Active Perception and Robot Vision, Sood, A., ed., Springer-Verlag: Heidelberg, 1991.

Real-Time Vision for Autonomous and Teleoperated Control of Unmanned Vehicles

Martin Herman, James S. Albus and Tsai-Hong Hong

Robot Systems Division
National Institute of Standards and Technology
(formerly National Bureau of Standards)
Gaithersburg, MD 20899
USA

ABSTRACT

This paper focuses on two related topics: a control system architecture that unifies autonomous and teleoperated control of unmanned vehicles, and examples of how real-time vision processing fits into this architecture. The NIST hierarchical real-time control system architecture and its application to unmanned vehicles is presented. The paper then discusses recent work at NIST in real-time vision for both teleoperated and autonomous vehicles. For teleoperated vehicles, we describe a system for video compression for low data rate remote vehicle driving. For autonomous vehicles, we describe passive range extraction from optical flow for applications such as target extraction and identification, vehicle driving, and terrain mapping. We also describe how each of these vision systems fits into the control system architecture.

1. Introduction

This paper focuses on two related topics: a control system architecture that unifies autonomous and teleoperated control of unmanned vehicles, and examples of how real-time vision processing fits into this architecture.

Unmanned vehicles are typically divided into two types, autonomous and teleoperated. However, it is often more convenient to think in terms of degrees of autonomy. A vehicle

where the remote operator has direct control of the motor or actuator drivers, i.e., control of the individual thruster drive currents, would be the least autonomous. On the other hand, a vehicle that has independent control of its own mission definition, or that could alter strategy and mission priorities, would be the most autonomous.

For the purpose of this paper, we will classify unmanned vehicles as follows:

1. Teleoperated: Type A -- With this vehicle, the remote operator has direct control of steering, brakes, throttle, and transmission.
2. Teleoperated: Type B -- The remote operator indicates short, obstacle-free motion pathways to the vehicle. The vehicle autonomously computes dynamically efficient movements that accomplish these paths.
3. Supervised Autonomous -- The remote operator indicates a list of intermediate, key poses that accomplish some vehicle task. The vehicle autonomously performs functions such as obstacle avoidance, road following, object approach, etc. which are required to drive the vehicle between consecutive key poses. A vehicle that performs these kinds of functions autonomously is typically classified as supervised autonomous.
4. Autonomous -- The remote operator indicates vehicle tasks such as follow-road, enter-building, rendezvous, search-an-area, etc. The vehicle autonomously computes intermediate key poses (the inputs to supervised autonomous vehicles) required to achieve these tasks. The vehicle also performs the functions performed by supervised autonomous vehicles.

A control system architecture that unifies autonomous and teleoperated control is the National Institute of Standards and Technology (NIST) Real-time Control System (RCS) architecture [1, 2, 4, 7, 17]. The system is a three legged, multi-level hierarchy of computing modules for task decomposition, world modeling, and sensory processing. This hierarchy is serviced by a communications system and a distributed common memory, and is interfaced to operator and programmer workstations. The architecture supports both teleoperated and autonomous control.

This paper first describes the NIST Control System Architecture. It then focuses on recent work at NIST in real-time vision for both teleoperated and autonomous vehicles. For teleoperated vehicles, we describe a system for video compression for low data rate remote vehicle driving. For autonomous vehicles, we describe passive range extraction from optical flow for applications such as target extraction and identification, vehicle driving, and terrain mapping. We also describe how each of these vision systems fits into the control system architecture.

2. Real-Time Control System Architecture

In the NIST Control System Architecture (Figure 1), the task decomposition hierarchy decomposes long range goals into drive signals to actuators. Task goals are decomposed both spatially and temporally. Task decomposition is accomplished by performing real-time planning, execution, and task monitoring.

The sensory processing hierarchy interprets the world at multiple levels of abstraction. This is accomplished by filtering, correlating and integrating sensory information over both space and time so as to detect, recognize amd measure patterns, features, objects, events, and relationships in the external world.

The world model hierarchy serves as an interface between the sensory processing and task decomposition hierarchies. The world modeling modules answer queries, make predictions, and compute evaluation functions on the state space defined by the information stored in global memory. Each level in the task decomposition hierarchy may access a number of different levels in the world model hierarchy. The global memory is a database which contains the system's best estimate of the state of the external world. The world modeling modules keep the global memory database current and consistent.

Operator interfaces allow operator access to any level of the hierarchy.

2.1. The Task Decomposition Hierarchy

Each module in the task decomposition hierarchy receives input commands from one and only one supervisor, and outputs subcommands to a set of subordinate modules at the next level down in the tree. Outputs from the bottom level consist of drive signals to motors and actuators.

The kinds of processing that occur at each level of the task decomposition hierarchy are described next. The reader should refer to Figures 1 and 2. Figure 2 shows processing that occurs in the lowest four levels for locomotion.

1. MISSION LEVEL -- Inputs to the mission level consist of commands to one or more groups of vehicles to perform a specific mission. The function of the mission level is to assign vehicles to groups, set priorities for group actions, and assign mission objectives to the groups. The activities of the groups are scheduled so as to maximize effectiveness of the mission. The outputs of the mission level are group subtasks and priorities, and these are transmitted to the next lower level in the hierarchy.
2. GROUP LEVEL -- Inputs to this level are group task commands which define actions to be performed cooperatively by groups of vehicles on multiple targets. This level decomposes group tasks into individual vehicle tasks, which are transmitted to the level below. The decomposition typically assigns to each vehicle a prioritized list of

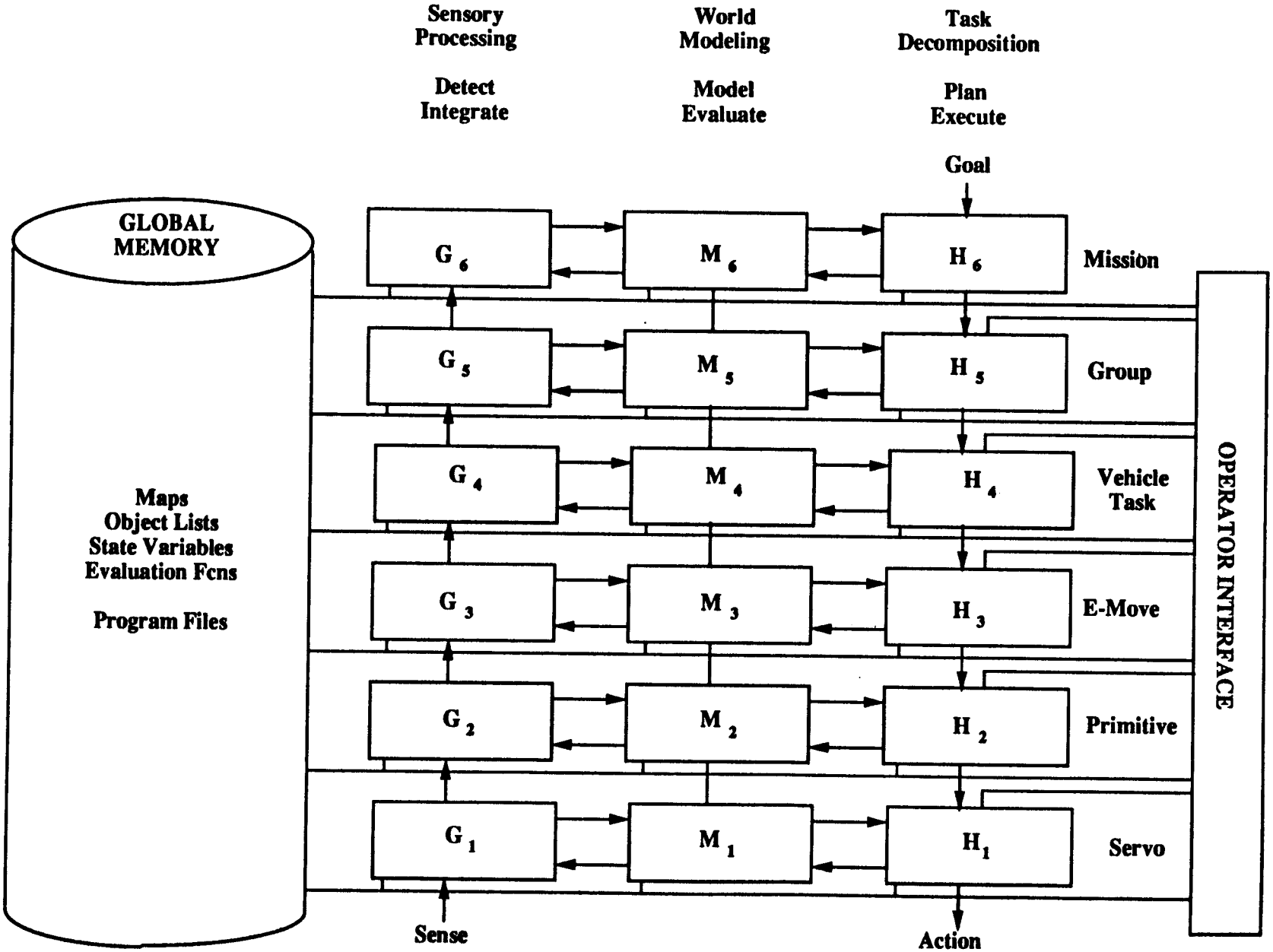


Figure 1. Real-time control system architecture for unmanned vehicles.

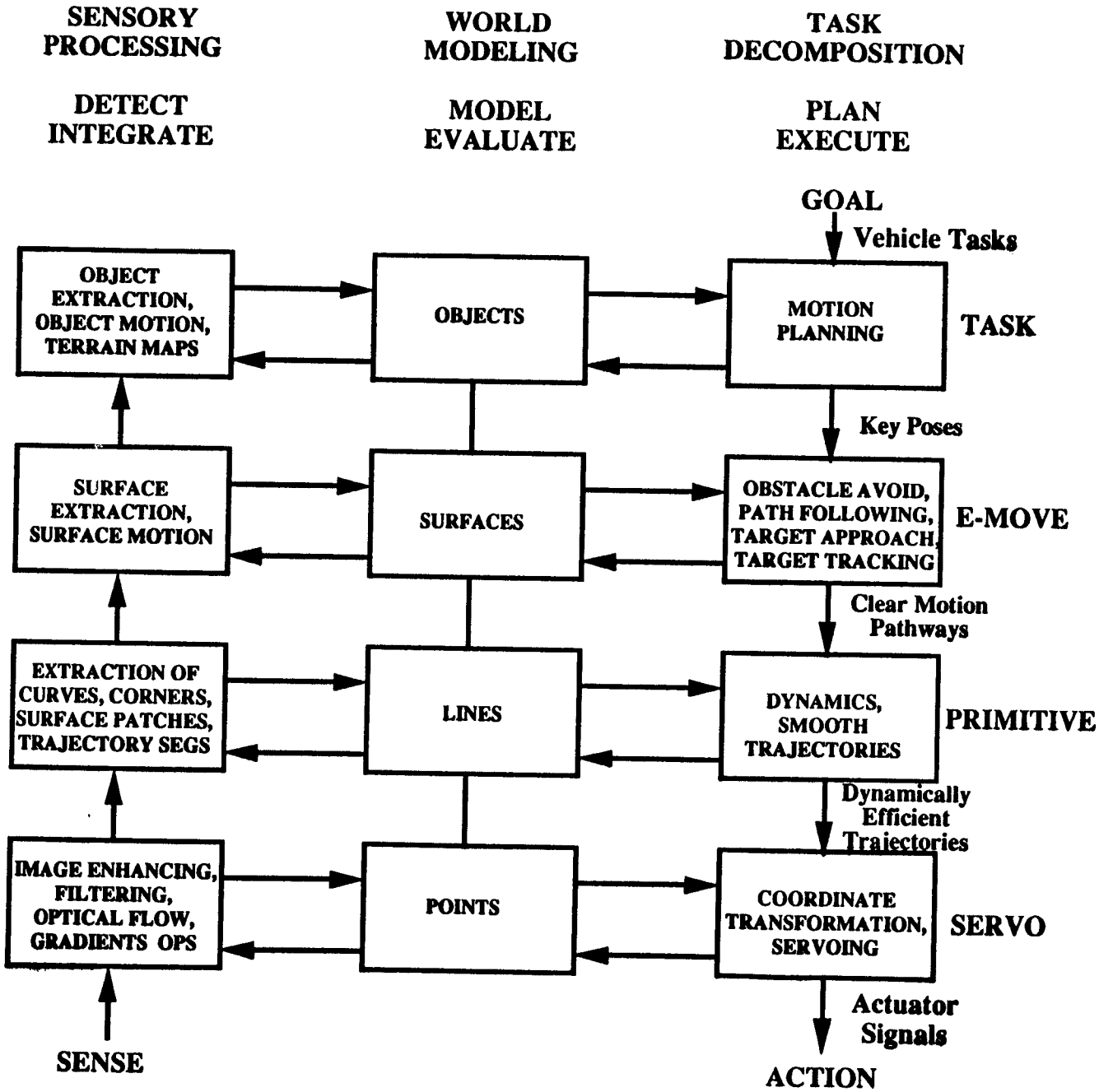


Figure 2. Classes of processing in the task decomposition and sensory processing hierarchies.

tasks to be performed on or relative to one or more other vehicles, objects, or targets. Tactics and vehicle assignments are selected to maximize the effectiveness of the group's activity. The actions of each vehicle are coordinated with the other vehicles in the group so as to maximize the effectiveness of the group in accomplishing the group task goal.

3. VEHICLE LEVEL -- Input task commands to this level define actions to be performed by a vehicle on, or relative to, a single object or other vehicle that is a target of attention. The function of the vehicle level is to decompose the input vehicle task into a sequence of tasks for each subsystem of the vehicle. These subsystem tasks are called elemental moves or actions (e-moves). Examples of subsystems are locomotion, sensors, and communication. This level must coordinate, synchronize and resolve conflicts between vehicle subsystem plans.

   For locomotion, this level performs motion planning by using maps to assure that there exists a clear pathway for movement between planned poses of the vehicle relative to the target. The cost, risk and benefit of various routes are evaluated, and a path that maximizes a cost-benefit evaluation function is chosen. For locomotion, the outputs of this level are key poses (involving position, orientation, velocity, etc.) representing clear pathways.

4. E-MOVE LEVEL -- The input to the e-move level is a command which is an elemental move or action involving a single subsystem. For example, a locomotion e-move command will specify some key pose to be achieved by the vehicle in space or time. This level determines clearance with obstacles sensed by on-board sensors and generates sequences of intermediate poses that define clear motion pathways.

5. PRIMITIVE LEVEL -- The primitive level computes inertial dynamics and generates smooth, dynamically efficient trajectory positions, velocities and accelerations. Inputs to this level consist of intermediate trajectory poses which define a path that has been checked for obstacles and is guaranteed free of collisions. The outputs of this level consist of evenly spaced trajectory points which define a dynamically efficient movement.

6. SERVO LEVEL -- The servo level transforms coordinates from a vehicle coordinate frame into actuator coordinates. This level also servos thruster direction and actuator power. Inputs to this level consist of commanded positions, velocities, thrust, power, orientation, and rotation rates of the vehicle. Outputs of this level consist of electrical voltages or currents to motors and actuators.

An example of task decomposition for vehicle locomotion is shown in Figure 3. The input to the task level is the command "Follow road to point P." The task level performs motion planning and decomposes this command into a sequence of intermediate key poses,

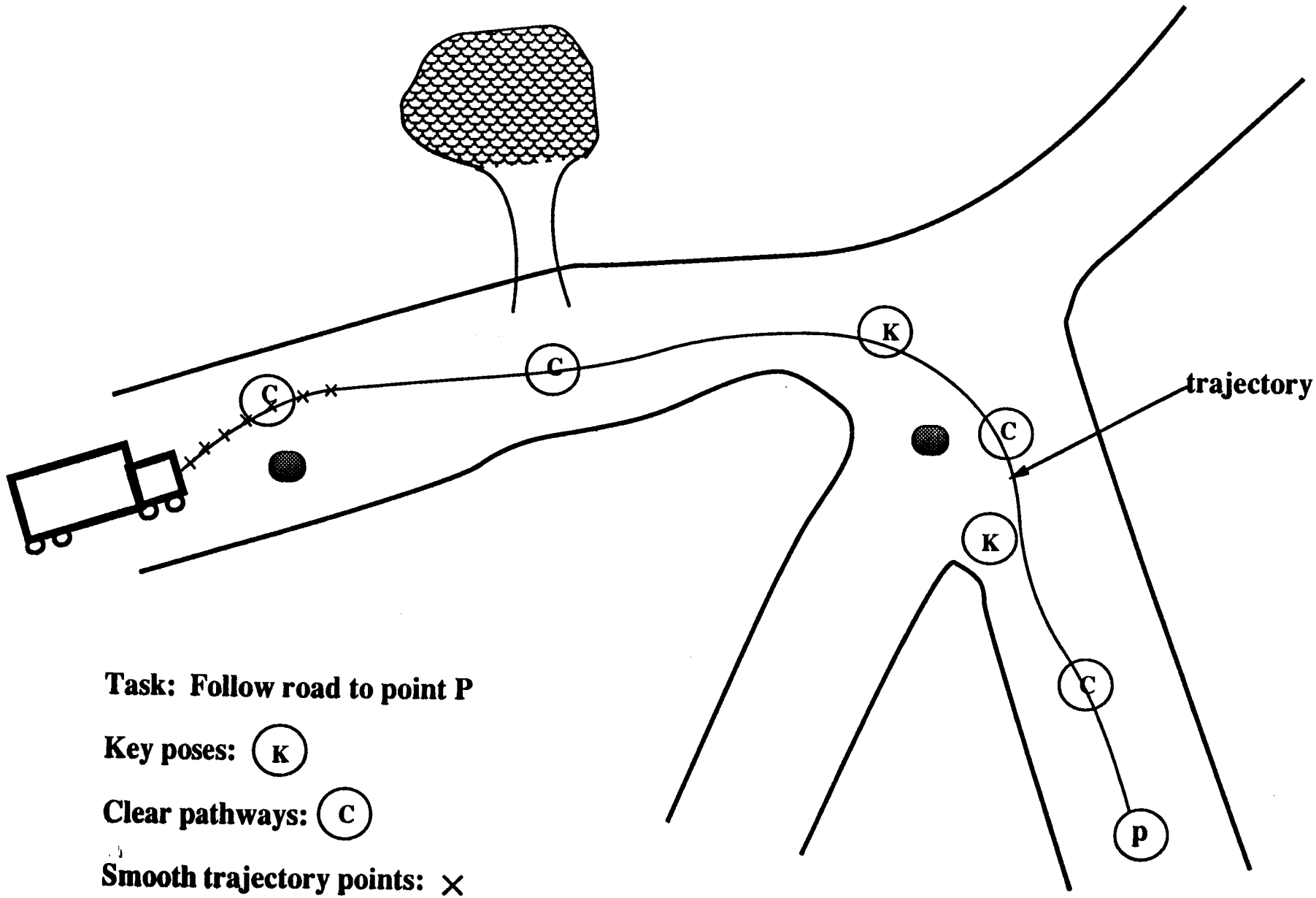


Figure 3. Task decomposition for vehicle locomotion.

represented by the set of points K in the figure. The planning at this level considers large-scale terrain, features, and obstacles. The intermediate key poses are sent sequentially to the e-move level (Figure 2), which decomposes each into a set of intermediate clear pathways represented by the set of points C in Figure 3. The decomposition performed at this level involves activities such as obstacle avoidance and path following. This level considers local terrain, features, and obstacles.

The clear motion pathways are sent sequentially to the primitive level (Figure 2), which decomposes each into a set of intermediate trajectory points represented by the set of points X in Figure 3. These trajectory points represent a smooth path which is obtained from the clear pathway by taking into account the inertial dynamics of the robot. The trajectory points are sequentially sent to the servo level which executes the motion by servoing the robot to these points.

2.2. Sensory Processing Hierarchy

This section describes the vision processing that occurs in the sensory processing hierarchy. The reader should refer to Figures 1 and 2. Figure 2 shows processing that occurs in the lowest four levels.

1. SERVO LEVEL -- Inputs to the servo level come directly from the pixels of a camera. The primary function of this level is to detect spatial and temporal gradients and, if possible, to compute a range value for each pixel. The image processing algorithms applied at this level assume the input to be an individual pixel (together with its spatio/temporal neighborhood), while the output is a processed pixel. Examples of the kinds of processing that occur at this level include preprocessing such as image enhancement, thresholding, contrast enhancement, smoothing, averaging, median filtering, low-pass filtering, sharpening, and high-pass filtering. Also included at this level are feature extraction techniques such as boundary points extraction, gradient extraction, template matching, texture extraction, optical flow extraction, and range from optical flow.
2. PRIMITIVE LEVEL -- Inputs to this level come from level 1. Sensory processing at this level recognizes linear features in space-time. Patterns of points in space-time are integrated into linear features such as lines, curves, edges, corners, and vertices, and into small surface patches. Range from stereo is computed here. Temporal patterns of moving points are segmented into temporal linear features such as trajectory segments and optical flow field boundaries, and into small flow field patches.
3. E-MOVE LEVEL -- Inputs to this level arrive from level 2. Sensory processing at this level recognizes surfaces in space-time. Surface attributes such as surface shape, texture, position and orientation, velocity, rotation, and deformation are determined.

4. VEHICLE LEVEL -- Inputs to this level come from level 3. Sensory processing at this level recognizes objects and volumes in space. Object attributes such as shape, size, texture, position, and orientation are determined. Object motion such as translation, rotation, velocity and smooth trajectories are determined. Terrain maps are also determined.
5. GROUP LEVEL -- Inputs to this level come from level 4. Sensory processing at this level recognizes descriptions of group relationships and group dynamics of objects, such as group center of gravity, density, spatial distribution, group membership, and group boundaries.
6. MISSION LEVEL -- Sensory inputs to this level come from level 5. Sensory processing deals with relationships between groups, and recognizes properties of the world that extend beyond the immediate neighborhood of the self group.

3. Video Compression for Remote Vehicle Driving

Remotely driving a ground vehicle involves an operator who sits at a remote control center and views video images that are transmitted from one or more cameras mounted on the vehicle. While observing these images, the operator drives the vehicle by means of driving controls involving steering, brakes, throttle, and transmission. These controls generate appropriate driving actuator signals which are transmitted to the vehicle.

In order for the operator to effectively drive the vehicle, the video images must be of sufficient quality and must be updated as frequently as possible. Full rate video transmission from the vehicle to the operator requires about 60 megabits per second for 512 x 512 images with 8 bits per pixel at 30 frames per second. However there are several problems with using the wide communication bandwidth required for such transmission. First, wide bandwidth radio communication requires direct line of sight between the transmitter and receiver. This is not feasible in realistic outdoor scenarios where vehicles are likely to be driven behind hills and mountains and therefore hidden from direct view by the operator station. Second, wide bandwidth links are relatively expensive. Finally, full rate video uses up a large part of the bandwidth allocations. This could present a problem if there are many vehicles being operated simultaneously, since wide bandwidth links for the vehicles could combine to exceed the entire communication spectrum. Fiber optic tethers, which have wide bandwidth communications capabilities, also have several problems, including limited ruggedness, difficulties in deployment and retrieval, and the problem of repairs.

Many of these difficulties can be overcome by utilizing narrow band radio links which have communication bandwidths on the order of 100 kilobits per second or less. Since the full video desired for teleoperation cannot be transmitted over narrow band links, efficient and effective techniques of real-time video compression offering compression ratios of

500:1 to 1000:1 must be developed.

3.1. Requirements for Remote Driving

A vision system used for remote driving can be evaluated in terms of how well it permits the remote operator to perform vehicle mobility operations. There are various factors that affect vehicle mobility. The first involves the ability of the operator to make trafficability and movement decisions. These decisions involve the following elements:

1. Detection of local obstacles such as depressions in the ground (e.g., holes), solid objects on the ground (e.g., rocks), vegetation (e.g., heavy brush), and tall objects (e.g., trees and telephone poles).
2. Classification of local terrain surfaces such as concrete, grass, snow, and water.
3. Determination of local terrain slopes.
4. Recognition of local landmarks such as trees, rocks, and telephone poles.
5. Local path planning.
6. Local path following, e.g., following roads, rivers, or vegetation.

A second factor that affects vehicle mobility involves the ability of the operator to visually maintain a sense of the global vehicle location relative to the background and landmarks.

A third factor involves the ability of the operator to visually maintain a sense of the motion parameters of the vehicle, such as speed, acceleration, and turning rate.

In order to allow the remote operator to effectively perform vehicle mobility operations, two requirements should be met. First, the video images transmitted from the vehicle should be displayed in real time. This means that the operator should have the appropriate visual information quickly enough so that he can drive the vehicle interactively. Driving is typically a servoing operation which requires a fast loop involving transmitting driving commands to the vehicle, getting fast visual feedback, and using this feedback to transmit additional driving commands.

A second requirement is that the video sequence that appears on the operator's monitor should seem natural (e.g., smooth and continuous).

These requirements should be satisfied whether the vehicle is driven with full video or with compressed video. A disadvantage of using video compression algorithms is that they take processing time, and many algorithms result in degraded imagery. A challenge for video compression techniques is therefore to meet all the requirements set forth above, that is, to provide real-time video, to provide a natural video sequence, and to provide imagery which is accurate and detailed enough so that the operator can make mobility decisions.

3.2. Approach

To solve the problem of video compression for remote driving, we have proposed a hybrid method which combines image processing compression (i.e., techniques whose input is an image and whose output is a compressed image), discrete cosine transform compression, and temporal frame rate reduction (i.e., transmitting much less than 30 frames per second). The sequence of events as they would occur in the system is as follows. Images are obtained from one or more cameras mounted on-board the vehicle. These images then undergo compression using the hybrid technique. After the compressed code is transmitted over a communication link to the operator station, it is decompressed so as to result in a sequence of full resolution images.

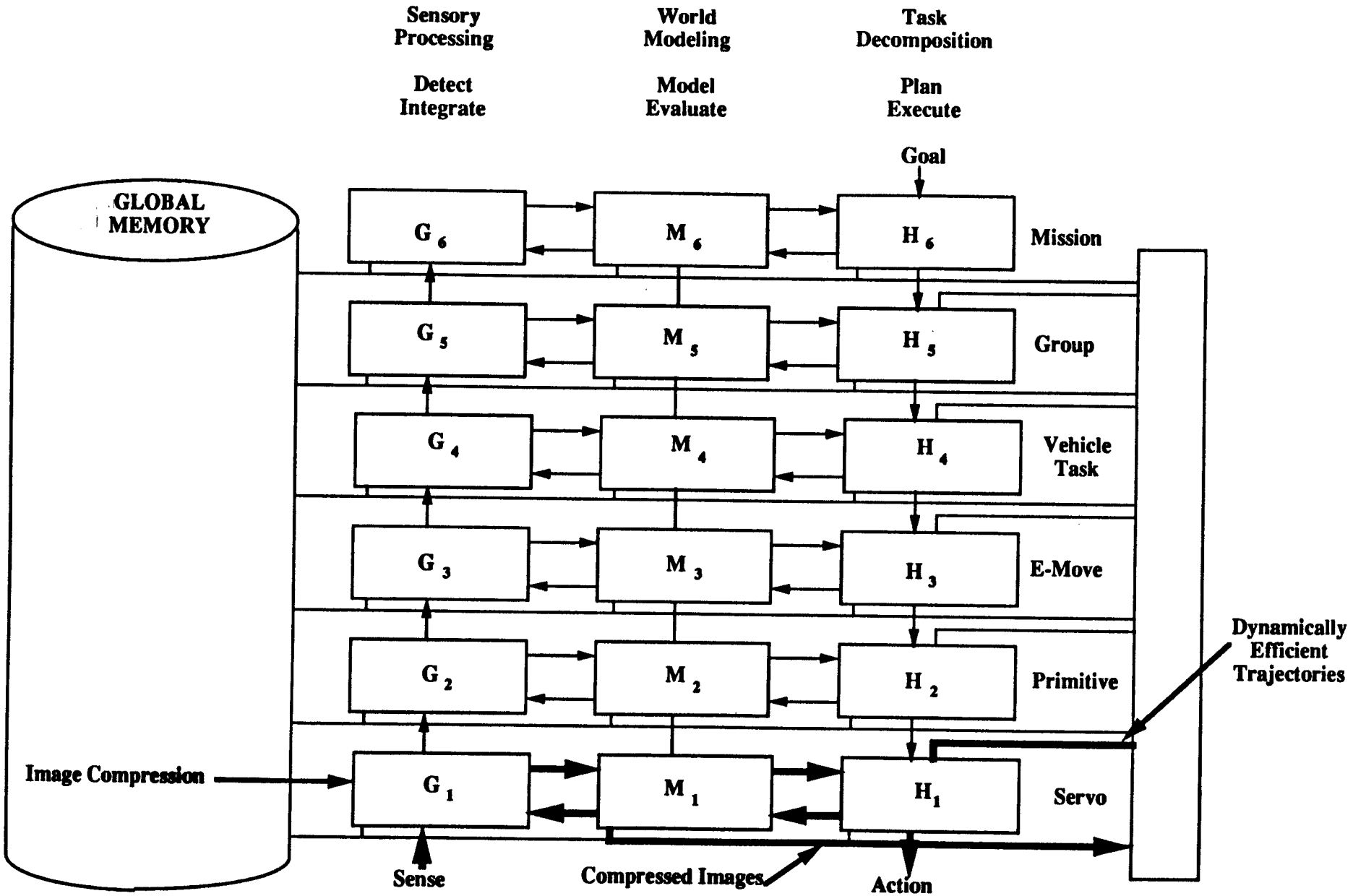
Several algorithms for the image processing compression portion of this method have been implemented to run in real time on the PIPE image processing machine [11] and have been used in actual field experiments for teleoperated control. The algorithms implemented on PIPE include grey level quantization, non-maximal edge suppression, foveal-peripheral simulation, image differencing, histogram slicing, binning, Laplacian pyramid decomposition and reconstruction, decimation and Poisson interpolation, and linear predictive coding. Details describing these algorithms and their implementation on PIPE are presented in [8].

This system fits into the NIST unified control system architecture at the servo level (Figure 4). The sensory processing component of the system performs image compression. The compressed images are transmitted to the operator and displayed on his monitor. The operator then looks at the displayed video to drive the vehicle by transmitting servo level commands to the vehicle. These commands, in the form of steering, brake, throttle, and transmission signals, represent dynamically efficient trajectories.

3.3. Towards Video Compression Using Simulation

Several of the compression algorithms running on PIPE were tested during real-world remote driving experiments. PIPE was set up at the remote operator station, using as input video returning from the vehicle. PIPE then performed real-time video compression on the input images, and the results were displayed on the monitor and used by the operator to drive the vehicle. Experiments were performed in a paved loading dock area surrounded by buildings, on an uneven grassy field, and on a dirt field on which obstacles were placed. The main difficulties that were encountered during the experiments, particularly when driving in open terrain, were

1. global vehicle location relative to the background and landmarks was very difficult to determine,



TELEOPERATED DRIVING 1

Figure 4. Teleoperated driving using compressed images. Thick lines indicate relevant communication pathways.

2. the slope of the local terrain was very difficult to determine,
3. ditches, gullies, rocks, and other obstacles were difficult to distinguish,
4. range from the vehicle to objects and terrain features were difficult to determine.

Object and terrain feature segmentation was very difficult in monochrome images. This was particularly true in the open field where the main way of discriminating between objects and the field was by color -- the field was covered with green grass and weeds, while objects and obstacles were brown, silver, or red. Further details on the results of these experiments are described in [9].

From these experiments, we have concluded that the critical element in achieving teleoperated driving control is to minimize the degradation of the imagery transmitted to the operator. Unfortunately, most compression techniques that achieve significant compression ratios result in significantly degraded imagery. In order to limit the degradation, we propose a technique involving the use of simulation to achieve real-time compression. The idea here is to transmit imagery at a very low frame rate (perhaps only a few times per second or once every few seconds), and to have a realistic simulation at the operator station which simulates the intermediate imagery that would appear in the vehicle camera. In order to achieve realistic simulation, both an accurate scene model and accurate navigation data (position, velocity, and acceleration of the vehicle) are required.

The technique of video compression using simulation involves transmitting an image (called the *master* image) relatively infrequently (perhaps once every few seconds), but then warping the master image in real time to generate a sequence of simulated images for the operator [14]. These images are meant to approximate those that would appear in the camera on the vehicle. When a new image arrives from the vehicle, it becomes the new master image and the procedure is repeated.

The basic concept is shown in Figure 5. Let c_0 be the pose of the camera when the master image is taken, and let c_i be the pose of the camera at some future point at which we want to generate a simulated image. For some pixel p_0 in the c_0 image, we know that the point P_s in the scene corresponding to this pixel lies along a ray from the camera focal point through the pixel p_0 out into the scene. In fact, the point P_s is at the intersection of this ray with the first surface it meets. If we know the position of this surface from a scene model, then we can determine P_s. The pixel p_i in c_i corresponding to pixel p_0 in c_0 is simply obtained by the intersection of the line containing P_s and the focal point of c_i with the image plane of c_i. Pixel p_i will then be given the same grey level value as pixel p_0. In practice, the point p_0 corresponding to pixel p_i will lie in between pixels. Therefore, the grey value assigned to pixel p_i will be obtained by interpolating pixel values in a small neighborhood around p_0. When this procedure is carried out for every pixel in c_i, we have the resulting warped image.

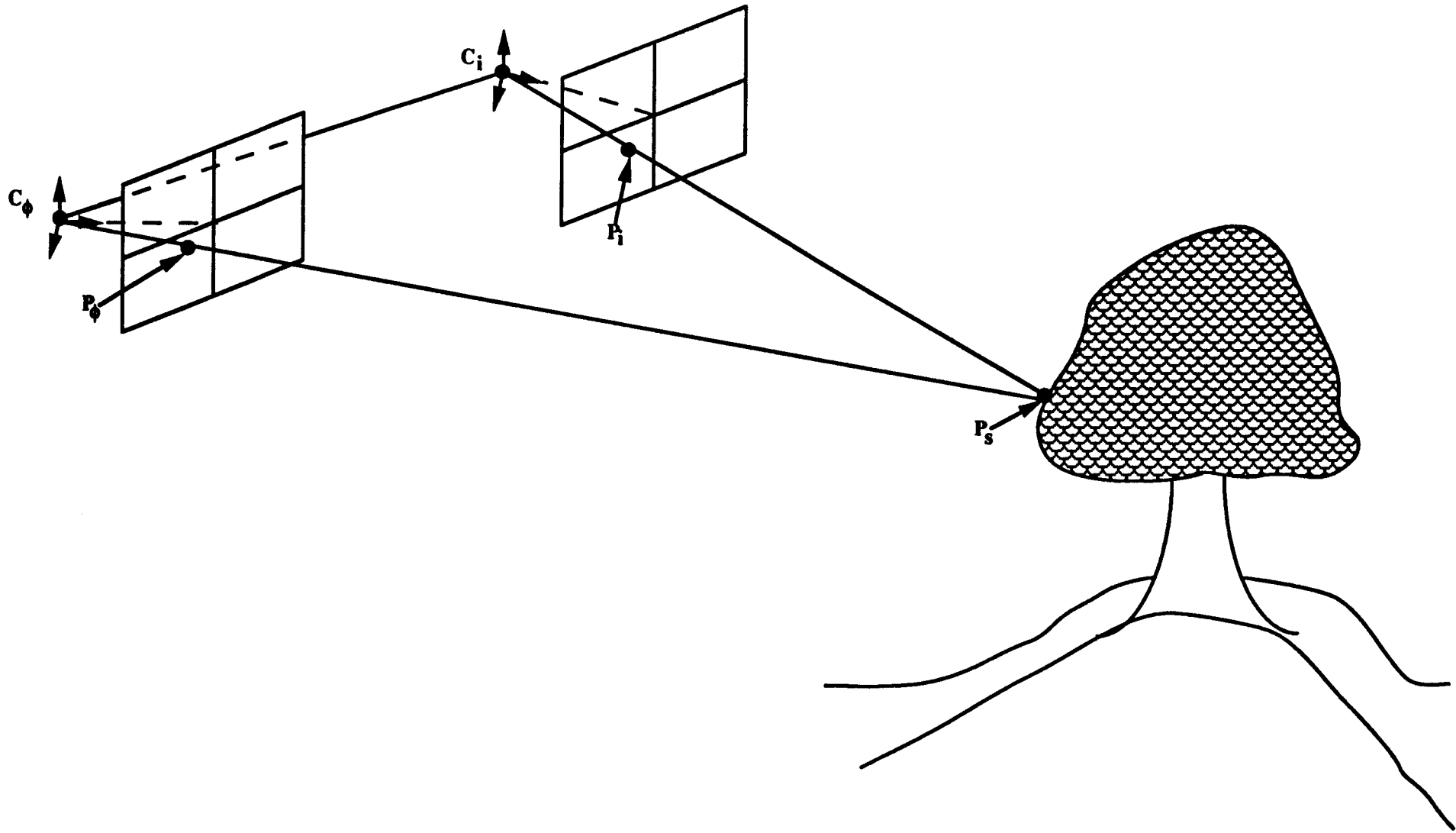


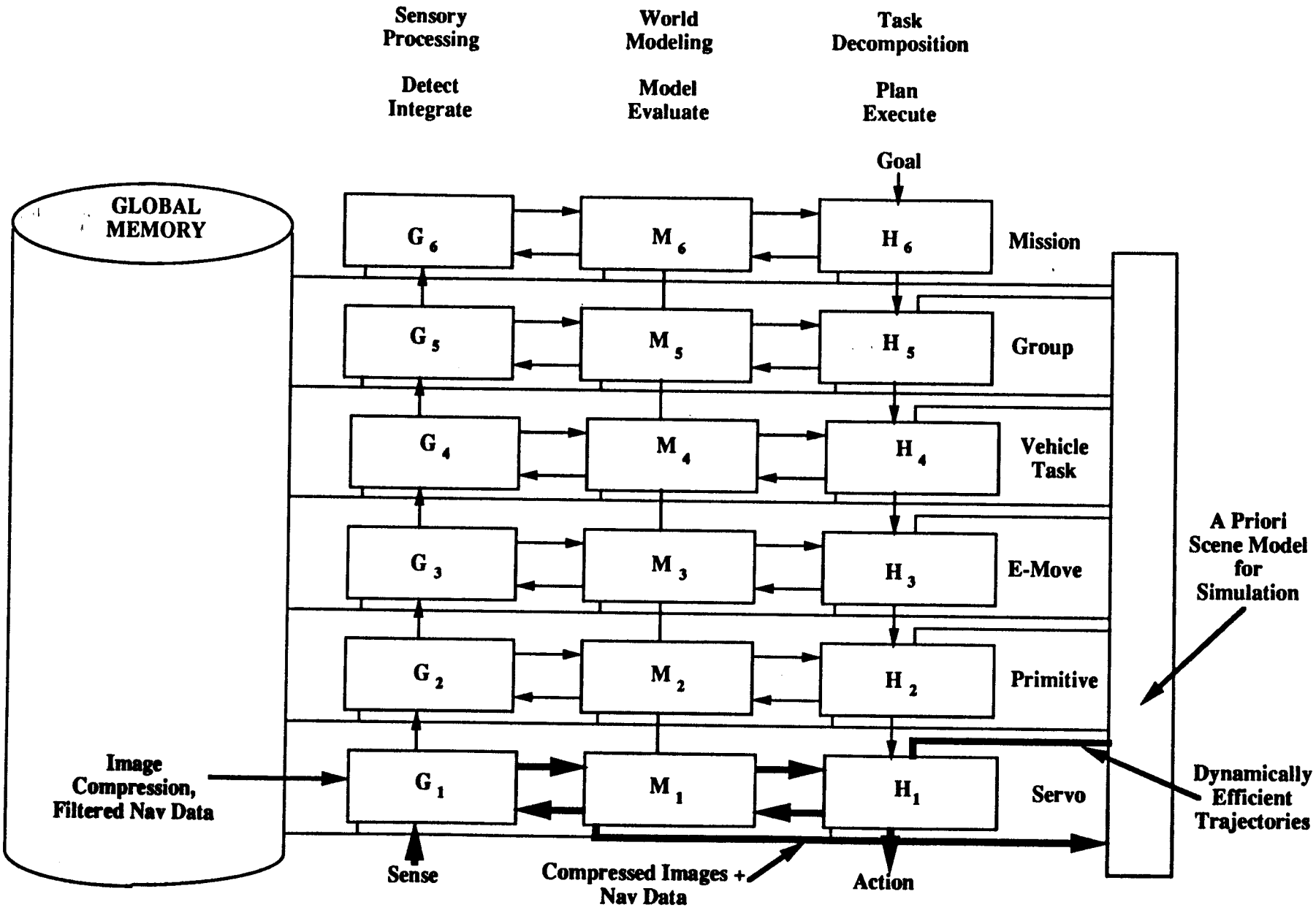
Figure 5. Geometry for generating simulated imagery.

Briefly, the operation of such a system is as follows. A video image is transmitted from the remote vehicle to the operator station at the rate of, say, 1 frame every 3 seconds. Time-tagged vehicle navigation data is transmitted from the vehicle continuously at 30 Hz. These data are used to estimate the position and orientation of the camera in the scene during the simulated-views. Time-tagged 3-D scene information may also be transmitted once every 3 seconds. Simulated images are then generated at 30 Hz from the master image and from the navigation data and scene information. These images are displayed to the operator and provide the approximate current vehicle scene. Because the vehicle's pose is always known to a high accuracy, all vehicle motions can be reflected instantly in the operator display. A Kalman filter is applied to the navigation data so that a smooth extrapolation is obtained to the point in time at which the image is to be displayed at 30 Hz.

Of fundamental importance to this approach is the ability to acquire, represent, and efficiently access a 3-D scene model. There are two types of scene models to consider. One is an a priori scene model, such as a flat earth model or digital terrain elevation model. The other is a scene model which is dynamically generated.

The simulation technique using an apriori model fits into the NIST unified control system architecture at the servo level as follows (Figure 6). The sensory processing component performs image compression and filters the navigation data. Compressed image frames (the master images) are periodically transmitted to the operator. Navigation data are transmitted very frequently, perhaps at 30 Hz. (Transmission of navigation data at this rate does not use up much bandwidth.) At the operator station, simulation is achieved by combining the navigation data with the a priori scene model and the master image as shown in Figure 5. The simulated imagery is used by the operator to drive the vehicle by transmitting servo level commands to the vehicle.

The proposed simulation technique using a dynamically generated scene model involves using real-time passive ranging from optical flow to obtain an accurate model. This model is represented in the form of range images. Techniques for extracting this kind of information are presented below. The advantages of a dynamic model over an apriori model are: (1) if the apriori model uses simplifying assumptions (e.g., a flat earth) or if it uses currently available databases (e.g., digital terrain elevation data), it will not be very accurate, (2) the dynamic model can incorporate the most recent changes in the environment. Teleoperated driving using this simulation technique fits into the NIST unified control system architecture (Figure 7) in a manner very similar to the method using simulation with an a priori model (Figure 6). The only difference is that the sensory processing component also computes optical flow and extracts a 3-D range image from the optical flow. This is done at the servo level. The range image is transmitted periodically to the operator along with the compressed master image. Navigation data are transmitted very frequently.



TELEOPERATED DRIVING 2

Figure 6. Teleoperated driving using an apriori 3-D scene model. Thick lines indicate relevant communication pathways.

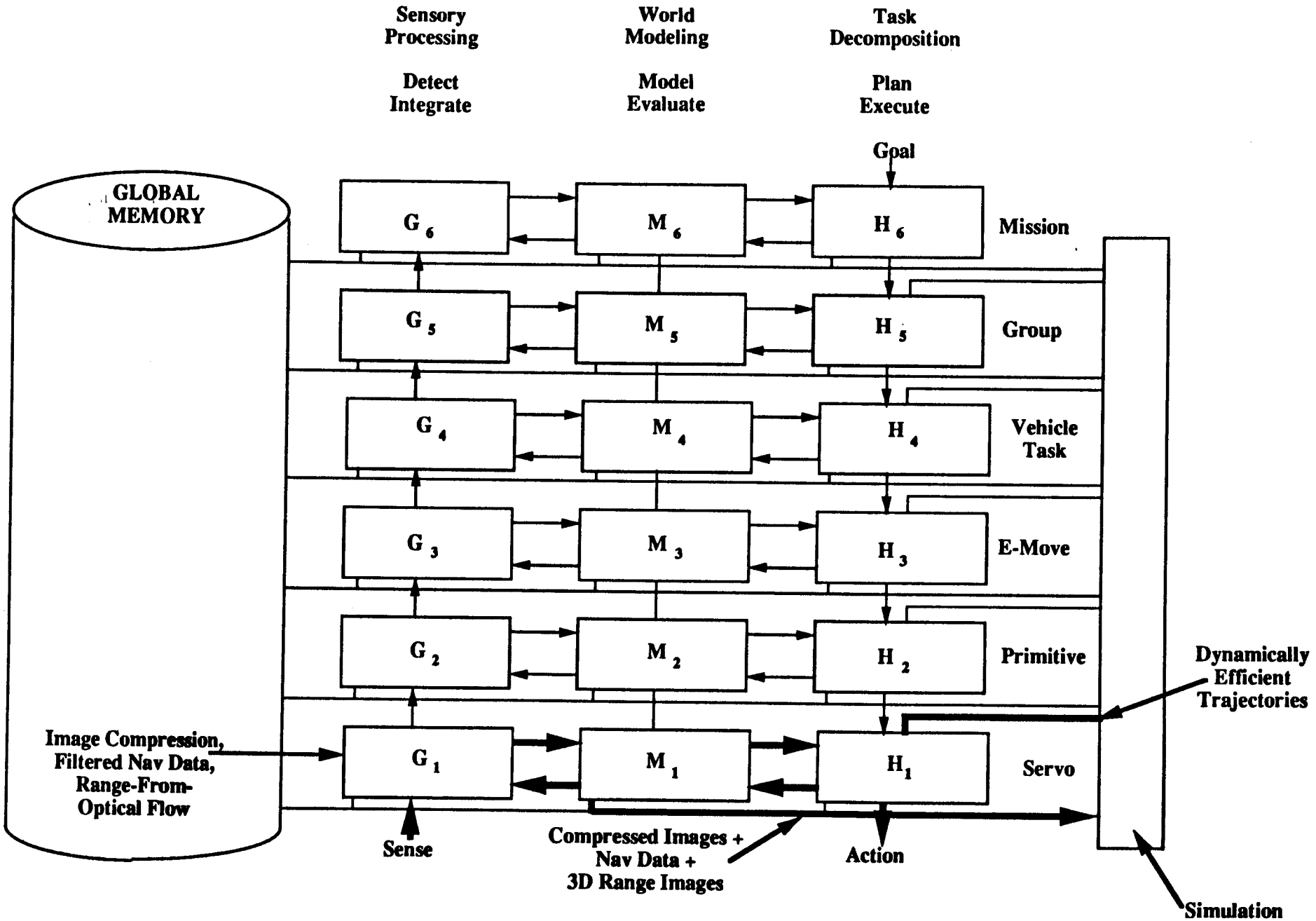


Figure 7. Teleoperated driving using a dynamically acquired 3-D scene model. Thick lines indicate relevant communication pathways.

At the operator station, the master image, the range image, and the navigation data are combined (as in Figure 5) to perform realistic simulation for the operator.

4. Real-Time Passive Ranging Using Optical Flow

Optical flow is the flow of intensity information across the image plane due to relative motion between the camera and objects in the environment. It may be represented in the form of an image, where each pixel has associated with it an instantaneous velocity vector representing image motion at that point. In practice, optical flow is extracted by processing a time sequence of at least two images. The goal of our work in this area has been to extract highly accurate optical flow in real time. This can be done if the flow direction at every point is known ahead of time.

4.1. Predicting the Flow Field

Our method of passive ranging assumes that (a) the camera is moving in a stationary world and (b) the camera motion is known. These assumptions lead to two conclusions. First, the optical flow field in the image (i.e., the flow direction at every point) can be predicted. Second, once optical flow has been extracted, the flow vectors can easily be converted to range values. To see why, we will consider three types of camera motion -- pure translation, pure rotation, and a combination of translation and rotation.

Consider a camera undergoing pure translation. Figure 8 is an illustration from Gibson [6] showing the optical flow induced by this motion in a stationary environment. The arrows represent angular velocities of flow vectors formed by spherical projection of the environment onto a sphere centered at the camera focal point. Note that the flow vectors are zero directly ahead of and behind the moving object. The point directly ahead is called the *focus of expansion* (FOE), and all points appear to flow outward from this point. All flow vectors lie along great circles on the sphere as shown in the figure. If a camera with a planar imaging element were located at the center of the sphere, then the great circles, when centrally projected into the camera, would appear as straight lines in the image plane. Furthermore, if the FOE were also projected into the camera, then all motion would appear to flow radially outward in straight lines from this point. Therefore, if the camera motion is known, the flow direction at each image point can be predicted.

To see how the flow vectors can be converted into range values, consider the spherical coordinate system shown in Figure 9. A point P in the scene is projected onto the sphere by intersecting the ray from the camera focal point to P with the surface of the sphere. In Figure 9, suppose that the positive y-axis is defined by the camera velocity vector. Then the set of great circles that intersect the y-axis represent optical flow lines. Consider a point of interest P' on the sphere. Let A be the angle from the positive y-axis to the

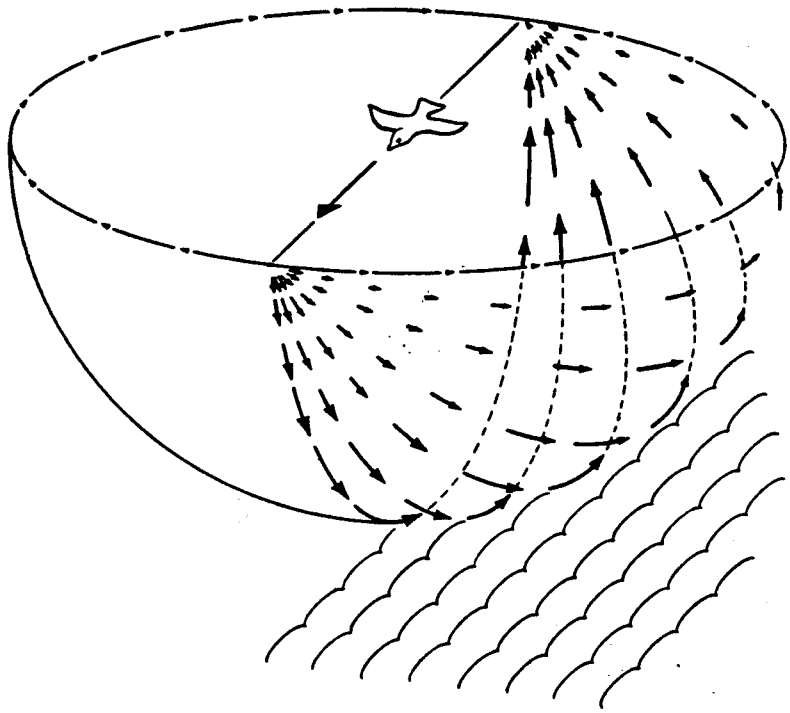


Figure 8. Optical flow induced by camera motion. (Reprinted from Gibson [6], fig. 9.3)

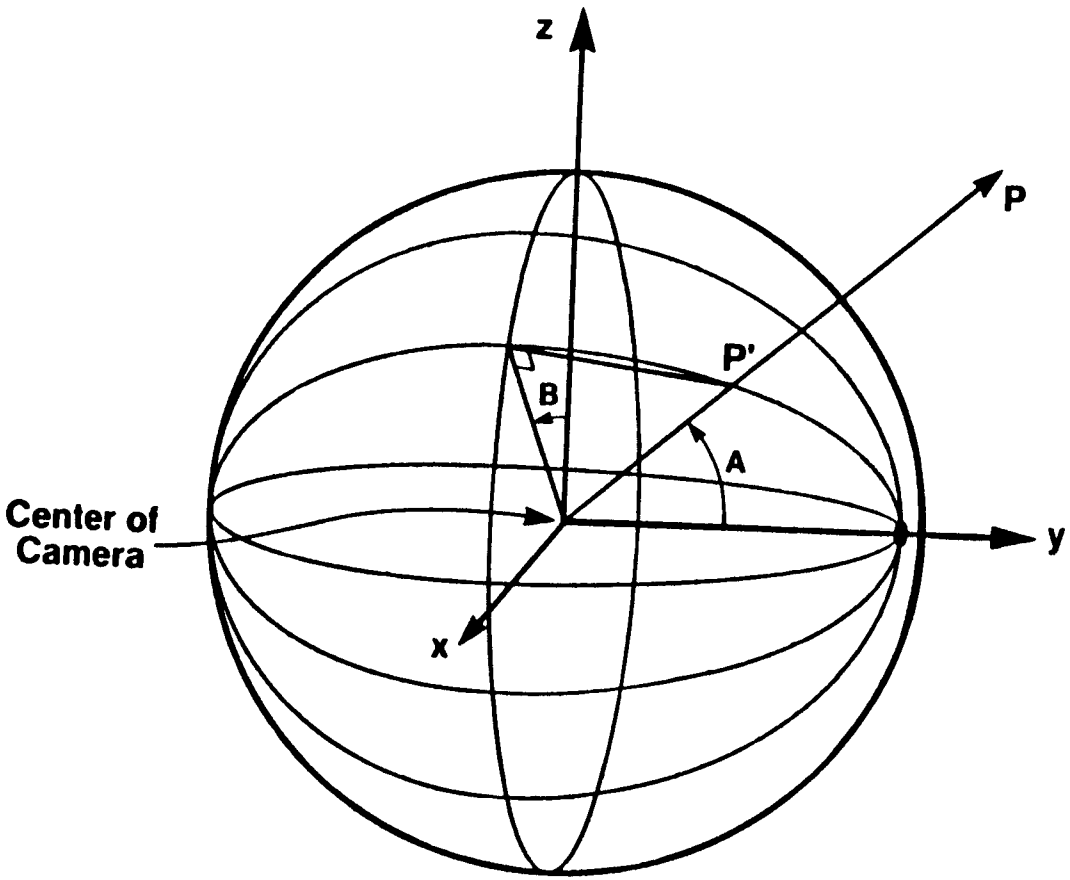


Figure 9. Spherical coordinate system for optical flow.

ray from the camera center to P' and let B be the angle from the positive z-axis to the plane of the great circle containing P'. If v is the camera velocity and dA/dt is the value of the optical flow on the sphere at angle A, then the following formula is used to calculate the range r to the point in the scene corresponding to the point at A [3]:

$$r = \frac{v \sin A}{dA/dt}. \tag{1}$$

For a camera undergoing pure rotation, all optical flow on the sphere will be along small circles perpendicular to the axis of rotation. For known camera motion, not only the flow direction but also the flow magnitude at each image point can be predicted. However, camera rotation provides no information for calculating range.

For a camera undergoing both rotation and translation, the optical flow at each image point is given by the vector sum of the rotational and translational components of flow. Figure 10 shows the situation where the translation vector is perpendicular to the rotational axis. At point P' on the sphere, the direction and magnitude of optical flow due to the known rotation can be predicted along the small circle as shown. The direction of flow due to the known translation can be predicted along the great circle as shown. Since the magnitude of the flow vector due to translation cannot be predicted from the camera motion, the resultant flow vector at point P' lies within a family of vectors as shown in the figure. One technique that can be used to extract and analyze the optical flow is to subtract out the flow due to rotation. This can be accomplished by warping each image so as to eliminate all grey scale displacements due to rotation. This is easily accomplished for known camera rotation. The result is a temporal sequence of images with no rotational flow components. For these images, the flow direction can be predicted, extracted, and converted to range values as described above for pure translational motion.

4.2. Accuracy and Real-Time Performance

The methods described above show how the flow field can be predicted. In this section, we describe how knowledge of this flow field can lead to extraction of highly accurate optical flow and to real-time performance.

First, when extracting the flow vectors, knowledge of the vector directions eliminates the aperture problem [12]. The aperture problem states that if image motion is detected using a spatially local operator, then only the component of motion parallel to the local brightness gradient can be computed. This component of motion is called normal flow. Typically, this problem is handled with a costly constraint satisfaction algorithm that combines the normal flow components over an extended region of the image. Because our approach assumes that the flow vector directions are already known, only the magnitudes of the flow vectors need to be computed. This leads to real-time performance. Another

OPTIC FLOW DUE TO COMBINED ROTATION AND TRANSLATION

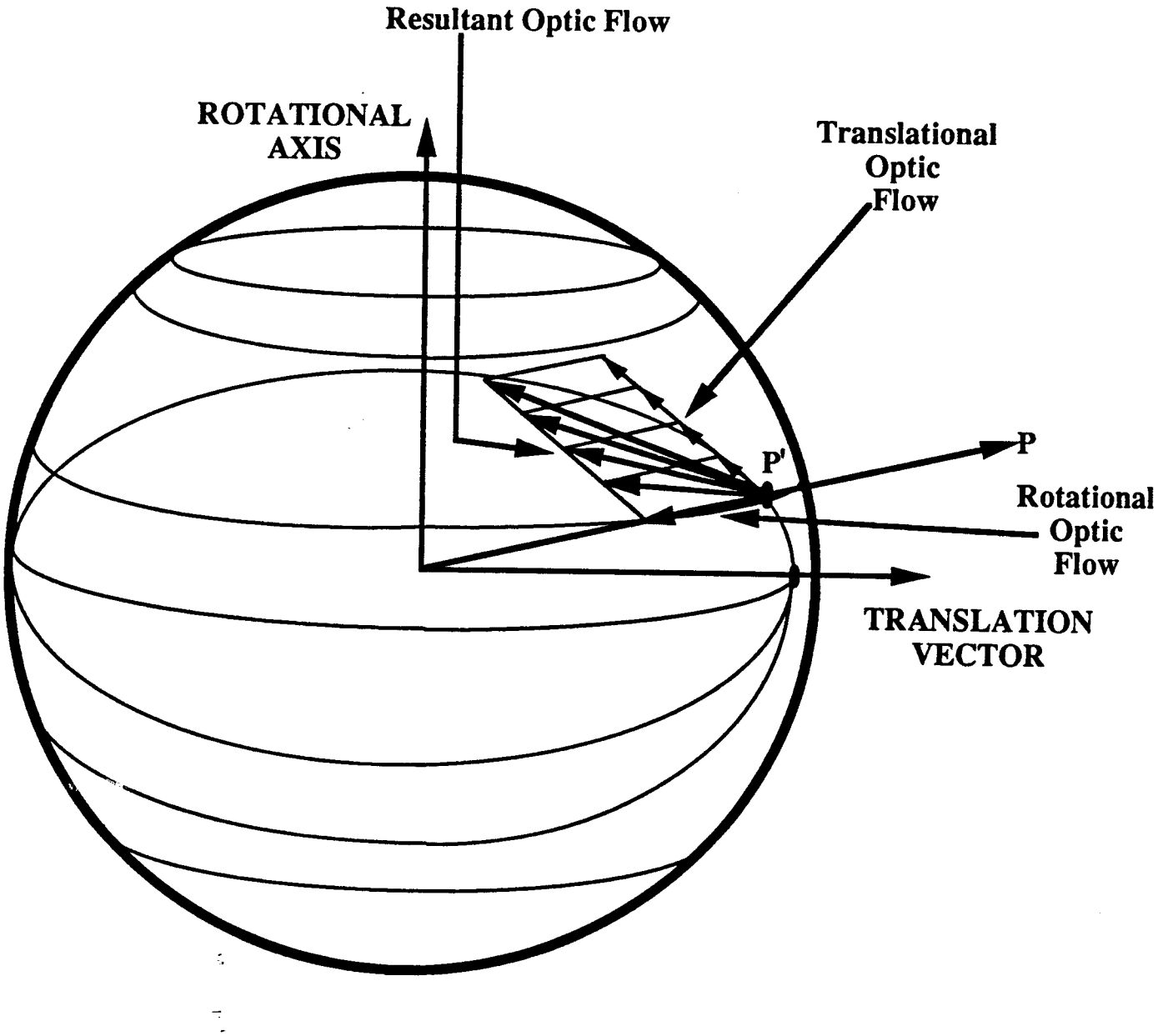


Figure 10.

factor leading to real-time performance is that by eliminating the aperture problem, local image operators can be used to extract the flow, and these local operators can run in parallel. Our approach should also lead to greater accuracy of extracted flow vectors since we already know the vector direction accurately.

Knowledge of the motion field also allows the use of temporal consistency to increase accuracy. Temporal consistency means that flow vectors should remain consistent over time. The concept behind this approach is shown in Figure 11. Suppose the camera is in two successive positions due to its motion over a short distance. Then for almost every point in the first image, there will be a point in the second image that corresponds to the same 3-D point in the scene. Corresponding points are assumed to have consistent flow values because they arise from the same point in the scene. The assumption of temporal consistency is that flow values for corresponding points change only as a function of geometry, i.e., as a function of the relative positions of the moving camera and as a function of the camera's speed. We utilize temporal consistency as follows. First calculate the flow values at each pixel for multiple image frames spanning a short period of time. Then determine corresponding points in the multiple images. Correspondences can be determined from the computed flow values since these values are proportional to pixel disparities. We then normalize these flow values to eliminate differences in geometry. Next, the resulting flow values are integrated over the multiple images. This should result in improved estimates of the flow values [13]. We have implemented this algorithm on PIPE by doing the integration using running averaging over about 10 image frames. A slightly different version of this algorithm which first converts flow values to range values and then integrates these values in a world coordinate system has been implemented on the Sun-3 computer (see [3] for details).

4.3. Flow Extraction Techniques

This section describes three optical flow extraction techniques that we have implemented for our experiments. These are (1) spatial and temporal derivatives, (2) spatio-temporal imagery and (3) temporal cross correlation. The technique of spatial and temporal derivatives is based on a method developed by Horn and Schunk [10] which involves calculating both spatial and temporal derivatives at each point in the image. Because a spatially local image operator is involved, this method normally computes only the flow component parallel to the spatial gradient. However we make use of the known flow direction to compute the true flow as the ratio of the temporal derivative to the directional spatial derivative along the flow line. This technique has been implemented on PIPE and is described in more detail in [16].

Temporal Integration of Flow Values

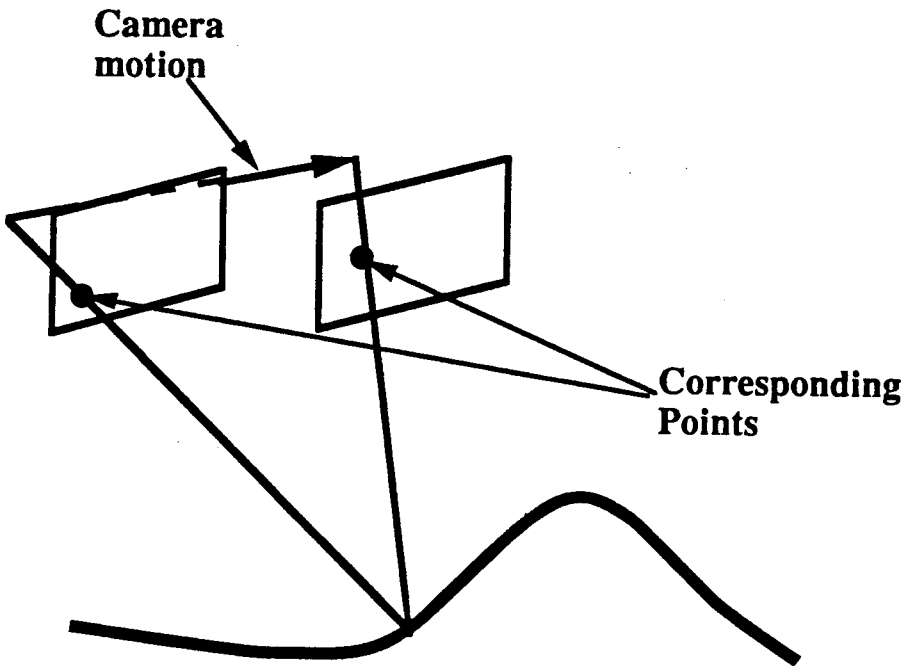


Figure 11. Corresponding points are assumed to have consistent optical flow values.

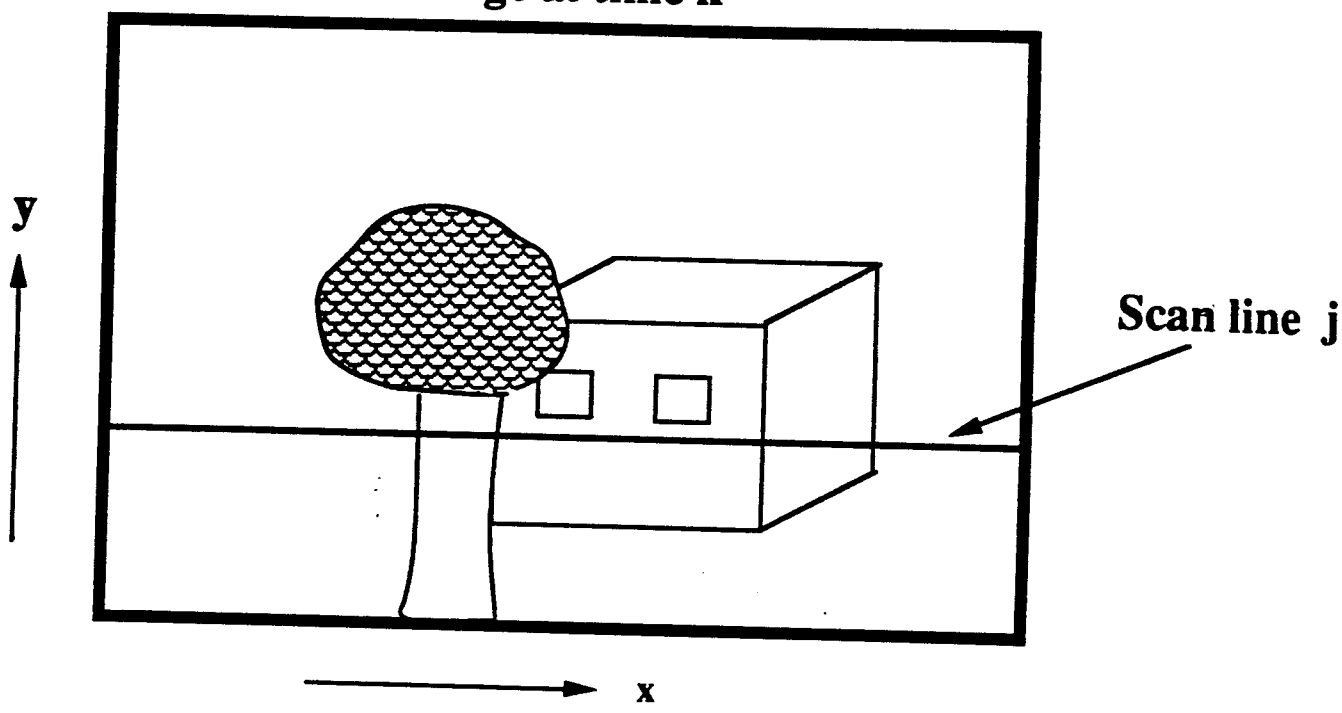
Another technique we have implemented is the method of spatio-temporal image analysis (or epipolar-plane image analysis [5]). Suppose that the camera is moving in the horizontal direction, along its scan lines, and consider scan line j in the image (Figure 12a). We can associate a spatio-temporal image with this scan line (Figure 12b) in which the x-axis represents the x-direction along the scan line and the y-axis represents time. Each row of the spatio-temporal image will show the given scan line at a different point in time. At each instant of time, the current rows in the image are shifted up by one row and the new version of scan line j is inserted at the bottom of the image. Each feature on the original scan line will generate a line, or *streak,* in the spatio-temporal image. The slope of the streak is proportional to the optical flow of the feature; the smaller the slope, the greater the flow velocity. This algorithm has been implemented to run in real time on PIPE. We have also implemented edge finding techniques to extract the slope of the streaks. The resulting slope images are thresholded to perform range segmentation. Further details are provided in [15]. Notice that this technique requires advance knowledge of the flow direction. In principle, if the flow direction is radially outward from the FOE rather than along a scan line, motion along the radial line as a function of time can be plotted in the spatio-temporal image.

Another technique that we have implemented is temporal cross correlation. This is shown in Figure 13. For some scan line j in the image, we obtain the temporal brightness profile (i.e., intensity vs. time) for each of two successive pixels i and i+1. These pixels may be adjacent to one another or may be separated by some distance. By correlating these profiles, we can determine the amount of time τ for brightness values to travel from one pixel to the next. Hardware for performing this will be described below.

4.4. Using Optical Flow for Unmanned Vehicles

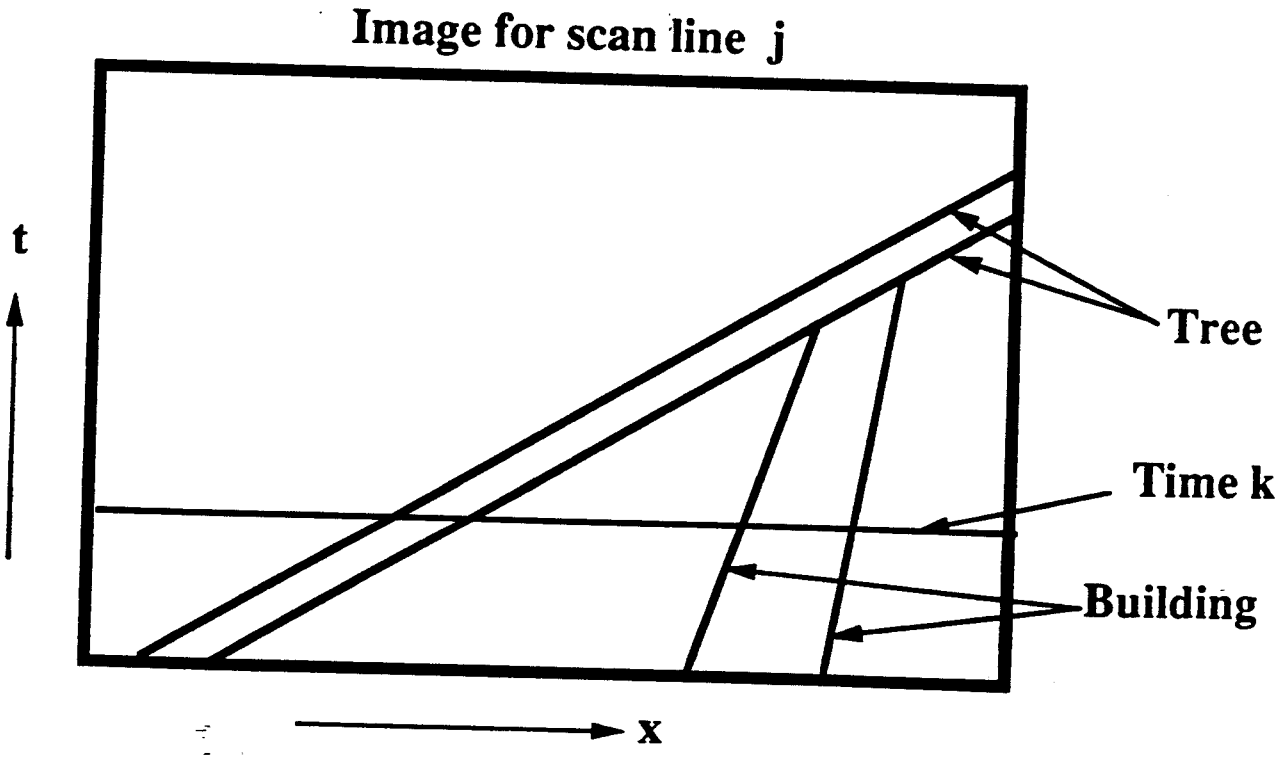
In order to use optical flow for unmanned vehicles, we classify the flow into two categories: (1) the global flow field induced by vehicle and camera motion, and (2) the local flow fields induced by motion of objects in the environment. The flow due to the first category is useful for vehicle driving, for 3-D terrain reconstruction, for acquiring stationary targets, and for 3-D landmark recognition. The flow due to the second category is useful for acquiring, identifying, and tracking moving targets or objects. To make use of the global flow field, we need an inertial navigation system that gives us vehicle and camera motion. Once we know the camera motion, we can subtract out the rotation of the camera (as described above) and calculate the FOE accurately. This allows accurate and real-time extraction of the global flow field. To make all of this work in real time, the camera should be stabilized so that optical flow due to camera jitter is eliminated.

SPATIO-TEMPORAL IMAGERY



Motion along scan lines.

(a)

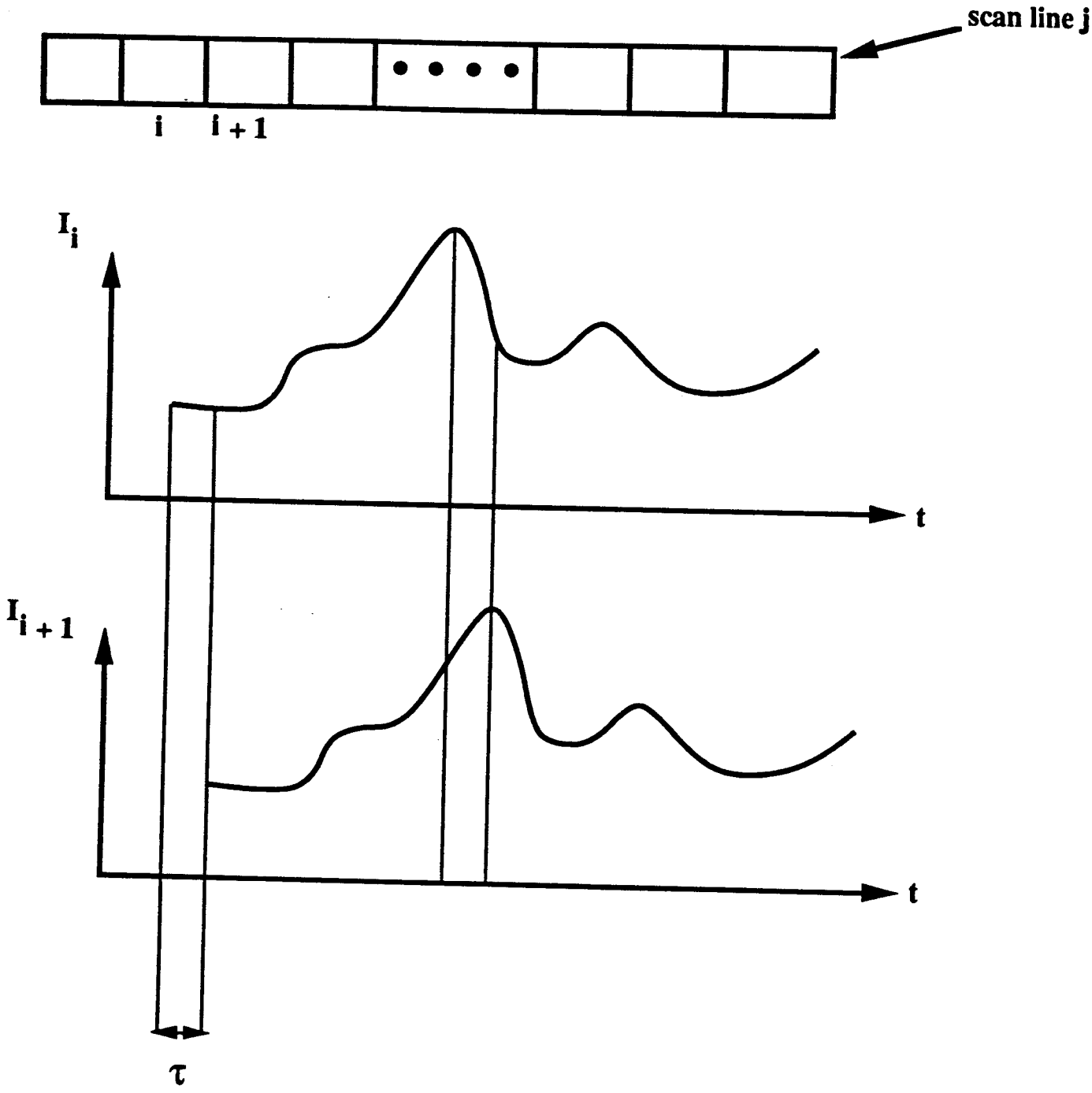


Slope of streak proportional to flow velocity

(b)

Figure 12.

TEMPORAL CROSS CORRELATION



$$\Phi(\tau) = \int^{\tau} I(t) \bullet I(t+\tau)\, dt$$

Figure 13.

Our method of passive ranging using optical flow can be used for autonomous driving. Such a system fits into the NIST unified control system architecture as follows (Figure 14). The sensory processing hierarchy performs the following. At the servo level, navigation data are filtered, optical flow is extracted, and range is computed from the optical flow. At the primitive level, the range map is segmented to obtain small range patches. At the e-move level, surface maps corresponding to objects are formed from the range patches. At the vehicle task level, objects are recognized and localized and terrain maps are formed. The information at each level extracted by the sensory processing modules is passed to the world modeling component, where it is used by the task decomposition hierarchy to formulate plans of action for the vehicle. The operator inputs vehicle tasks at the task level. The control system autonomously decomposes these commands into actions for the vehicle.

4.5. Massively Parallel Hardware

For high speed unmanned vehicles, such as low-flying aircraft traveling at several hundred miles per hour, it is important that range updates occur faster than 30 times per second (the normal video frame rate). For example a vehicle traveling at 100 miles per hour will move 4.9 feet during one frame time (1/30 second). A vehicle traveling at 500 miles per hour will move 24.5 feet during one frame time. For applications such as 3-D mapping, we need range updates to occur perhaps 1000 times per second. This can be accomplished with massively parallel hardware.

We have designed two parallel hardware systems, one that computes optical flow using temporal cross correlation and one that uses the gradient-based method. These methods work best on linear camera arrays. Consider again the spherical system in Figure 9. The set of great circles that intersect the y-axis represent optical flow lines. Linear arrays can be positioned along these radial flow lines as shown in Figure 15. By positioning them so that they do not contain the FOE, the direction of flow is the same at each point in the array. The image acquisition rates of linear arrays can be made very fast, as much as 1000 linear images per second.

The first parallel hardware design we describe is for computing temporal cross correlation. The algorithm for this method is shown in Figure 16. The figure depicts a bright dot moving against a dark background along a straight line. The dot moves at velocity v along line L from left to right, and as it moves it encounters the pixels $I_1, I_2, \cdots$. Let point C be the focal point of the camera. Then the brightness profile at points $I_1, I_2, \cdots$ as a function of time are rectangle functions as shown. As was described above, the optical flow between two points is calculated by correlating their brightness profiles. The peak in the correlation is obtained with a temporal shift τ, and this shift corresponds to the

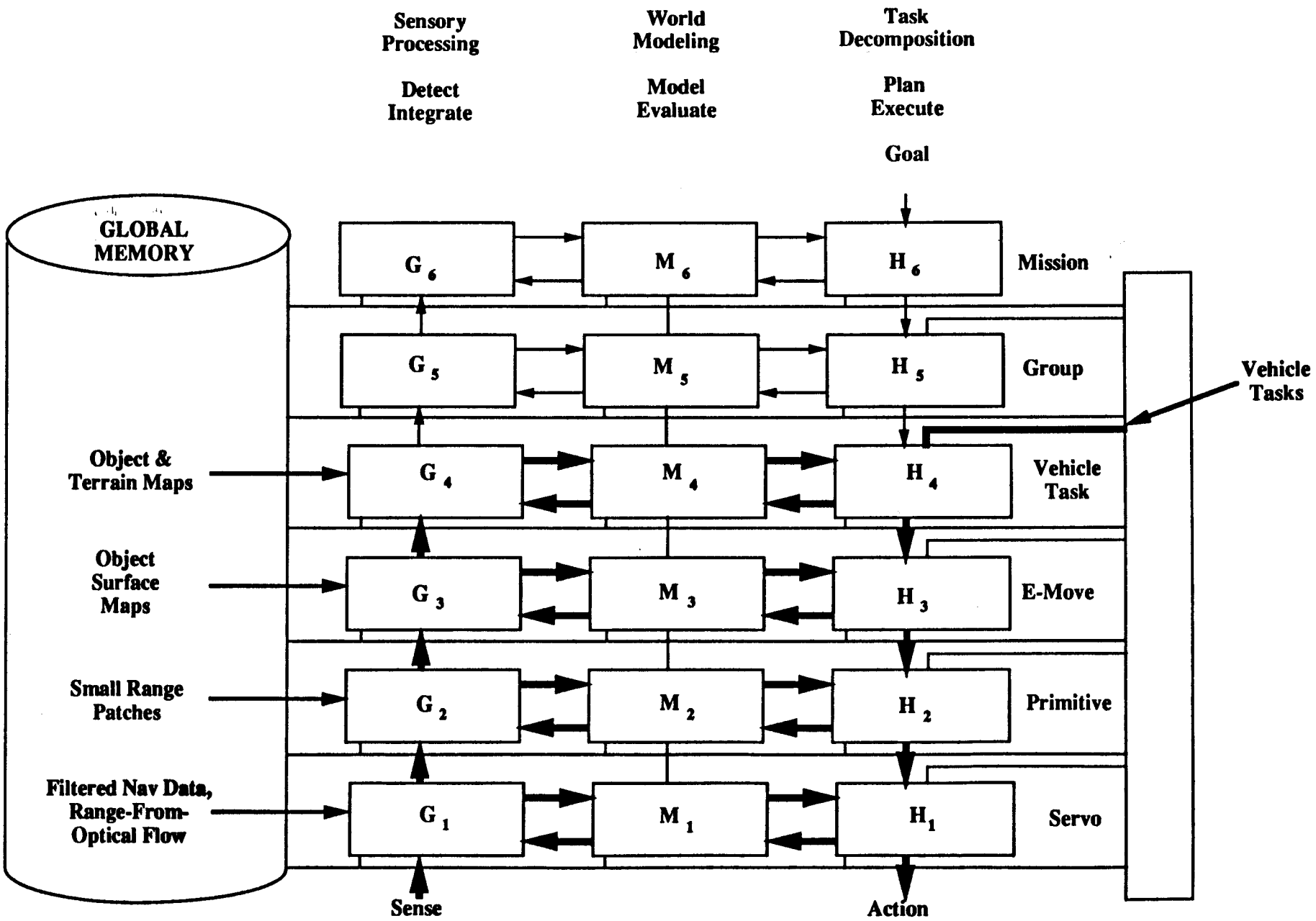


Figure 14. Thick lines indicate relevant communication pathways.

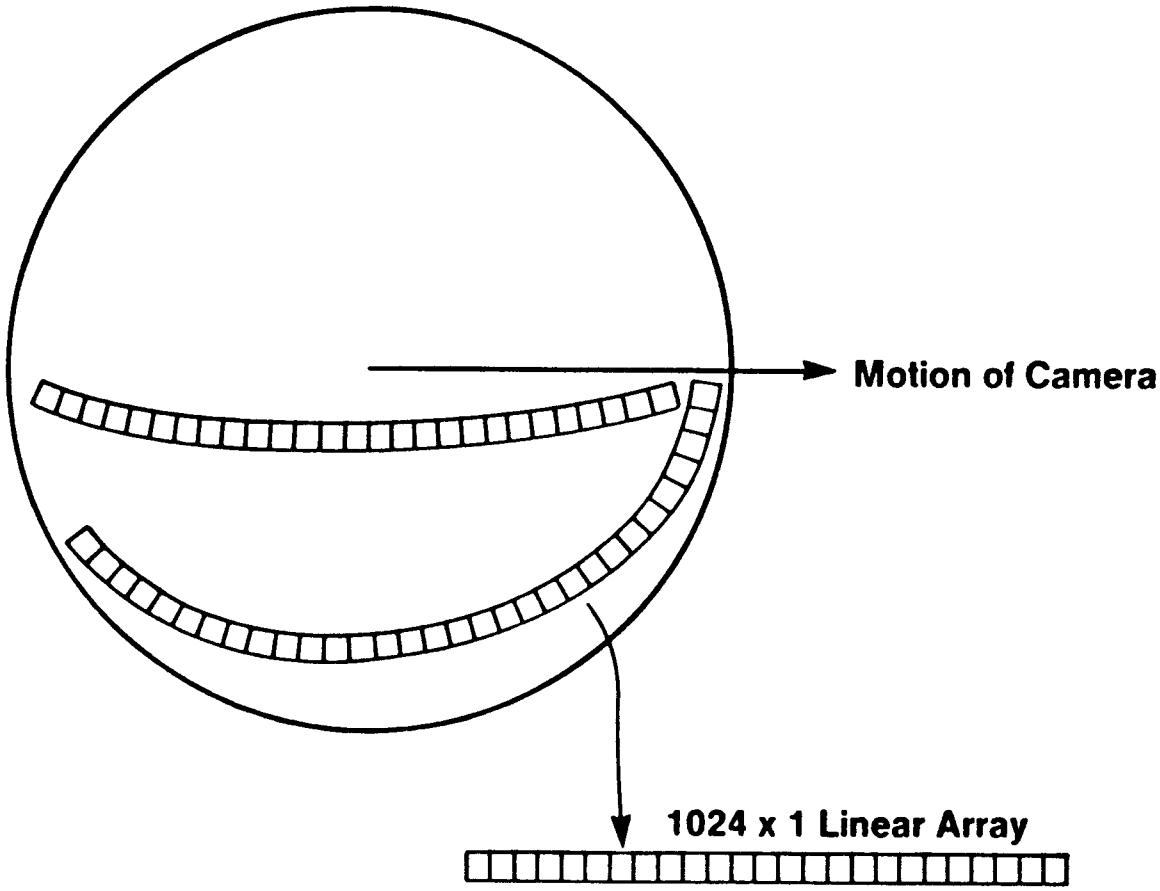


Figure 15. Positioning linear camera arrays along optical flow lines.

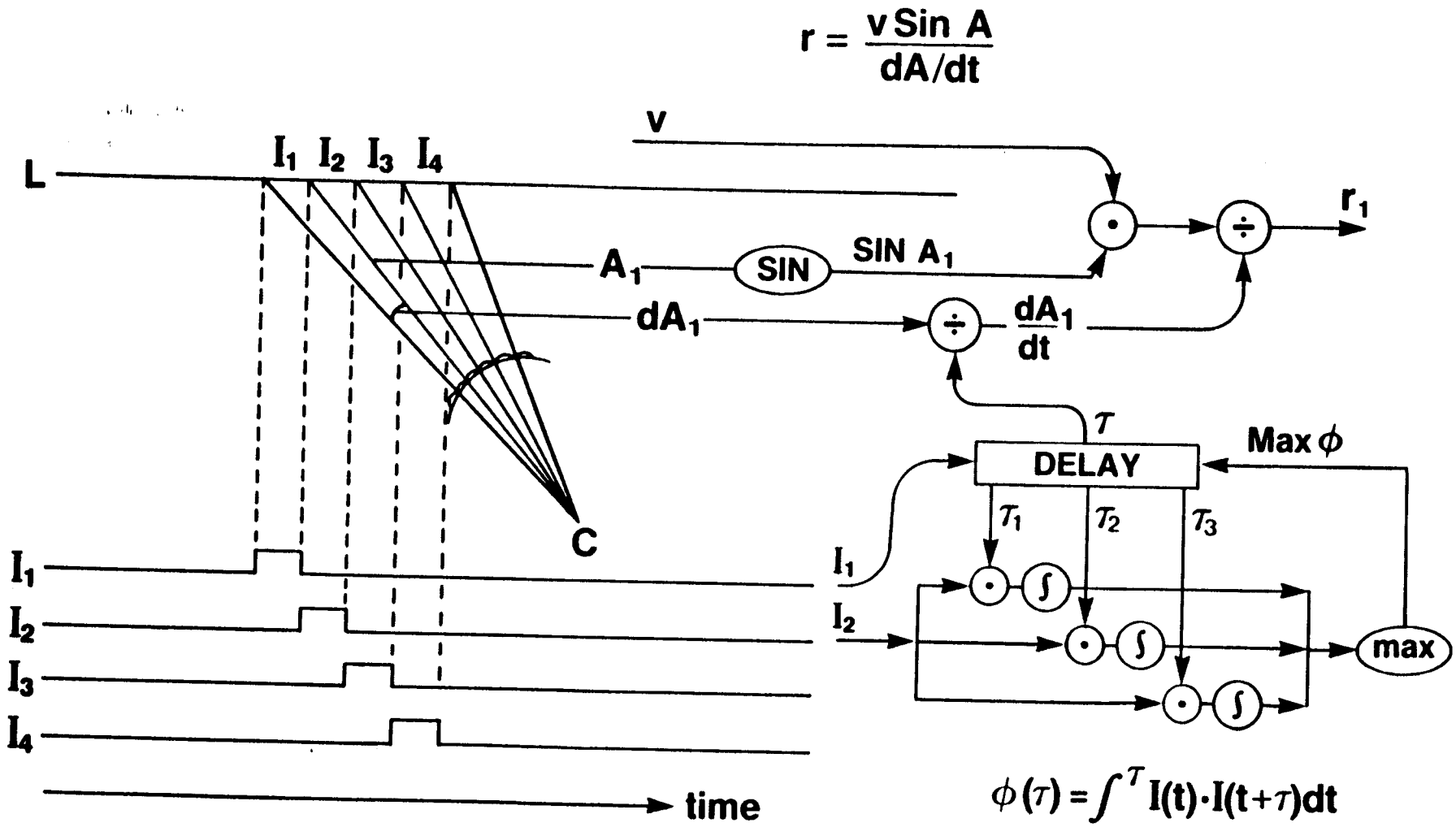


Figure 16. Obtaining range from images using temporal cross correlation to compute optical flow.

amount of time for the dot to move between the two pixels. Note that this algorithm makes use of the assumptions, discussed above, that optical flow is along the linear array represented by $I_1, I_2, \cdots$, and that the flow along this array is in only one direction.

In order to achieve real-time processing, we propose that all relevant pairs of pixels be processed in parallel. For each such pair, the hardware would be identical. Figure 16 shows the processing that occurs for pixels I_1 and I_2. The cross correlation is defined as

$$\phi(\tau) = \int I_2(t) \cdot I_1(t+\tau)\, dt. \tag{2}$$

A delay unit is used to delay the signal I_1 by discrete values $\tau_1, \tau_2, \tau_3, \cdots$. For each value of τ, the cross correlation $\phi(\tau)$ is calculated by separate cross correlation (i.e., multiply and integration) hardware units. The value of τ for which $\phi(\tau)$ is a maximum is proportional to the optical flow between I_1 and I_2. Further details on this design can be found in [3].

Additional hardware, shown in Figure 16, is then used to calculate range using equation (1). Again identical hardware is used for each relevant pair of pixels. Figure 16 shows how this equation is implemented for pixels I_1 and I_2. The angle A_1 and differential angle dA_1 corresponding to these pixels is known a priori. The velocity v of the camera is obtained from other sensors, such as an inertial navigation system on-board the vehicle. The optical flow dA_1/dt is then calculated from the maximum correlation shift τ. This allows the calculation of the range r_1 corresponding to the point at A_1.

A parallel hardware design that uses the gradient-based method is described next. The gradient-based formula for calculating optical flow at a point is [3, 10].

$$\frac{dA}{dt} = \begin{cases} \dfrac{dI/dt}{dI/dA} & \text{if } dI/dA \neq 0, \\ undefined & \text{if } dI/dA = 0. \end{cases} \tag{3}$$

Let us consider Figure 17 to see how this algorithm would be executed in hardware. To calculate the optical flow at, say, point I_1, first the temporal difference at this point, $I_1(t) - I_1(t - \tau_0)$, is calculated. This approximates the value dI_1/dt. As shown in Figure 17, a delay unit is used to delay the signal I_1 by a value τ_0, and the resulting signal is subtracted from the signal I_1 at time t.

The spatial difference, $I_2(t) - I_1(t)$ is also calculated, resulting in an approximation for the value dI_1/dA. The ratio of the temporal and spatial differences results in an approximation of the optical flow at point I_1. Additional hardware similar to that in Figure 16 is then used to calculate the range r_1 corresponding to the point A_1 (see Figure 17). Note that in order to achieve real-time processing, we propose that all pixels be processed in parallel, using the identical hardware depicted in Figure 17.

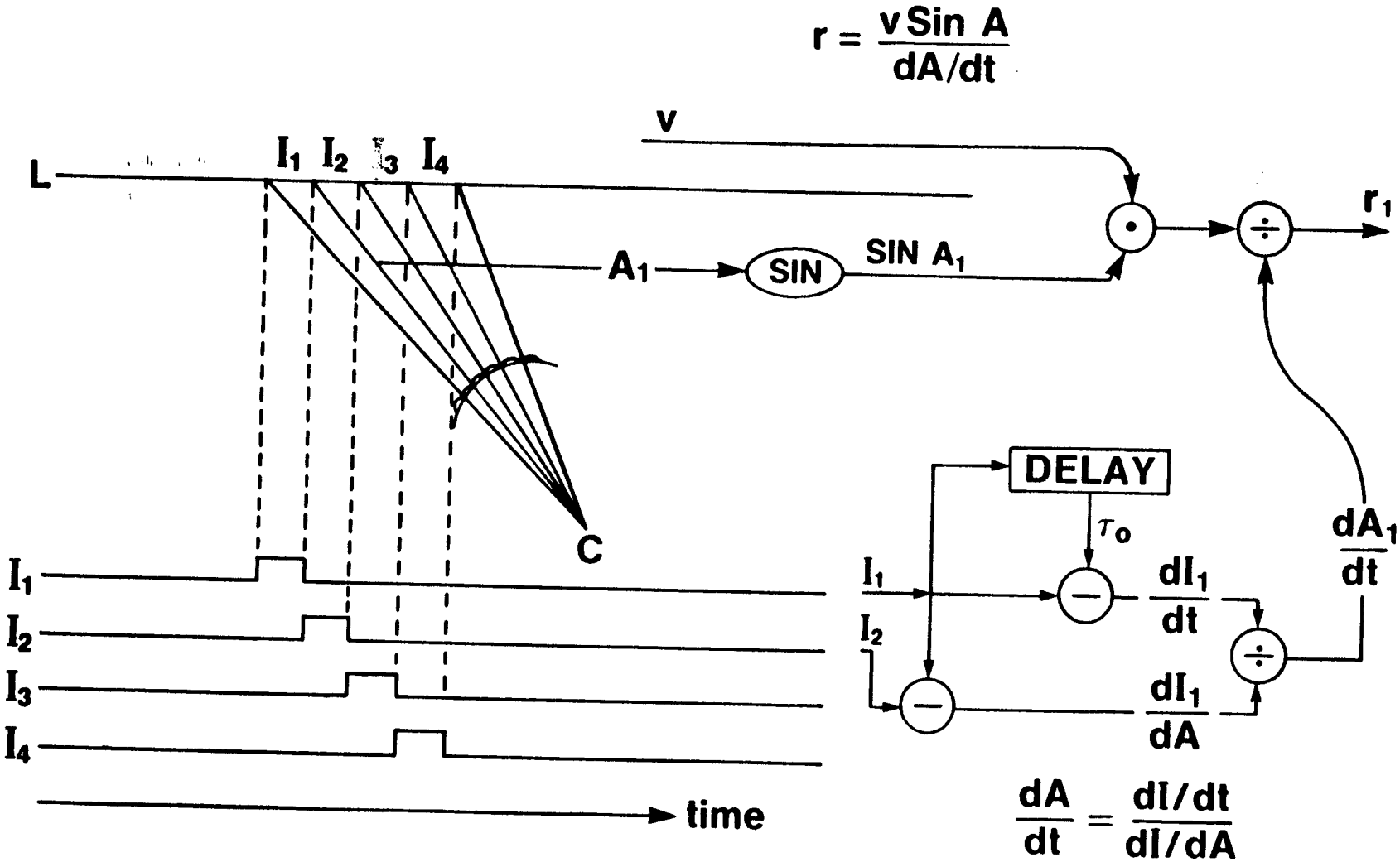


Figure 17. Obtaining range from images using the gradient-based method to compute optical flow.

5. Integration of Teleoperation and Autonomy

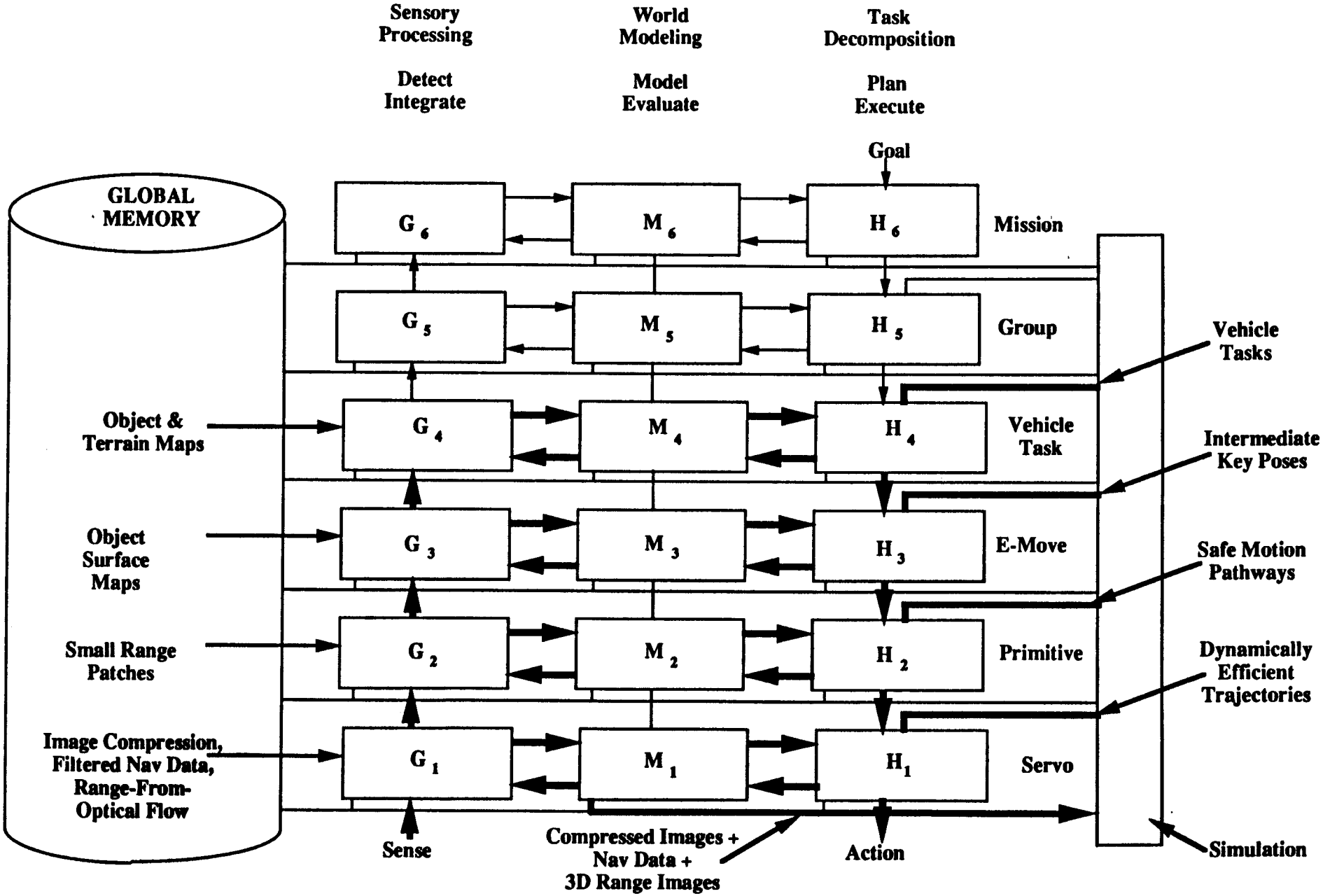
The various techniques for vision-based teleoperated and autonomous driving discussed above can be unified under the NIST hierarchical control system architecture, resulting in a single integrated system (Figure 18). In such a system, compressed imagery from the vehicle camera and range images computed from optical flow are transmitted periodically to the operator station. Navigation data are transmitted very frequently. Through simulation, continuous video is displayed on the operator's console. As he monitors the video, he may enter commands into the control system at any level he desires. Normally, he will enter vehicle task commands and monitor their autonomous execution. If he wants a finer level of control, he will enter intermediate key poses. If the system does not seem to be executing the desired actions, or if a delicate situation is at hand, the operator will want to be able to enter the control system either by indicating safe motion pathways or by specifying dynamically efficient trajectories (i.e., directly controlling steering, brakes, throttle, and transmission). The control system will allow the operator to impose his own commands at any level of the hierarchy and at different levels as the situation requires.

6. Conclusion

Vision is very important for controlling both autonomous and teleoperated unmanned vehicles, particularly in outdoor environments. This paper has presented real-time vision systems for low data rate remote driving and for passive range extraction. It was then shown how these systems can fit into the NIST unified control system architecture. The paper has also shown how a single integrated system can be developed. Such a system would allow the operator to visually monitor execution of commands by the vehicle and to enter commands into the control hierarchy at any level he desires. Such a system would also allow powerful vision techniques such as optical flow to be used in varied control situations.

Acknowledgements

This work has been supported by the U.S. Army Laboratory Command, Human Engineering Laboratory, and by the Defense Advanced Research Projects Agency (DARPA), Tactical Technology Office. Special thanks go to Mr. Charles M. Shoemaker of the Human Engineering Laboratory and to Dr. Jasper Lupo of DARPA for their direction and guidance.



INTEGRATED TELEOPERATED AND AUTONOMOUS DRIVING

Figure 18. Thick lines indicate relevant communication pathways.

References

1. Albus, J.S. "A Control System Architecture for Intelligent Machine Systems." *IEEE Conf. on Systems, Man, and Cybernetics,* Arlington, VA, October 1987.
2. Albus, J.S. "System Description and Design Architecture for Multiple Autonomous Undersea Vehicles." NIST Technical Note 1251, National Institute of Standards and Technology, September 1988.
3. Albus, J.S. and Hong, T.-H. "Motion, Depth, and Image Flow." *Proc. 1990 IEEE International Conference on Robotics and Automation,* Cincinnati, Ohio, May 1990.
4. Albus, J.S., McCain, H.G. and Lumia, R. "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)." NBS Technical Note 1235, National Bureau of Standards, July 1987.
5. Bolles, R.C. and Baker, H.H. "Epipolar-Plane Image Analysis: A Technique for Analyzing Motion Sequences." *Proc. DARPA Image Understanding Workshop,* Miami Beach, Florida, December 1985, 137-148.
6. Gibson, J.J. *The senses Considered as Perceptual Systems.* Houghton Mifflin, Boston, 1966.
7. Herman, M. and Albus, J.S. "Overview of MAUV: Multiple Autonomous Undersea Vehicles." *Unmanned Systems Magazine,* Vol. 7, No. 1, Winter 1988-89, 36-52.
8. Herman, M., Chaconas, K., Nashman, M. and Hong, T.-H. "Low Data Rate Remote Vehicle Driving." *Proc. IEEE International Symposium on Intelligent Control 1988,* Arlington, VA, August 1988, 376-381.
9. Herman, M., Chaconas, K., Nashman, M. and Hong, T.-H. "Video Compression for Remote Vehicle Driving." *Proc. SPIE Advances in Intelligent Robotics Systems: Mobile Robots III,* Vol. 1007, Cambridge, MA, November 1988, 136-143.
10. Horn, B.K.P. and Schunck, B.G. "Determining Optical Flow." *Artificial Intelligence,* 17, 1981, 185-203.
11. Kent, E.W., Shneier, M.O. and Lumia, R. "PIPE (Pipelined Image Processing Engine)." *J. Parallel and Distributed Computing,* 2, 50-78, 1985.
12. Marr, D. and Ullman, S. "Directional Selectivity and Its Use in Early Visual Processing." *Proc. R. Soc. Lond. B,* Vol. 211, 151-180, 1981.
13. Matthies, L., Szeliski, R. and Kanade, T. "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences." Technical Report CMU-CS-87-185, Carnegie Mellon University, Computer Science Department, December 1987.
14. Narendra, P. Personal communication. Honeywell Systems and Research Center, 1988.
15. Rangachar, R., Hong, T.-H., Herman, M., Luck, R. and Lupo, J. "Real-Time Differential Range Estimation Based on Time-Space Imagery Using PIPE." *Proc. SPIE Real-Time Image Processing II: Algorithms, Architectures, and Applications,* Orlando, Florida, April 1990.
16. Rangachar, R., Hong, T.-H., Herman, M. and Lupo, J. "Real-Time Implementation of a Differential Range Finder." *Proc. SPIE Real-Time Image Processing II: Algorithms, Architectures, and Applications,* Orlando, Florida, April 1990.
17. Szabo, S., Scott, H.A. and Kilmer, R.D. "Control System Architecture for the TEAM Program." *Proc. Second International Symposium on Robotics and Manufacturing Research, Education and Applications,* Albuquerque, NM, November 1988.