proc. 6th Int. Symp. Artificial Intelligence, Robotics, and Automation in Space* (ISAIRAS 01), Montreal, Canada, June 2001.

[9] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *Intl. J. Robot. Res.* vol. 22, no. 7–8, pp. 583–601, 2003.

[10] A. Kelly, "Mobile robot localization from large scale appearance mosaics," *Intl. J. Robot. Res*, vol. 19, no. 11, pp. 1104–1125, Nov. 2000.

[11] R. Unnikrishnan and A. Kelly, "A constrained optimization approach to globally consistent mapping," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (IROS 2002), Lausanne, Switzerland, pp. 564–569, Sept. 2002.

[12] A. Kelly and R. Unnikrishnan, "Efficient construction of optimal and consistent LADAR maps using pose network topology and nonlinear programming," in *Proc. 11th Int. Symp. Robotics Research* (ISRR 2003), Sienna, Italy, Nov. 2003.

[13] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *J. Intell. Robotic Syst.*, vol. 18, no. 3, pp. 249–275, 1997.

[15] J. Shi and C. Tomasi, "Good features to track," *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 593–600, July 1994.

[16] W.S. Kim, "Computer vision assisted virtual reality calibration," *IEEE Trans. Robot. Automat.*, vol. 15, pp. 450–464, June 1999.

[17] Frost and Sullivan (May 1999), "World AGV and AEM Markets," [Online]. Available: http://www.frost.com.

Alonzo Kelly received the B.A. Sc. degree in aerospace engineering from the University of Toronto in 1984, the B.Sc. in computer science from York University in 1990, and the Masters and Ph.D. degrees in robotics from Carnegie Mellon University in 1994 and 1996, respectively. He has worked in the robotics industry for Spar Aerospace and the Jet Propulsion Laboratory. He is now an associate professor at the Robotics Institute of Carnegie Mellon. His research focuses on perception, planning, control, guidance, and user interfaces for mobile robots designed for both structured indoor and unstructured outdoor environments.

Bryan Nagy received his B.S. in computer science in 1998 from Carnegie Mellon University, followed by his M.S. in robotics in 2000. His research interests include mobile robot systems, multiagent task allocation and range-data based computer vision. He is Principal Research Programmer at NREC, where he has developed planning, navigation, and perception systems and techniques for a variety of robotic applications, including agricultural field container handling, an industrial autonomous material transport system consisting of robot forklifts and towmoters, and various outdoor off-road mobile robots. Currently he is working on TraderBots, an application independent market-based task allocation system.

David Stager received a B.S. in math/computer science and electrical and computer engineering in 1995 from Carnegie Mellon University. He is a senior commercialization specialist at National Robotics Engineering Center; a division of the Robotics Institute at Carnegie Mellon University. His research focuses on robotic systems design, autonomous vehicles, and custom robotic platforms.

Ranjith Unnikrishnan graduated from the Indian Institute of Technology, Kharagpur, with the BTech degree (Hons.) in electronics and electrical communication engineering in 2000. He received the M.S. degree from the Robotics Institute at Carnegie Mellon University in 2002 for his work on automated large-scale visual mosaicking for mobile robot navigation. He is currently pursuing the Ph.D. degree at the Robotics Institute, working on extending scale theory to problem domains such as nonuniformly sampled 3-D data and vector-valued 2-D images and developing new low-level vision algorithms in those domains. His research interests also include the development of performance metrics for vision algorithms and new techniques for fast laser-camera calibration.

Address for Correspondence: Alonzo Kelly, Robotics Institute, Carnegoe Mellon University, Pittsburgh, PA 15213-3890. E-mail: alonzo@ri.cmu.edu.