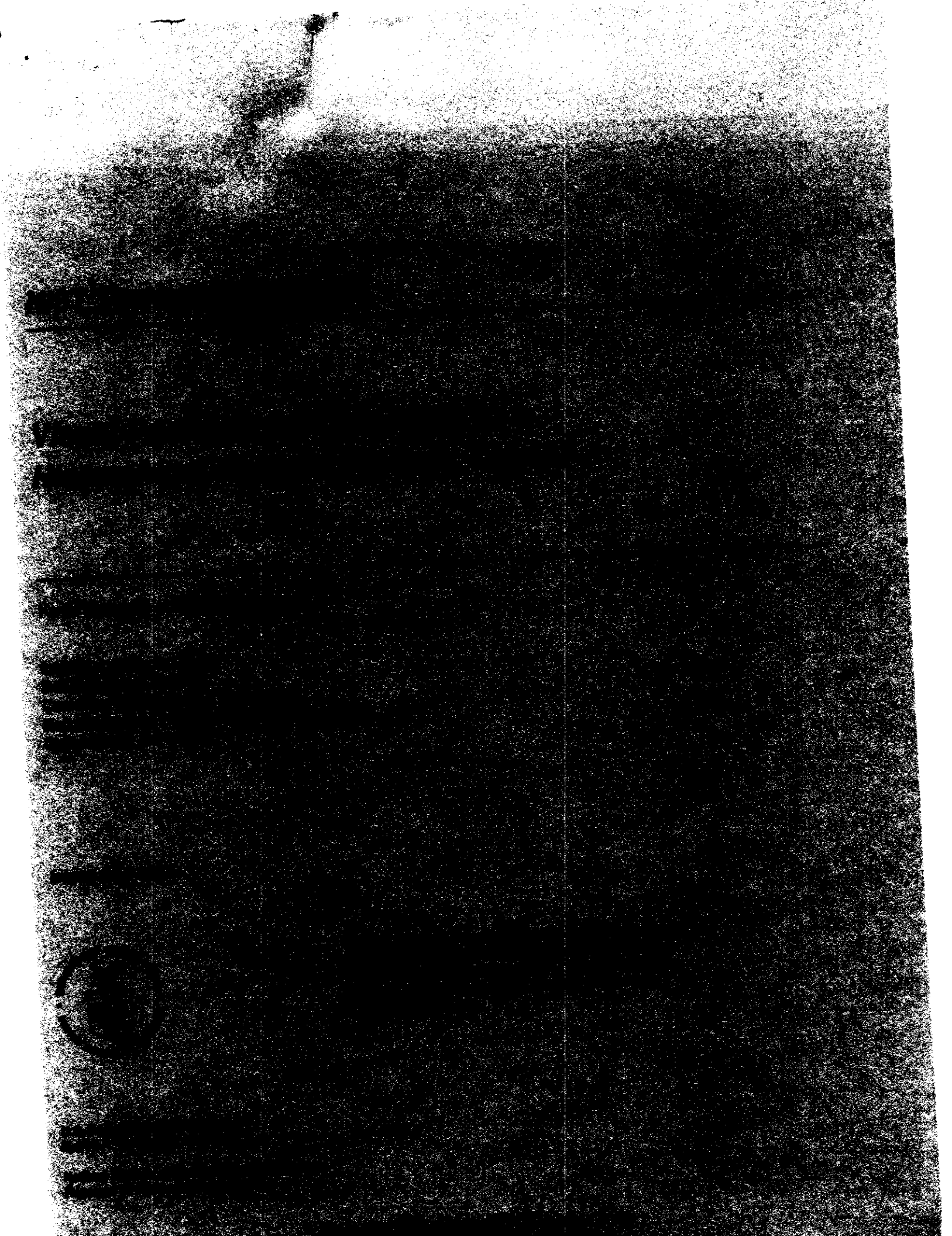


National [illegible]
and Technology
Technical [illegible]
Natl. Inst. [illegible]
Tech. Note [illegible]
49 pages [illegible]
CODEN: [illegible]

[illegible]

[illegible]

Table of Contents

1. Introduction

The telerobot control system architecture discussed in [ALBUS87] describes a hierarchical framework that has been used to control complex robot systems. It decomposes plans both spatially and temporally to meet system objectives. It monitors the environment with system sensors and maintains the status of system variables in order to control system resources.

The control system is composed of three parallel systems that cooperate to perform telerobot control (fig. 1). The task decomposition system breaks down objectives into simpler subtasks to control physical devices. The world model supplies information and analyzes data using support modules. It also maintains an internal model of the state of the environment in the global data system. The sensory processing system monitors and analyzes sensory information from multiple sources in order to recognize objects, detect events and filter and integrate information. The world model uses this information to maintain the system's best estimate of the past, current, and possible future states of the world.

Each device or sensor of the telerobot has a support process in each of the three columns of the control system, as shown in figure 2. For example, the task decomposition functions associated with planning the actions for processing camera data reside in the task decomposition hierarchy; the world modeling functions for supporting those plans reside in the world model hierarchy, and the image processing techniques required for executing those plans reside in the sensory processing hierarchy. The modules can be logically configured according to their function in the system, as shown in figure 3. The system pictured consists of two main *branches*; the left branch contains the perception processes and the right branch contains the manipulation processes. The perception branch of the tree supports processes which provide sensory feedback to the manipulator system such as cameras, range sensors, tactile array sensors, acoustic devices, etc. The manipulator branch of the tree supports processes which are responsible for planning and executing manipulator trajectories. The two branches decompose tasks in most cases independently and communicate via the global data system.

The world modeling support modules communicate asynchronously with the task decomposition and sensory processing systems. Data flows bidirectionally between adjacent levels within any given hierarchy. The interfaces to the sensory processing system allow it to operate in a combination of bottom-up (data driven) and a top-down (model driven) modes. Bottom-up processing involves the extraction of knowledge from sensory data, and top-down processing is used to correlate predicted information from the world model with extracted information from the environment. The interfaces between the sensory processing system and the world model allow updated information to be sent to the world model and predicted information or sensory processing parameters to be sent to the sensory processing system.

This document describes the interfaces and functionality of Level 1 of the perception branch for a camera that is part of a telerobotic control system. This level corresponds to the one highlighted in figure 3. Processing is performed on individual pixels. Level 1 gathers raw information (readings) from each camera, filters the information, and, when applicable, enhances it. It then extracts edge points, surface patches, and information relevant to the op-

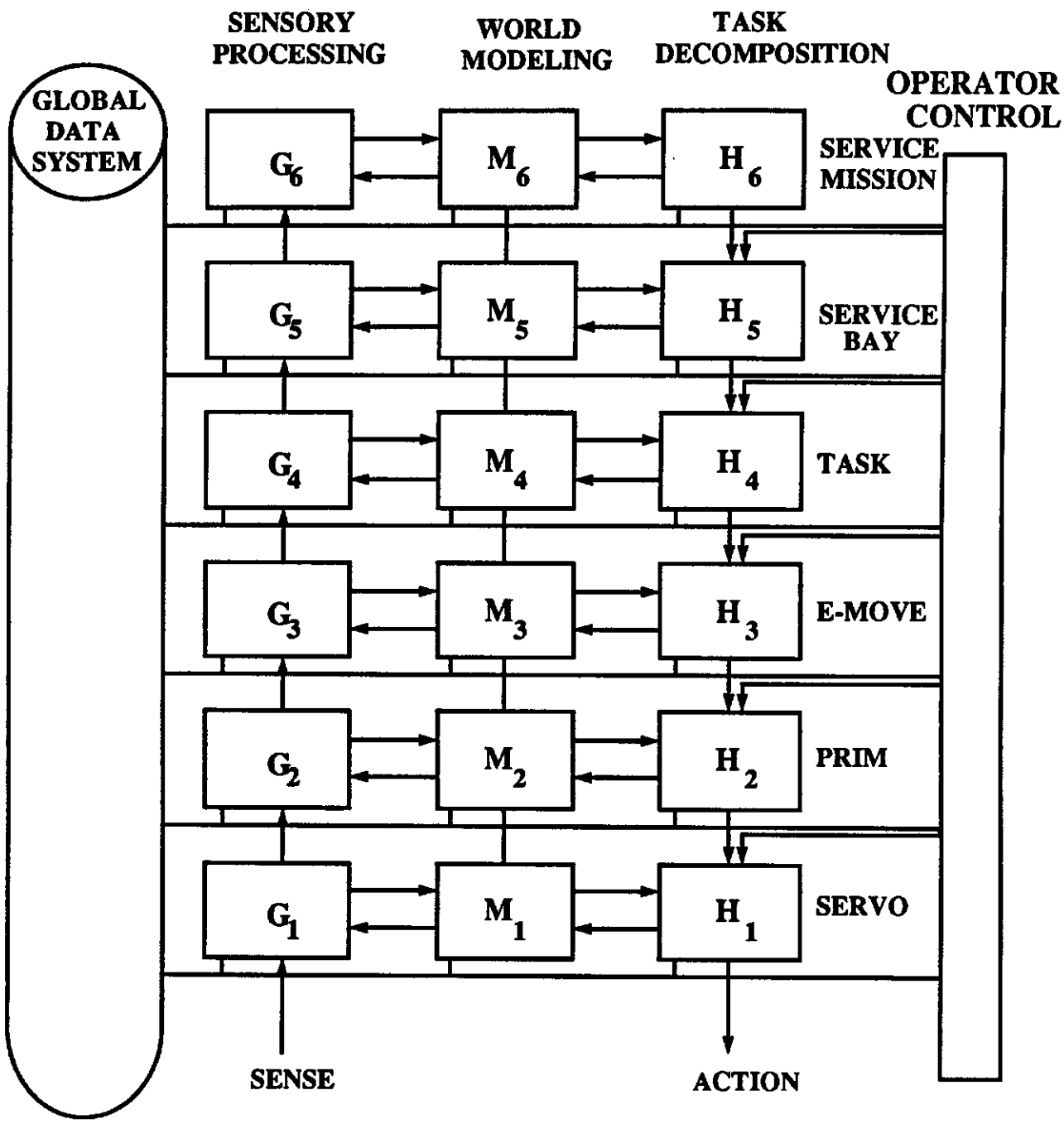


Figure 1. The NASREM Architecture.

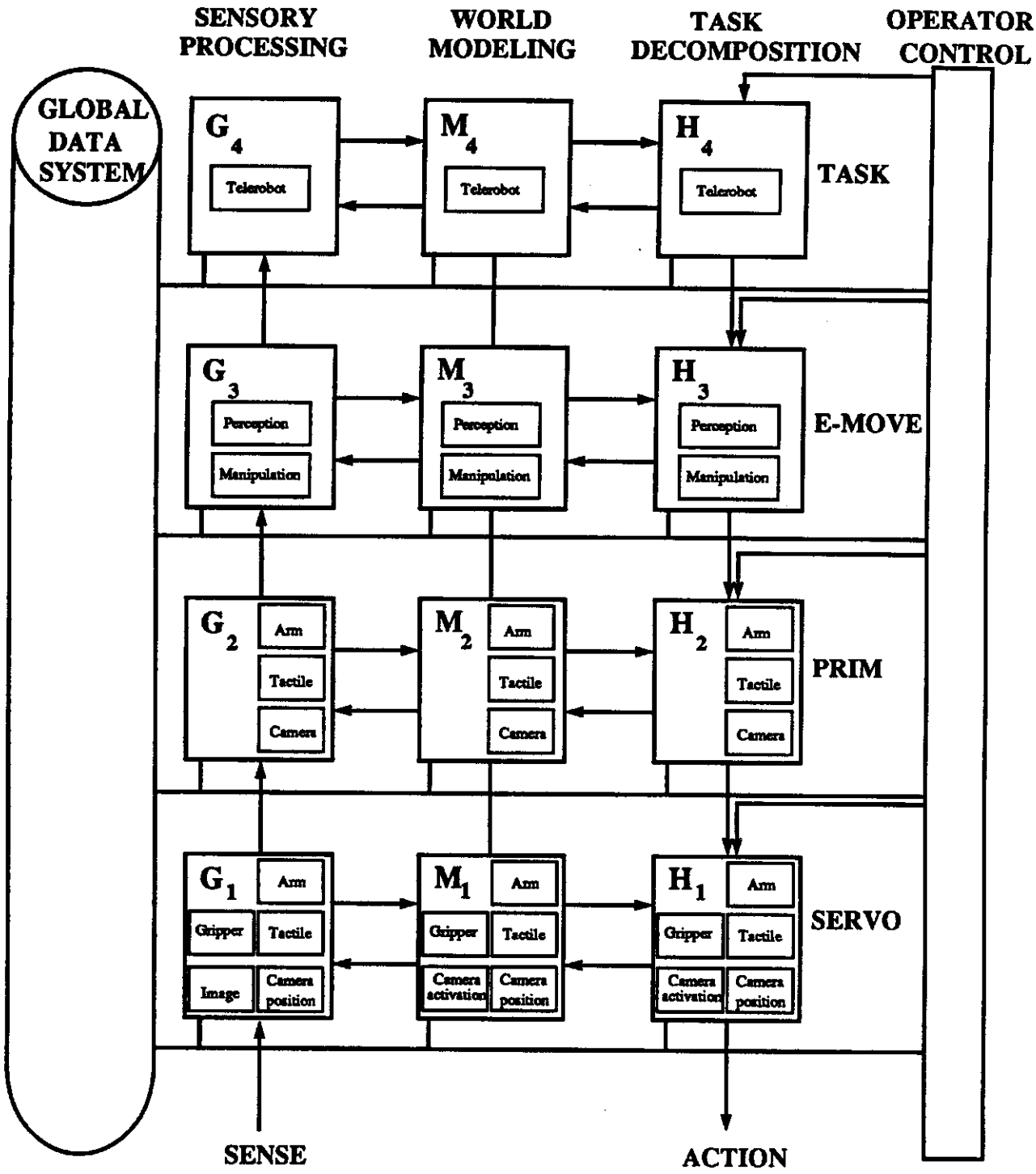


Figure 2. The NASREM Architecture for Control of a Telerobot.

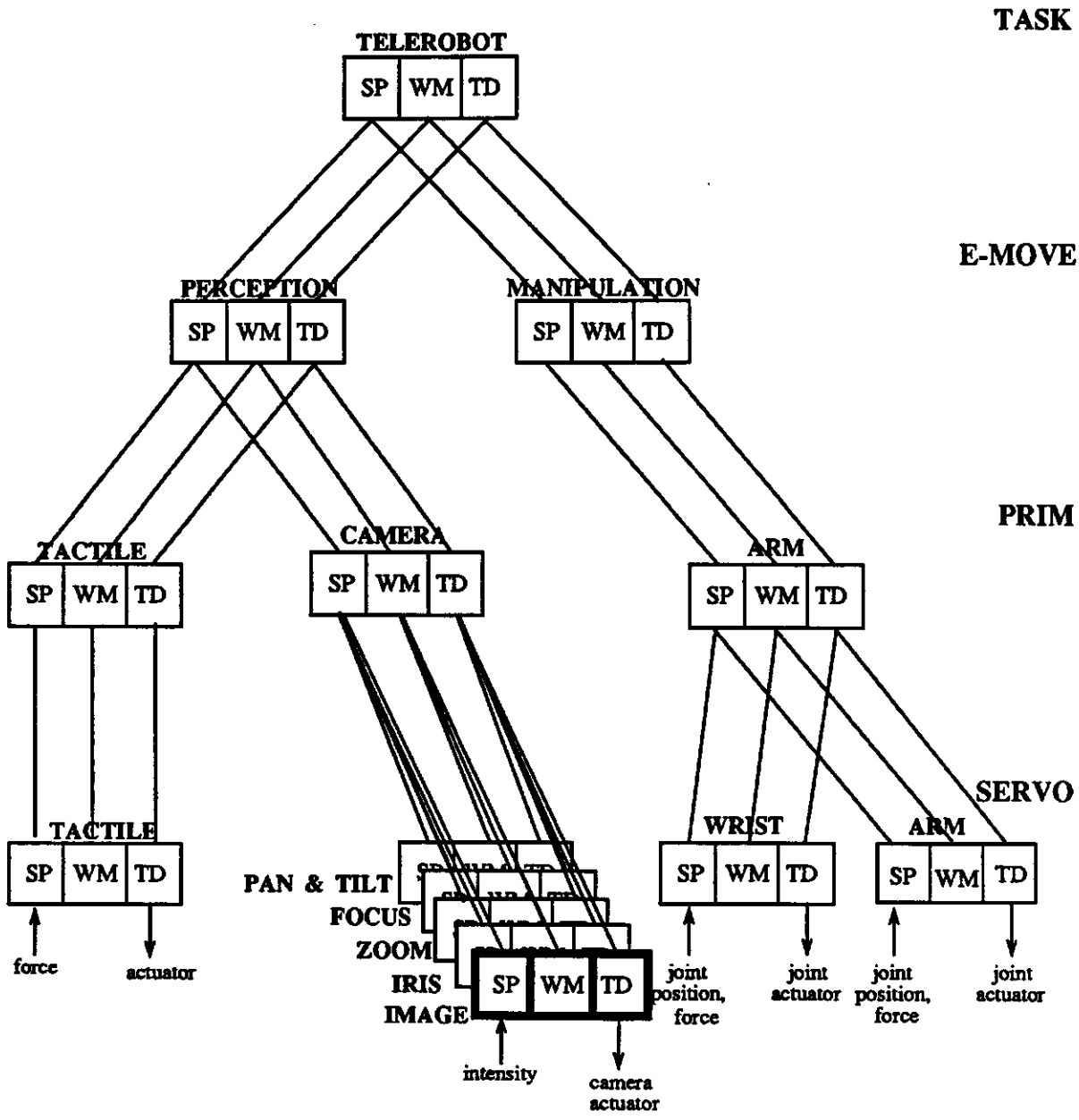


Figure 3. Functionality of the NASREM Architecture for Control of a Telerobot.

tical flow of pixels. Section 2 discusses the general architecture of a computational level of the system and defines the functions and the interfaces of the task decomposition, world model, and sensory processing modules. Section 3 describes the functions and interfaces specific to Level 1 processing for a camera. Section 4 provides an example of the interactions between modules in performing a typical telerobotic task. Appendix A describes preprocessing algorithms that can be applied at Level 1. Appendix B describes edge point extraction algorithms, surface patch or region extraction algorithms, and the first level of optical flow extraction algorithms.

2. General System Architecture

Before describing the functionality of Level 1 of the perception system, a description of the general structure of a computational level is presented. Each level consists of a task decomposition module, a world model support module, and a sensory processing module (fig. 4). The task decomposition module bases its decisions on information extracted by the sensory processing module. The sensory processing module is driven by predictions of the state of the world provided by the world model. The world model maintains the best estimate of the past, current and possible future states of the world [ALBUS81].

2.1. Task Decomposition

The task decomposition module consists of three submodules: Job Assignment (JA), Planner (PL), and Execution (EX). These modules have the same general functions at each level of the system. The Job Assignment module accepts and queues commands from the world modeling support module or the operator. The commands are passed to the Planner module, which analyzes the request and selects the most appropriate sensory processing algorithm for achieving the desired output. The Execution module obtains confidence factors from the world model, updates and modifies algorithm parameters as required, and passes this information through the world model to the sensory processing system. In this way, the evaluation of sensory processing algorithms serves as a learning tool for improving the performance of the algorithm. It is also responsible for activating or deactivating the sensor itself.

Each of the three modules execute cyclically to process commands and pass information. They read inputs, perform computations, and generate outputs independent of the other modules. This type of processing allows the system to operate quickly and efficiently. It prevents system deadlock that can occur when one process waits indefinitely on another for data. It also allows the system to respond to new information without being explicitly commanded for updated calculations.

To coordinate the requested commands among modules, the Planner and Execution modules are directed by one Job Assignment module. The single Job Assignment module interacts with s Planner modules, where s is the number of classes of processing algorithms at a given level of the system. At Level 1, there are five classes of algorithms: filtering, enhancing, edge point extraction, surface patch extraction, and optical flow. Each of the Planner modules communicates with t Execution modules, where t represents the number of algorithms that supply the type of features in the class (fig. 4).

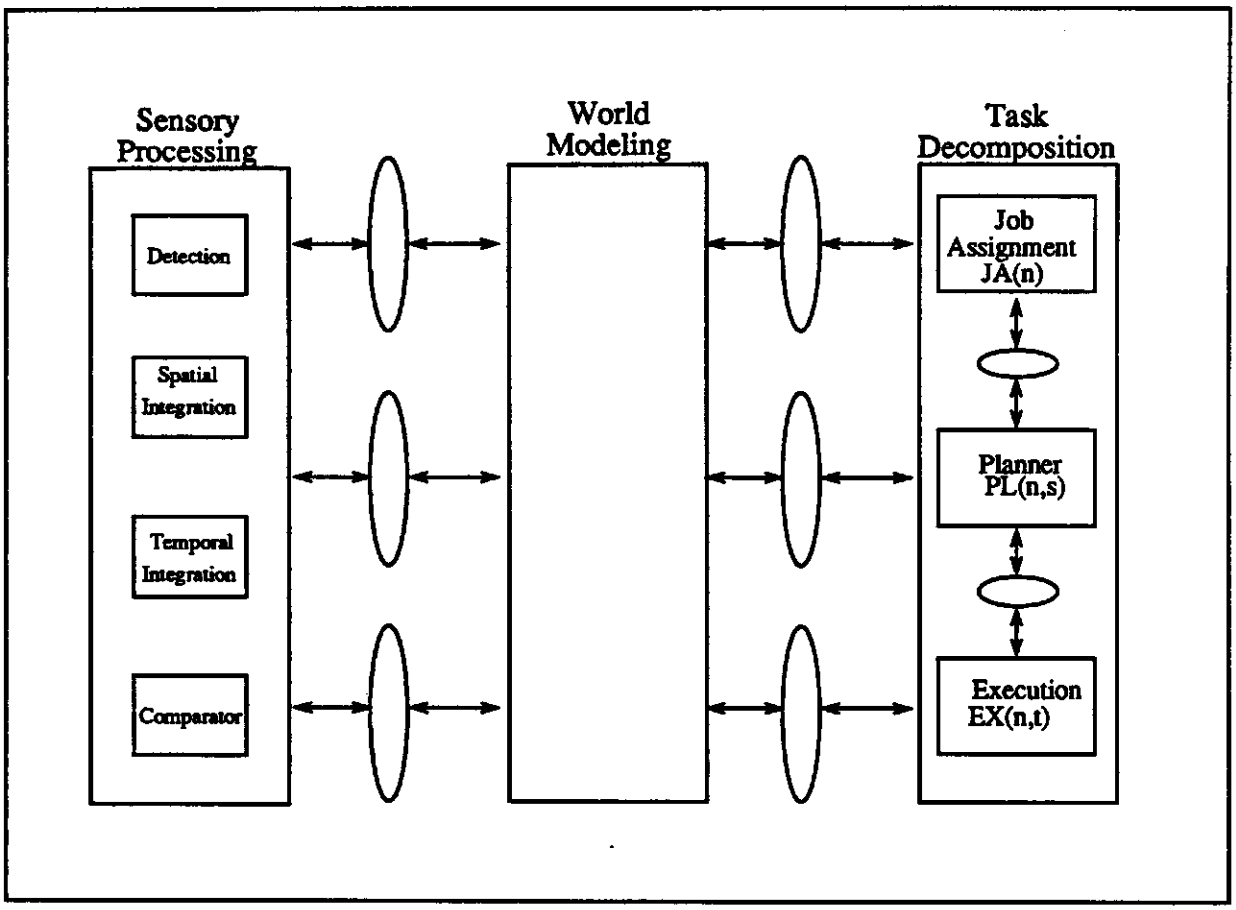


Figure 4. Computational Modules in a Level of the Hierarchical Control System.

2.1.1. Job Assignment Module

Within a computational level, the Job Assignment module maintains a queue of commands received from the world model and the operator. It accepts all incoming commands and assigns them a position in the queue according to the priority level assigned to the command. The priority level is based on the requirements of the plan developed by the Planner at the next higher level. For example, when task decomposition requires information about an object's position, it activates a plan to detect the identifying features of that object and to update their positions. The activation of a plan raises the priority level assigned to the class of algorithms responsible for extracting the required information.

The use of a queue enables incoming commands to be prioritized as they are received. In this way, the information needed most immediately is always serviced first, but all information buffers are updated at specified time intervals. At the completion of execution, the Job Assignment module returns status to the requesting process.

At each level of the hierarchy, the operator interfaces only with the Job Assignment module. He/she may request a specific type of output, output from a particular algorithm, or termination of execution of an active process. He/she may also request a change of parameters for a specific algorithm or request processing in a special window of interest. The Job Assignment module writes parameters supplied by the operator into the world model global memory where they can be read by the Execution module at any level. The operator also specifies the mode of operation for each command he/she issues: either continuous operation until a "halt processing" command is received or execution for a fixed number of times. In all cases, an operator request is assigned the highest priority. Output from an operator's command is returned in the form of graphic displays, ASCII strings, or other easily understandable formats.

2.1.2. Planner Module

The Planner module reads commands from the top of the Job Assignment queue. It distinguishes between commands to control hardware and commands which will initiate a sensory processing algorithm. For the former case, it interprets and passes activation commands to the Execution module. For the latter case, it determines which algorithm within the general class of algorithms capable of being performed in the sensory processing module at the given level is best suited for providing results. Since each class of algorithms contains many methods of computing the required output (Appendices A and B describe the types of algorithms included in the class of filtering, enhancing, and segmentation techniques), the Planner module acts as a rule based system to choose the most appropriate algorithm for a given situation. Decisions are based on criteria such as timing requirements, precision requirements, statistical analysis of sensed information, and knowledge about the environment (lighting conditions, power constraints, etc.). The world model global memory contains this information, and the Planner module reads and analyzes it as required. At completion of the command, it returns status information to the Job Assignment module.

2.1.3. Execution Module

The Execution module receives its commands from the Planner module in the same level of the hierarchy. It is responsible for issuing commands to control a physical device or sensor or

passing algorithm parameters to sensory processing and activating the sensory processing system to execute the selected algorithm. When a particular algorithm is chosen for execution, the Execution module reads the parameters required for its execution from the world model global memory. The types of parameters stored in the world model include threshold values, histories of past performance for each algorithm, and sensor model information such as physical sensor parameters, initial conditions, etc. The Execution module then passes the algorithm command (or a pointer to the algorithm command) and all parameters needed for its execution to the world modeling module.

2.2. Sensory Processing

The sensory processing modules of the real-time control system compare incoming data with predicted information, integrate sensory data over space and time, and determine the detection of an event. At each level of the hierarchy, this information is used to update the world model. Each sensory processing module consists of four submodules: comparators, temporal integrators, a spatial integrator, and a detection threshold (fig. 5). A specific example of how these modules interact at a given level is given in section 3.2, where the sensory processing module at Level 1 is discussed.

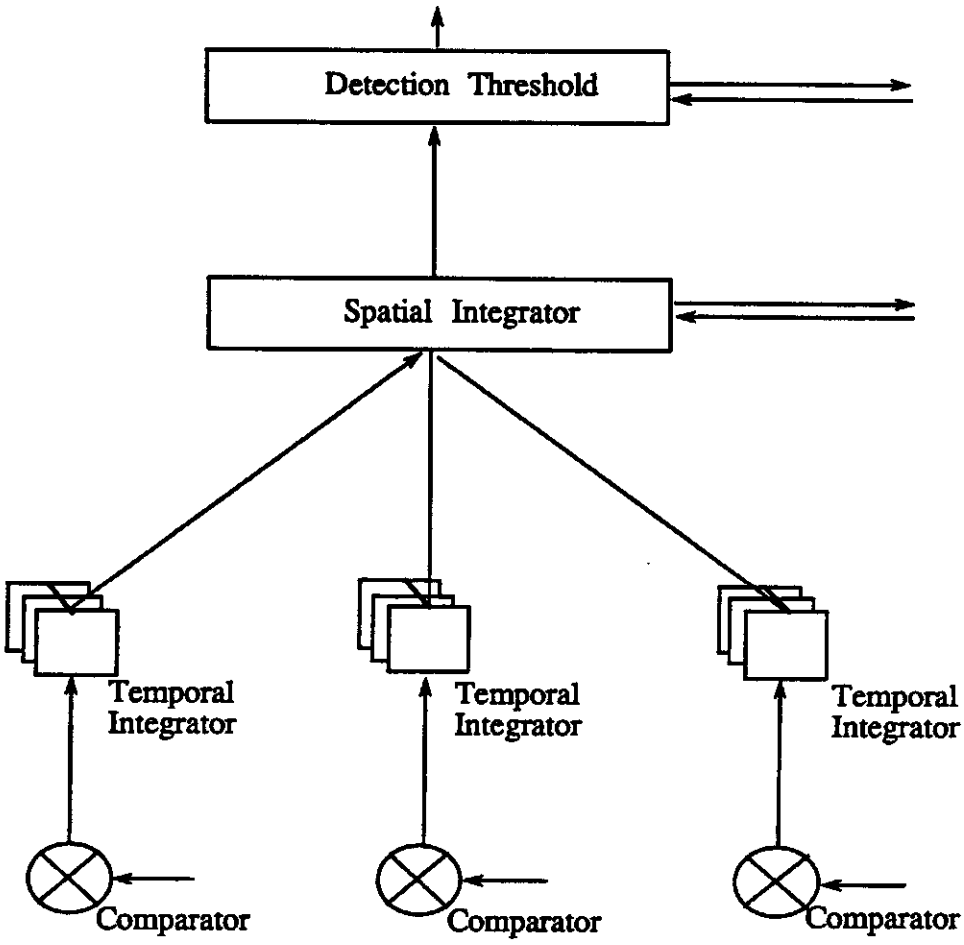


Figure 5. Submodules in the Sensory Processing System.

The order of the integrator modules can be reconfigured a priori depending on the algorithm applied. It may be appropriate for a specific application to perform temporal integration after spatial integration, such as when tracking a centroid of a moving object, or it may be unnecessary to do either spatial or temporal integration.

2.2.1. Comparator Module

The comparator modules receive input from two sources: the world model and the sensory processing module at the next lower level. The input from the world model is a model of the expected output. The input from the level below in the sensory processing hierarchy consists of the results generated by that level. The comparator modules perform algorithm specific computations using these two inputs to generate values which are passed either to the temporal integrators or the spatial integrator.

2.2.2. Temporal Integrators

Each temporal integrator combines its inputs over a given time window. The length of the time interval is supplied by the world model and depends upon factors such as timing and accuracy requirements. In addition, the window usually covers a shorter interval at lower levels of the control hierarchy and a longer interval at higher levels. The output from the temporal integrators is passed to both the world model and to the spatial integrator.

2.2.3. Spatial Integrator

The spatial integrator module integrates values over space to produce a single response value. The range of the spatial integral is supplied by the world model, and the results of the spatial integration are sent to the model to update confidence factors.

2.2.4. Detection Module

The output from the spatio-temporal integration process is passed to the detection module for evaluation or event detection. When the output surpasses a prespecified threshold, indicating correspondence between observations and the prediction of the world model, event detection occurs. An event can be defined to be the detection of an edge point, the fit of a line, or the recognition of an object, depending on the level in the control hierarchy at which the detection is occurring. The correspondence of a prediction occurs when, for example, a moving object's centroid is within a small distance from its prediction based on a past centroid measurement and the object's velocity. The results of event detection are passed to the world model to update global memory.

2.3. World Modeling

World modeling maintains the system's internal model of the world by continuously updating the model based upon sensory information. It consists of two components: support processes or functions which simultaneously and asynchronously support sensory processing and task decomposition, and the global data system which is updated by the world modeling support processes. The term *world model* refers to the two hierarchies of support pro-

cesses together with the global data system. Throughout this document, the terms world model, world model support, and global database will be used interchangeably. Any of these terms implies the combined function of the world modeling Level 1 support module and the global data system.

2.3.1. World Modeling to Task Decomposition Interfaces

The interface with the world model provides decision-making criteria to the task decomposition system. It allows the Planner module to access global memory in order to select the optimal algorithm in a given situation. The Planner uses histories of performance, timing criteria, lighting conditions, expected range to the object, etc. to choose an algorithm or to manipulate hardware. This information is stored in the world model database. The Execution module selects the parameters or initialization conditions required for sensory processing or it actually executes the control algorithm. These parameters are also stored in the world model.

2.3.2. World Modeling to Sensory Processing Interfaces

The interfaces from the world model to sensory processing allow sensory processing to read the algorithm selected by the task decomposition Planner, the parameters selected by the Execution module, and any additional command parameters, such as integral ranges. The world model support module analyzes the selected algorithm in order to provide the model required by the sensory processing comparator. In addition to providing sensory processing with an algorithm and its parameters, the world model also provides a prediction to the detection module. The prediction is a range of acceptable values that are used to determine whether an event has been successfully detected. A threshold value used in edge detection or a window for the centroid value of a moving object are two examples. The results of the sensory processing integration and detection processes are sent to the world model where they are used to update confidence factors and global memory.

3. Level 1 Interfaces and Operation

The following sections describe the functions of the task decomposition module, the sensory processing module, and the world model at Level 1 of the visual perception branch of the control system. Within the task decomposition system, the Job Assignment module accepts and queues commands from Level 2 and the human operator. The commands are passed to Planner modules which plan to activate or deactivate the camera and select the most appropriate preprocessing and/or segmentation algorithm. Execution modules are responsible for sending current to the camera actuators and obtaining algorithm parameters and writing the command, the selected algorithm, and its parameters into an area of the world memory. The sensory processing modules read the status of the camera and execute the selected algorithm on any incoming image data.

3.1. Level 1 Task Decomposition Module

Information that resides in the world model global data system is required by the task decomposition system to guide algorithm selection for the sensory processing system.

Figure 6 details the information requirements of Level 1 modules from the world model, from other processing levels, and from a human operator.

3.1.1. Level 1 Job Assignment Module

The Job Assignment module at Level 1 maintains a prioritized command queue for requests for processed data received from the Level 2 task decomposition module and/or the operator. A background or default algorithm associated with each class of processing is assigned a priority so that it is performed periodically for system reliability and is executed when no other requests are pressing. Commands received from the operator are always assigned highest priority, and the Job Assignment module places these commands at the top of the queue. In this way, operator commands are always acted upon immediately. When the Job Assignment module receives status information indicating the completion of an operator command, it reads the output information from a predefined buffer and displays it in an easily understandable manner such as a graphic display.

3.1.1.1. Level 2 to Level 1 Job Assignment Module Interface

The Job Assignment module at Level 1 interfaces with Level 2 and an operator. It accepts requests to control the camera or to choose which operations are performed on brightness pixels. All incoming commands are coordinated through this module by prioritizing them in a single queue. The contents of these commands are described in the following sections and are detailed in figure 7.

The commands from Level 2 request that either the camera be activated and that preprocessing and/or segmentation be performed on pixel data or that the camera be turned off. Each command includes some or all of the following information:

Command number

The process desiring data must be able to identify its status. Level 1 associates the condition of a request with its unique command number.

Processing request

Level 2 or the operator request the type of information to be extracted from the data. The request states which class of information to extract. For example, if edges are needed by Level 2, the direction sent may specify the need for an edge point image.

Timing requirements

The update rate of results is specified to keep current information supplied to the rest of the system. The mode may be specified as continuous so that information is processed without needing to be requested repeatedly. The results may need to be supplied within a specified amount of time so that other processes may rely on its accuracy. The velocity and acceleration of the manipulator impacts the amount of time required in locating image features. High rates of velocity and acceleration of the robot manipulator imply a high update rate.

Precision requirements

The distance between the manipulator and objects in its workspace dictate the amount of

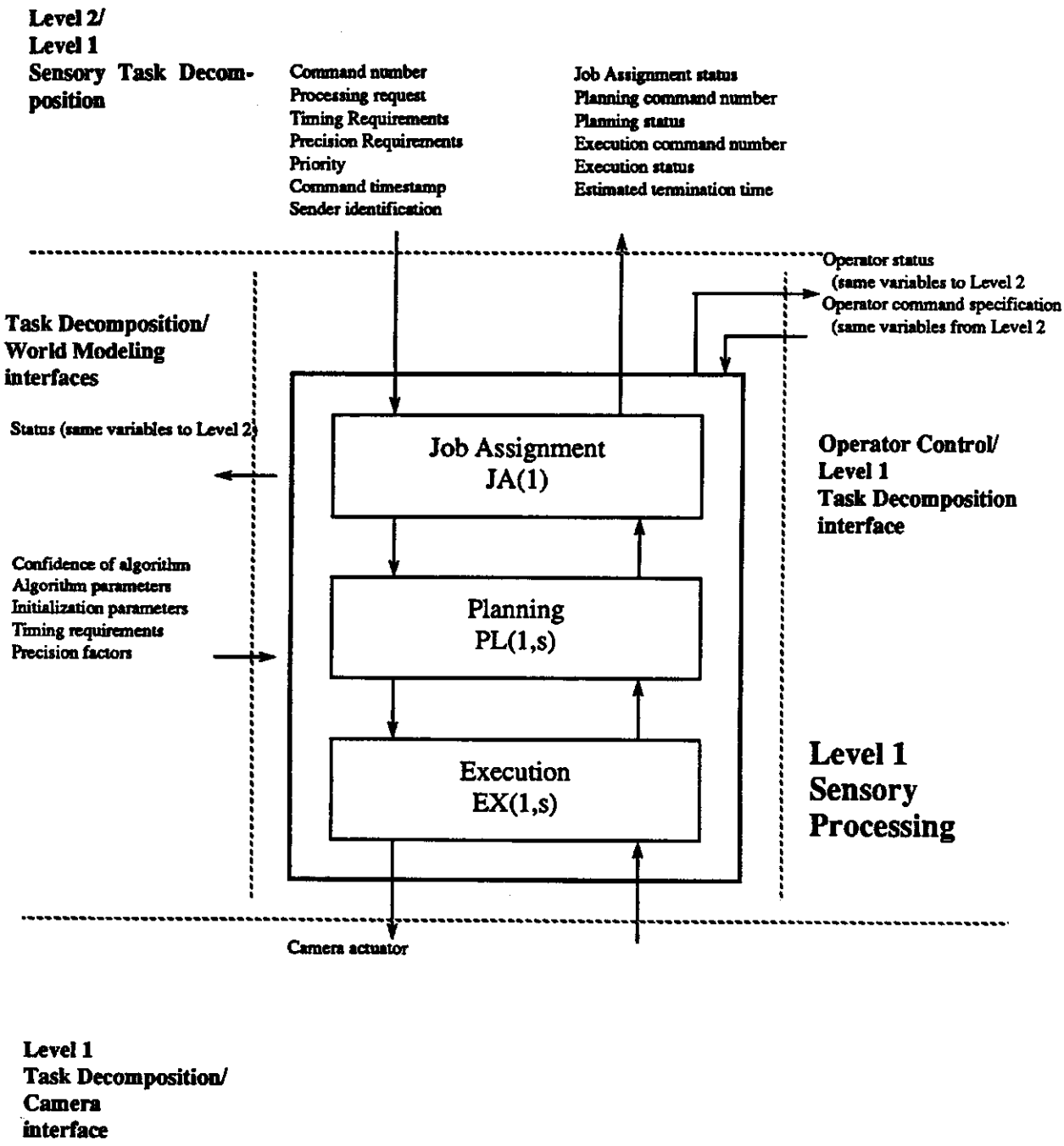


Figure 6. Level 1 Task Decomposition Module Interfaces.

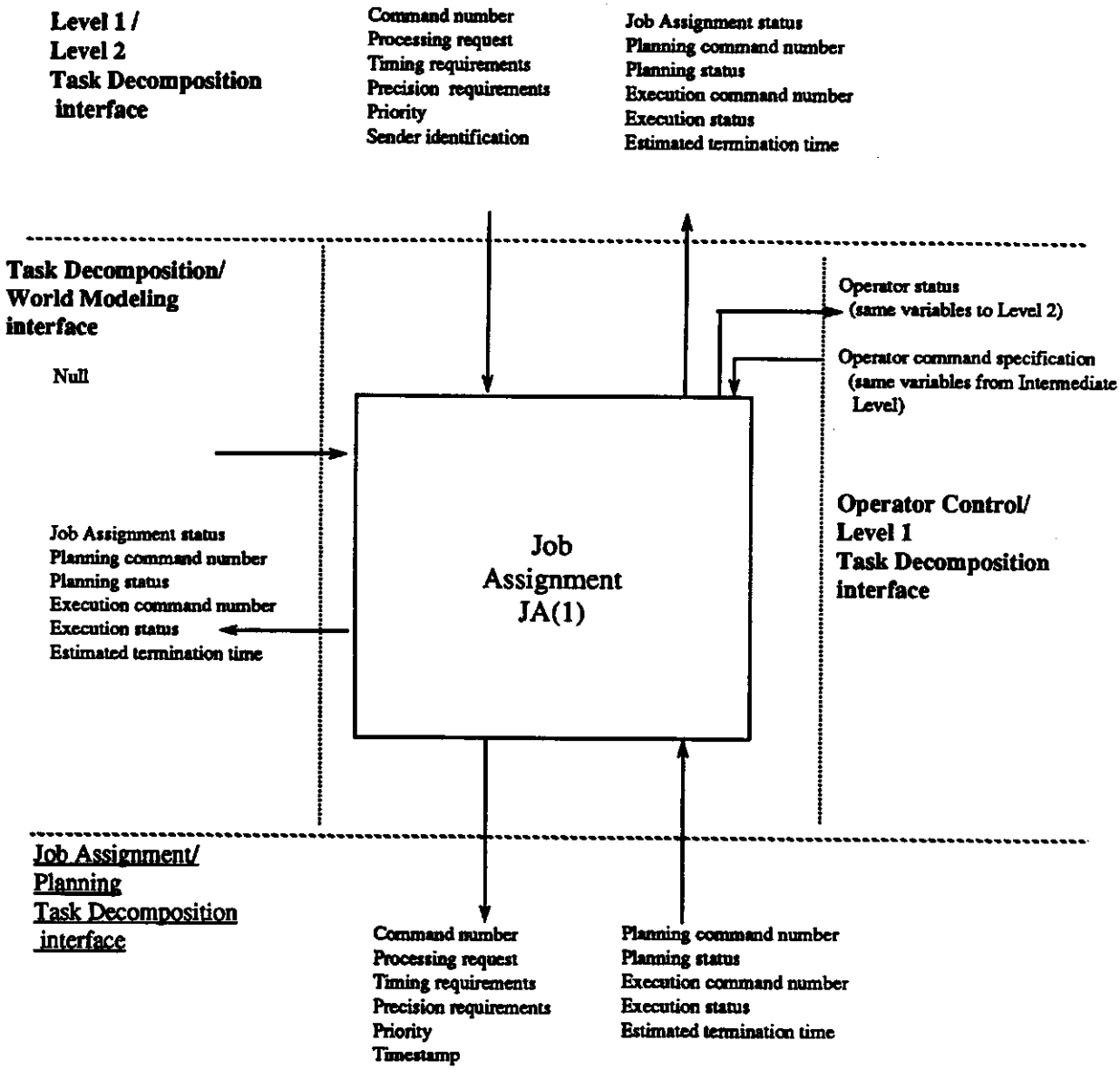


Figure 7. Level 1 Job Assignment Module Interfaces.

precision necessary when extracting features. As an end effector nears an object it is attempting to grasp, the exact location of that object becomes more crucial.

Priority

Each command is prioritized based on its relative importance to the rest of the system. The priority is assigned by the requesting process and reflects the global importance of the command to the rest of the system.

Command timestamp

The command initiation time is used to determine whether timing requirements are being met.

Sender Identification

A code identifying the sender of the command accompanies each command.

The Job Assignment module returns status to the requesting process. This information indicates whether Level 1 modules have completed execution of a request. The variables passed include:

Job Assignment status

This variable indicates the condition of the queue in the Job Assignment module. The queue may be full, empty, or accepting commands. Since operator commands are of the highest priority, the Job Assignment queue always accepts an operator command. In the event that the queue is full, the lowest priority command already on the queue is aborted. (An "abort" status is returned to the requestor.) The queue then accepts the operator's command.

Planner command number

This value reflects the number of the command that the Planner module is processing.

Planner status

The Planner module notifies other modules whether it is busy executing a command, idle and waiting for a command, or handling an error that has arisen while processing a command.

Execution command number

The Execution module records which command it is processing.

Execution status

The state of the Execution module's processing is busy or idle.

Estimated termination time

Level 1 reflects the estimated time before results are complete. Level 2 uses this parameter when deciding whether to terminate a Level 1 command or to wait until its completion.

3.1.1.2. Operator Control to Level 1 Job Assignment Interface

During subsequent execution of a command, errors may arise due to uninterpretable results. To establish more meaningful results, it is important for an operator to intervene with alternative commands. The operator modifies output by changing parameters for any given algorithm or by changing the algorithm to be executed. The operator inputs commands in a similar manner to those issued from Level 2 (See sec. 3.1.1.1) except that the sender identification field is declared "operator". Output to the operator appears as images on a monitor or easily understandable ASCII messages on a terminal.

3.1.1.3. Level 1 Job Assignment to Level 1 Planner Interface

The prioritized algorithm commands are passed from the Job Assignment module to the Planner module. The request passes the indicated information and is the same as initially defined in 3.1.1.1 and are shown in figure 8.

Command number

Processing request

Timing requirements

Precision requirements

Priority

Command timestamp.

3.1.2. Level 1 Planner Module

The Planner module in Level 1 reads the highest priority command from the Job Assignment queue. Since there are five general classes of sensory processing performed at Level 1, there are five Planner modules: one for enhancement processes, one for filtering processes, one for boundary detection, one for region detection, and one for computation of optical flow.

When an algorithm command is received, the Planner modules choose the specific technique within the class of algorithms specified by the Job Assignment module most appropriate for the type of data being processed. For example, when the camera Job Assignment module receives a command to perform a filtering operation, the Planner module chooses the appropriate filtering algorithm from the class of filtering techniques available in the sensory processing module based on time constraints, environmental conditions, the form required of the output, etc. The request for the execution of a specific algorithm is passed to the Execution module. Status information is returned to the Job Assignment module. Figure 8 explicitly shows the information passed to and from the Planner module.

3.1.3. Level 1 Execution Module

The Execution module receives either the request for camera control or the algorithm selected by the Planner module (fig. 9). In the case of algorithm selection, it reads all parameters required for the execution of the particular algorithm from the world model database. The algorithm name and supporting parameters are passed to the sensory processing mod-

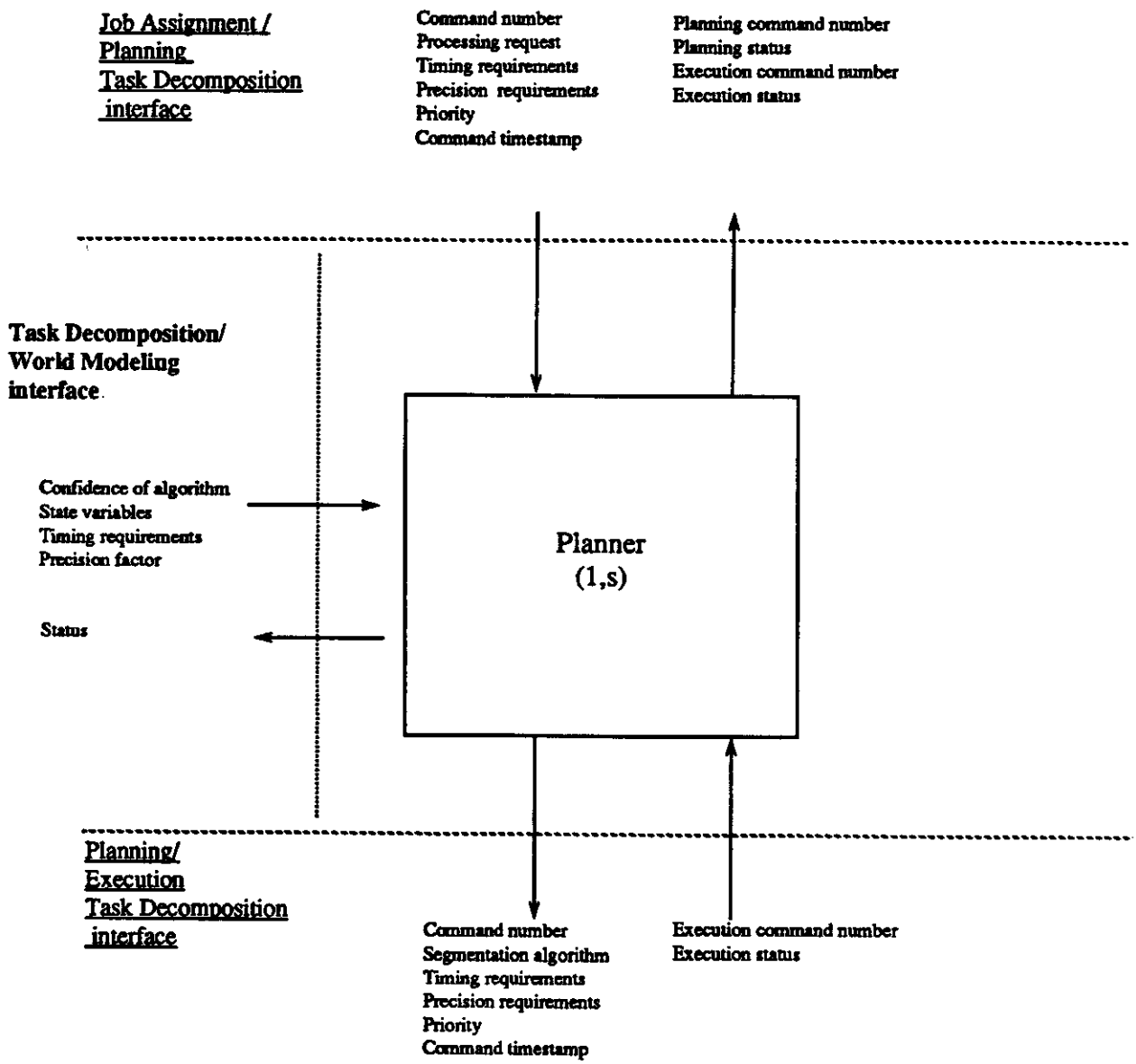


Figure 8. Level 1 Planner Module Interfaces.

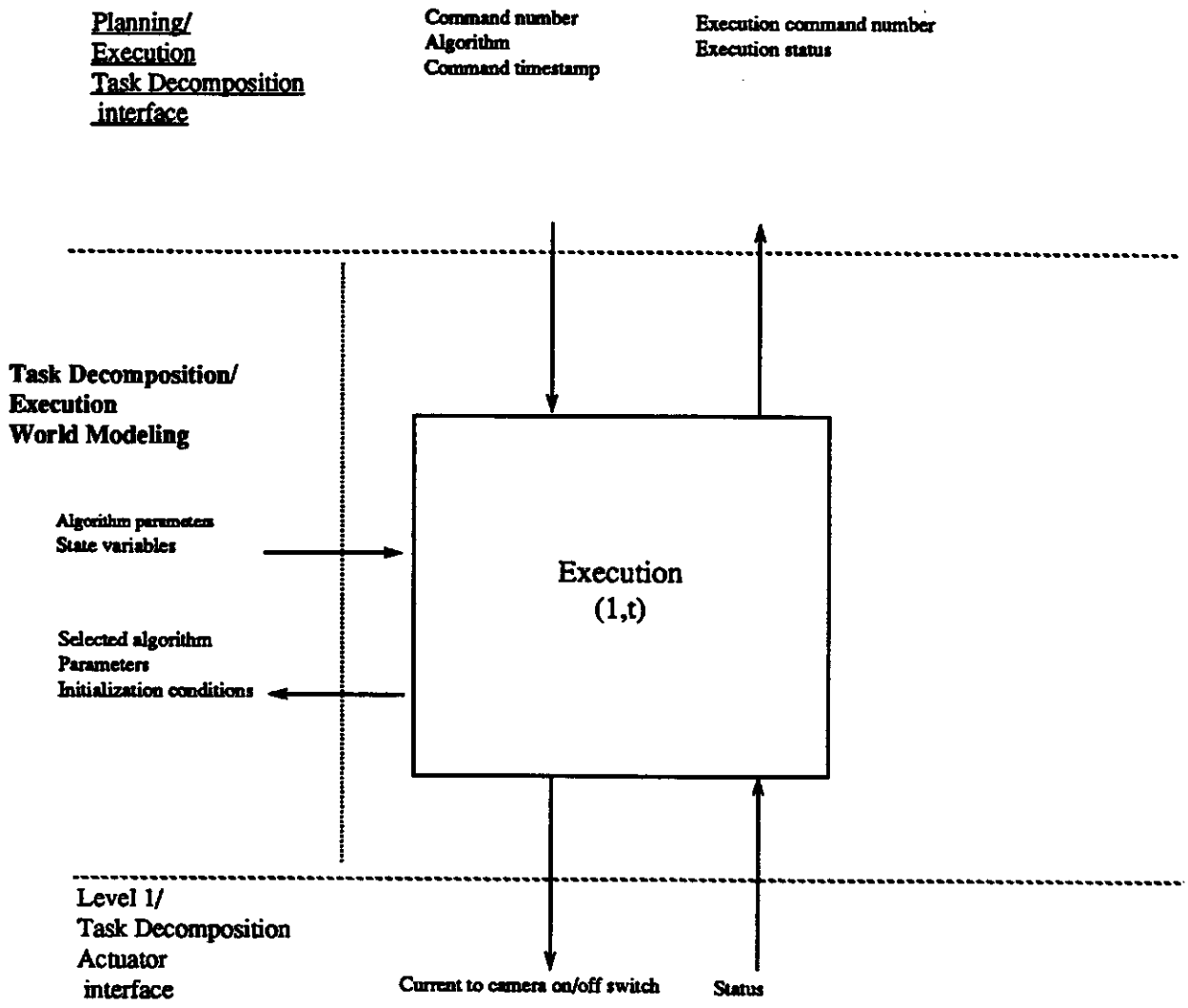


Figure 9. Level 1 Execution Module Interfaces.

ule where the actual execution of the algorithm is performed. In addition, the Execution module is responsible for activating or deactivating the camera sensor. It acts upon commands received from the planner module to activate or deactivate the camera at the appropriate time.

3.2. Level 1 Sensory Processing Module

Sensory processing at Level 1 accepts pixel brightness values as input and processes each pixel according to the algorithm chosen by the task decomposition Planner module. Each pixel is passed through the comparator module, the temporal integrators, the spatial integrator and the detection module. The pixels can be enhanced or filtered using the algorithms described in Appendix A, or they can be categorized according to their grey level characteristics into edge pixels, surface patch pixels, or pixels of motion. The latter methods are described in Appendix B. To clarify the type of processing done at Level 1, figure 10 depicts the functions of the sensory processing modules for labelling pixels as edge or non-edge points using the Sobel edge detection method (Appendix B8.1.1).

In the comparator module, pixel brightness value input is received from the camera sensor. Conceptually, there is a dedicated comparator for each pixel in the image array. The prediction supplied by the world model which can consist of the 3 x 3 convolution mask described in Appendix B8.1.1, is a model of the feature to be tested at that pixel location. As shown in figure 10, each input pixel in the image is multiplied by the appropriate element of the Sobel edge detector mask.

The output generated by the multiplication of the pixel's intensity value and its corresponding value in the Sobel mask is passed to the temporal integrators. The results from an averaged sequence of pixels at the same location in the image are gathered over a time span defined in the world model.

The temporally integrated pixels are passed to the spatial integrator. The size of the spatial window is defined in the world model. The pixels are summed over this window.

Lastly, the results of the spatio-temporal information are evaluated by comparing the results to a threshold parameter stored in the world model. Pixels which exceed this threshold value are labelled edge points, while those pixels falling below the threshold are labelled non-edge points.

3.3. Level 1 World Model

Input to world modeling from Level 1 sensory processing consists of point features extracted from sensor readings. This information is used to update confidence factors in the model and in the global map. Point data from the spatial integrator is stored by world modeling to be further processed by higher level sensory processing modules or to be fused with data from other sensors. Associated with each reading is a sensor identification number which includes both the sensor type and the instance of the sensor. For example, the sensor identification number might specify that the reading is coming from pixel i, j from camera k. In addition, each sensor reading has an associated timestamp t. Thus, we can refer to B(i, j, k, t) as the brightness of pixel i, j from camera k at time t.

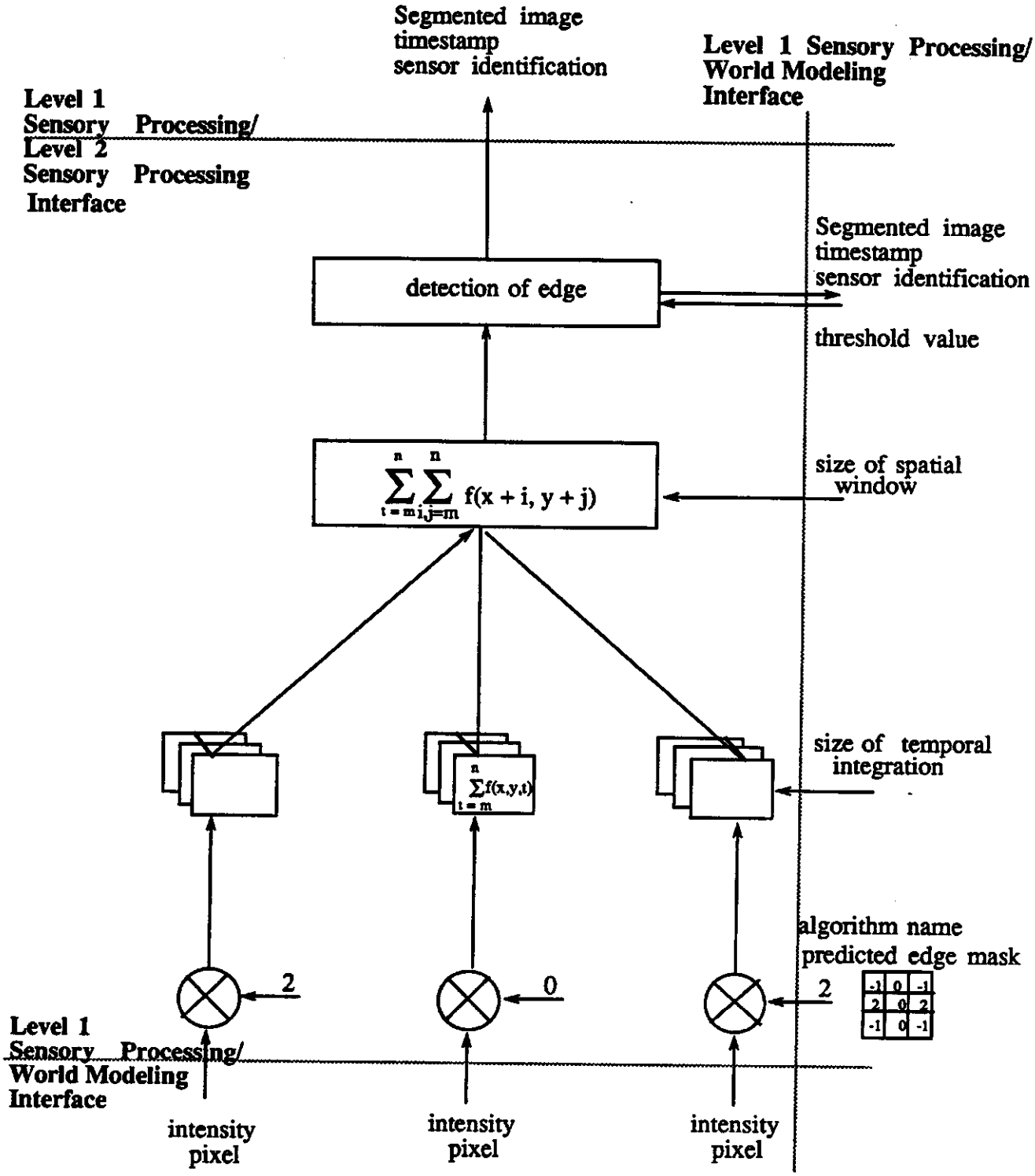


Figure 10. Sensory Processing for a Sobel Edge Detection Algorithm.

Each Level 1 sensory processing module has a corresponding support module in the world model. After the Level 1 module processes the data, the world modeling module accepts the results and transforms it, if necessary, to the coordinate system specified by task decomposition. The data are then stored in the global data system for use by the task decomposition module, as well as other world modeling processes.

Input to the world model from task decomposition consists of the specific algorithm selected by the Planner to be performed in Level 1 sensory processing. The Planner accesses the world model global memory to select the best algorithm for the situation. The world model passes its prediction to sensory processing based on the algorithm selected. For example, if the Planner selects the Sobel edge detection algorithm, the world model passes the Sobel edge mask (Appendix B8.1.1) as its prediction to sensory processing.

4. A Vision System Application

To clarify the operation of the computational triple at Level 1 for a camera, this section provides a specific example of system interfaces. Assume that Level 3 initiates a command for information required for tracking a particular surface of a moving part. Further, assume that task decomposition has selected a particular instance of a camera and positioned it appropriately. The world model contains information about the object model and an initial prediction of the location of the part. Level 2 is directed to extract the symbolic information required to define the features of the object surface and their attributes (the centroid of the object aspect, the equations of its boundaries, their length and their orientation) by the priority levels set by the Level 2 Planner. The input it requires for these computations resides in buffers written by Level 1. Similarly, Level 1 is directed to segment its data in accordance with priority levels set by its Planner.

The remainder of this section describes in detail the role of the Level 1 processing unit associated with the particular camera in this example. The sensor plan stored in the world model generates requests to the sensory processing module in response to the need for updated information. These requests are created by assigning priority levels to the classes of output produced by Level 1. For the sake of example, we assume that the plan requires an edge point image 20 times per second and a surface patch image of the same scene 10 times per second. When these commands are received by the Job Assignment module, they are prioritized (edge point images will be processed more frequently than surface patch images) and placed on the queue.

The Planner module receives its commands from the top of the Job Assignment queue. In order to obtain an edge point image, it must decide which algorithm among all the gradient extraction algorithms residing in the sensory processing module is most likely to provide satisfactory results in this particular situation. The Planner module determines the best algorithm based on a performance history residing in the world model.

The Planner module also considers the update rate required by the plan. The execution time of each algorithm is known to the system and is stored in a world model parameter. In addition to timing requirements, the Planner module also must take into account the accuracy required. For example, when the camera is far from the object being tracked, interior texture information is not visible, and does not have to be smoothed from the image. However, texture could be visible in a closer view of the same object and must be removed in order to

avoid false edge information. Thus different algorithms are used depending on the distance between the camera and the object, lighting conditions, object surface reflectivity, etc.

The Planner module performs the same type of analysis in choosing an appropriate region classification routine. Assume that after analyzing all factors, it requests output from a series of two Sobel edge detection algorithms (Appendix B8.1.1) followed by a thresholded image (Appendix A7.2) for a period of thirty seconds. The Execution module reads any parameters that are required by the individual algorithms from the world model global memory area. In this example, the edge point algorithm needs a threshold value to suppress low magnitude edges, and the thresholding algorithm needs a parameter for converting grey scale pixels to binary pixels.

The Execution module passes the algorithm name and the associated parameters to the world model. This results in the sensory processing module activating the chosen algorithms from each class of algorithms: Sobel edge detection from the class of gradient extraction techniques and thresholding from the class of region classification techniques. These algorithms are cyclically executing processes, and are continuously reading raw camera data. The output from each algorithm is written into a predefined buffer area where it is available to be read by the requesting process. The sensory processing module appends a timestamp to the results based on the system time at which the algorithm was initiated. Status information is sent to the world model module at the completion of processing.

5. Conclusion

This document has described Level 1 of the perception branch of a realtime control system hierarchy. The components and functions of the computational triple of task decomposition, world modeling, and sensory processing were defined, and the specific functions of each component were discussed. Interfaces between the modules, including the operator, have been defined. Appendix A discusses the realtime data enhancement and filtering algorithms capable of being performed in the sensory processing module. Appendix B discusses gradient extraction algorithms, surface patch extraction algorithms, and optical flow algorithms. The concept of grouping these algorithms into classes of algorithms allows the flexibility of adding or deleting algorithms at any stage of implementation without changing the structure of the system.

Although specific hardware requirements are not defined in this document, the amount of array information required to be processed at this level (~64 K bytes of information per image), suggests the use of parallel processing machines for realtime output. Many algorithms implemented at Level 1 operate on image data by using local information in a non-sequential manner. Because of the large amount of data to be processed and the need to process that data as close to video rate as possible, most serial computers cannot meet the requirements of Level 1 processing. Parallel computers have been developed in recent years to specifically fulfill the need of real-time processing of image data [ASPEX87, KENT85, LUMIA85], and although the machines differ in architectural design and implementation, they share the goal of being able to process an entire image or a region of an image in real-time.

6. References

[ALBUS81] Albus, J. S., Brains, Behavior, & Robotics, Byte Books, Subsidiary of McGraw-Hill, 1981.

[ALBUS87] Albus, J. S., McCain, H. G., and Lumia, R., NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM), NASA Document SS-GS-FC-0027, July, 1987.

[ARKIN87] Arkin, R. C., Riseman, E. M., and Hansom, A. R., "AuRA: An Architecture for Vision-Based Robot Navigation", Proceedings: Image Understanding Workshop, February, 1987.

[ASPEX87] Aspex, Inc., "PIPE--An Introduction to the PIPE System", 1987.

[BALLA82] Ballard, D., and Brown, C., Computer Vision, Prentice Hall, 1982.

[BARRO81] Barrow, H. G. and Tenenbaum, J. M., "Interpreting Line Drawings as Three Dimensional Surfaces", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 75-116.

[BEAUC80] Beauchamp, K., Data Acquisition for Signal Analysis, George Allen & Unwin, 1980.

[BINFO81] Binford, T. O., "Inferring Surfaces from Images", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 205-244.

[BRADY81] Brady, J. M., "Preface - The Changing Shape of Computer Vision", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 1-16.

[BROOK81] Brooks, R. A., "Symbolic Reasoning Among 3-D Models and 2-D Images", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 285-348.

[BROWN88] Brown, C., ed., "Advances in Computer Vision", Vol. 1, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1988.

[BROWN86] Browne, A., and Leonard, W. N., "Vision and Information Processing for Automation", Plenum Press, New York, 1986.

[CANNY86] Canny, J. "A Computational Approach to Edge Detection", IEEE Transactions on PAMI, Vol. 8, No. 6, November 1986, pp. 679-698.

[CROWL84] Crowley, J. L., and Parker, A. C., "A Representation for Shape Based on Peaks and Ridges in the Difference of Low-Pass Transform", IEEE Transactions on PAMI, Vol. PAMI-6, No. 2, March 1984, pp. 156-170.

[DAILY86] Daily, M., Harris, J., and Reiser, K., "Detecting Obstacles in Range Imagery", Hughes Artificial Intelligence Center, 1986.

[DAVIS80] Davis, L. S., and Rosenfeld A., "Cooperating Processes for Low-Level Vision: A Survey", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, 1980, pp. 245-264.

[DAVIS86] Davis, L. S., "Image Texture Analysis Techniques - A Survey", Vision and Information Processing for Automation, Arthur Browne, ed., Plenum Press, New York,

1986.

[DORF83] Dorf, R. C., Robotics and Automated Manufacturing, Reston Publishing Co. Inc., 1983.

[DUNCA88] Duncan, J. H., and Chou, T-C., Temporal Edges: The Detection of Motion and the Computation of Optical Flow:, CS-TR-2042, University of Maryland, 1988.

[FIALA88] Fiala, J., "Manipulator Servo Level Task Decomposition", NBS Document ICG #002, NIST Technical Note 1255, October, 1988.

[FLINC81] Flinchbaugh, B. E., and Chandrasekaran, B., "A Theory of Spatio-Temporal Aggregation for Vision", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, August 1981, pp. 387-408.

[FREI77] Frei, W., and Chen, C., "Fast Boundary Detection: A Generalization and a New Algorithm", IEEE Transactions on Computer Vision, Vol C-26, No. 10, 1977, pp. 988-998.

[GONZA77] Gonzalez, R., and Wintz, P., Digital Image Processing, Addison-Wesley Publishing Co., 1977.

[GROOV86] Groover, M., Weers, M., Nagel, R., and Odrey, N., Industrial Robotics, Mc-Graw- Hill, 1986.

[GROSS85] Gross, T., Lam, M., and Webb, J., "WARP As A Machine for Low Level Vision", IEEE Conference on Robotics and Automation, 1985.

[HALL71] Hall, E. L., Kruger, R. P., Dwyer, S. J., and Hall, D., "A Survey of Preprocessing and Feature Extraction Techniques for Radiographic Images", IEEE Transactions on Computers, Vol. C-20 #9, 1971.

[HARAL87] Haralick, R., "Recognition Methodology Algorithms and Architecture", SPIE Vol 755, Image Pattern Recognition Algorithms, Implementations, Techniques and Technologies, 1987.

[HILLE82] Hilles, W. D., "A High Resolution Imaging Touch Sensor", International Journal of Robotics Research, Vol 1 #2, MIT Press, 1982.

[HONG89] Hong, T. H., "Using Known Camera Motion to Detect Targets from Image Sequences", National Institute of Standards and Technology, Gaithersburg, MD,1989 [TBP].

[HORN81] Horn, B. K., and Schunck, B. G., "Determining Optical Flow", Artificial Intelligence #17, 1981.

[HUECK71] Hueckel, M. H., "An Operator Which Locates Edges in Digitized Pictures", Journal of the ACM, Vol. 18, No. 1, January 1971, pp. 113-125.

[IKEUC81] Ikeuchi, K., and Horn, B. K. P. , "Numerical Shape from Shading and Occluding Boundaries", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 141-184.

[INTEG81] Integrated Computer Systems, Digital Image Analysis and Understanding Course Notes, March,1981.

[JARVI84] Jarvis, R. A., "Application-Oriented Robotic Vision - A Review", Robotica,

Vol. 2, 1984, pp. 3-15.

[KAFRI84] Kafrissen, E., and Stephans, M., Industrial Robots and Robotics, Reston Publishing Co. Inc., 1984.

[KALMA80] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems", Transactions of the ASME, Journal of Basic Engineering, Vol. 82D, March 1980.

[KANAD81] Kanade, T., "Recovery of Three-Dimensional Shape of an Object from a Single View", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 409-460.

[KEARN87] Kearney, J. K., Thompson, W. B., and Boley, D. L., "Optical Flow Estimation: An Error Analysis of Gradient Based Methods with Local Optimization", IEEE, PAMI, 1987.

[KELMA88] Kelmar, L., "World Model Level 1", NIST Technical Note 1258, December 1988.

[KENT85] Kent, E., Shneier, M., and Lumia, R., "PIPE-Pipelined Image Processing Engine", Journal of Parallel and Distributed Computing, Vol 2, 1985, pp. 50-78.

[KITCH82] Kitchen, L., and Rosenfeld, A., "Grey-level Corner Detection", Pattern Recognition Letters, December 1982, pp. 95-102.

[KOHL] Kohl, C. A., Hanson, A. R., and Riseman, E. M., "Goal-Directed Control of Low-Level Processes for Image Interpretation".

[LIM87] Lim, H. S., and Binford, T. O., "A Survey of Parallel Computers", Image Understanding Workshop Proceedings, February, 1987.

[LUMIA85] Lumia, R., Shneier, M., and Kent, E., "A Real-Time Iconic Image Processor", IEEE International Conference on Robotics and Automation, 1985, pp. 873- 878.

[MARR75] Marr, D., "Early Processing of Visual Information", MIT AI Memo 340, 1975.

[MARR77] Marr, D., and Poggio, T., "A Theory of Human Stereo Vision", Memo 451, Artificial Intelligence Lab, MIT, Cambridge, Mass., November, 1977.

[MAYHE81] Mayhew, J. E. W., and Frisby, J. P., "Psychophysical and Computational Studies Towards a Theory of Human Stereopsis", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 349-386.

[MERO75] Mero, L., and Vassy, Z., "A Simplified and Fast Version of the Hueckel Operator for Finding Optimal Edges in Pictures", Proceedings of the Fourth International Joint Conference on Artificial Intelligence, 1975, pp. 650-655.

[NEVAT77] Nevatia, R., "Evaluation of a Simplified Hueckel Edge-Line Detector", Computer Graphics and Image Processing, Vol. 6, 1977, pp. 582-588.

[NAVAT80] Nevatia, R., and Babu, K. R., "Linear Feature Extraction and Description", Computer Graphics and Image Processing, 13, pp. 257-269, 1980.

[NAZIF84] Nazif, A. M., and Levine, M. D.,"Low Level Image Segmentation: An Expert System", IEEE Transactions on PAMI, Vol. PAMI-6, No. 5, September 1984, pp. 555-577.

[NISHI81] Nishihara, H. K., "Intensity, Visible-Surface, and Volumetric Representation", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 265-284.

[O'GORM78] O'Gorman, F., "Edge Detection Using Walsh Functions", Artificial Intelligence, Vol. 10, 1978, pp. 215-223.

[REDFO86] Redford, A. H., Lo, E., Robots in Assembly, Halsted Press, 1986.

[ROSEN88] Rosenfeld, A., "Robot Vision", Machine Intelligence and Knowledge Engineering for Robotic Applications, Andrew K. C. Wong and Alan Pugh, eds., Springer-Verlag, New York, 1988.

[ROSEN82] Rosenfeld, A., and Kak, A., Digital Picture Processing, Volume 1 Second Edition, Academic Press, 1982.

[SINGH87] Singh, A., "Image Processing on PIPE", Philips Laboratories, Document #TN-87-083, July, 1987.

[STANL84] Stanley, W. D., and Dougherty, G. R., Digital Signal Processing, Prentice Hall, 1984.

[STEVE81] Stevens, K. A. "The Visual Interpretation of Surface Contours", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 47-74.

[TANIM78] Tanimoto, S., "Regular Hierarchical Image and Processing Structures in Machine Vision", Computer Vision Systems, Academic Press, 1978.

[TECHN87] Technical Arts Corporation, User's Manual-White Scanner 100A, 1984.

[VEATC87] Veatch, P., Davis, L., "Range Image Algorithms for the Detection of Obstacles by Autonomous Vehicles", U. of MD. Center for Automation Research, Report #CAR-TR-309, July,1987.

[WAVER88] Wavering, A., "Manipulator Primitive Level Task Decomposition", NBS Document ICG #003, NIST Technical Note 1256, October, 1988.

[WAXMA87] Waxman, A. M., and Bergholm, F., "Convected Activation Profiles and Image Flow Extraction", LSR-Tech. Report 4, Sensory Robotics, College of Engineering, Boston Univ., 1987.

[WESZK78] Weszka, J. S., "A Survey of Threshold Selection Techniques", Computer Graphics and Image Processing, 1978.

[WITKI81] Witkin, A. P., "Recovering Surface Shape and Orientation from Texture", Artificial Intelligence, Vol. 17, Nos. 1-3, August 1981, pp. 17-46.

7. Appendix A: Preprocessing Techniques

This section discusses preprocessing techniques that are applied in the sensor processing module of Level 1 of the perception hierarchy. The input to these algorithms is considered to be an individual pixel (image point), and the output is in the form of a processed pixel. An image is considered to be the spatial integration of individual pixels in a camera array. For convenience, the terms "image" and "data array" are used interchangeably.

A7.1. Array Data Enhancement

Image preprocessing consists of "an application-dependent technique for enhancing preselected features or for removing irrelevant detail" [HALL71]. The effectiveness of data enhancement techniques is dependent on the information being analyzed and the environmental conditions under which that data was generated. The causes of a poor image can be non-optimal lighting conditions (either too little or too much illumination), specularity of the objects in the scene, inappropriate viewing angle, poorly visible details, noise, etc. Not all of these problems can be improved with data enhancement techniques (if an important feature is occluded, no preprocessing technique can make it visible), but many enhancement techniques exist for improving degradations. This section will discuss methods of filtering, smoothing, thresholding and enhancing contrast for improving data quality.

The input data to be enhanced consists of rectangular arrays of digitized information. The size of the array is dependent on the sensor from which the data was read. Individual pixels are addressed by their row and column position in the array (fig. A1).

The input array can also be a binary image which contains only two values, black and white. Binary arrays provide useful information when there is a high contrast between the objects of interest in the scene and the background. Grey scale information can be converted to binary information by using specialized hardware in the digitizer or by software techniques which will be described later in this section.

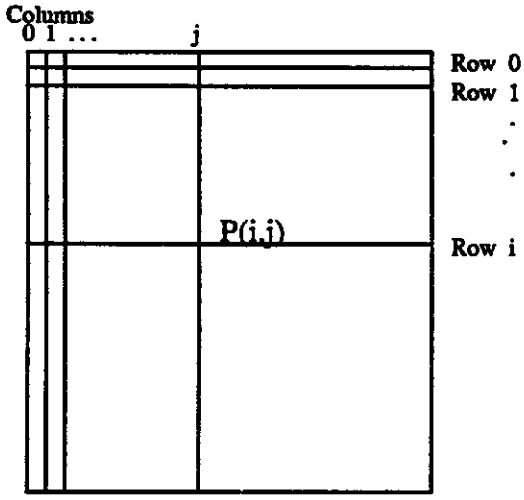


Figure A1. Pixel Position.

A7.2. Thresholding

Thresholding of an image is a preprocessing technique that segments a grey level array into a binary array containing "object" and "background" areas. In order to convert the input data into a binary representation, a threshold value, T, must be chosen. All grey scale values in the original array whose intensities are less than T are assigned an output value of 0 (black), while those whose intensities are greater or equal to T are assigned a value of 255 (white). The effectiveness of this technique is heavily dependent on the scene; it is useful in situations where there is good contrast between the object of interest and background. It is not an effective method of segmenting a complex scene or one in which the objects of interest are specular [WESZK78].

Choosing a value T to use as a threshold value varies among images, and a histogram of the image is required to choose that value. A histogram is a graph of the frequencies with which each grey level in the image occurs. In many images, the objects of interest fall in one range of grey levels while the background falls in another range. By choosing the threshold between the two peaks, good segmentation results have been obtained [ROSEN82].

A7.3. Contrast Enhancement

Contrast enhancement is a method used to improve the clarity of details in an image. Because of variable lighting conditions, especially in the environment of the space station, camera data is usually compressed either at the low end of the histogram (dark image) or at the high end (light image) (fig. A2). Histogram equalization is a technique that stretches high concentrations of grey levels while compressing less populated grey levels (fig. A3). It creates a transformation that enhances contrasts and brings out details in poorly contrasted or heavily shadowed portions of the image. [BALLA82].

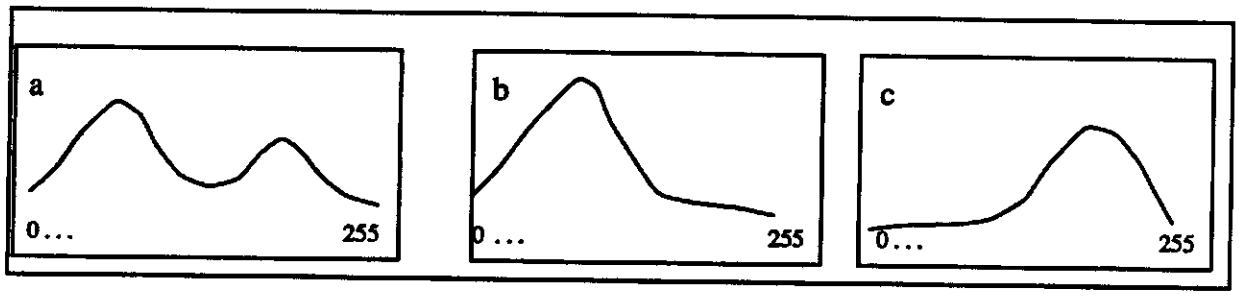


Figure A2. Histograms a. Ideal b. Dark c. Light.

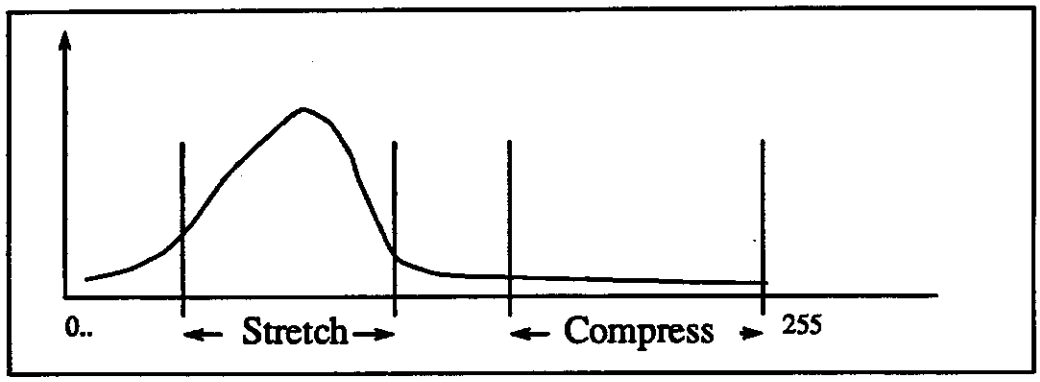


Figure A3. Histogram Equalization.

The result of this transformation is to spread the grey level intensities more evenly throughout the image, sharpening details. A potential disadvantage of contrast stretching is that a global property of the image, the histogram, is used to generate a local operation. Therefore, when there is a large variation in the grey level values of the background, the equalization transform will tend to stretch the un-interesting ranges [INTEG81].

A7.4. Smoothing

Smoothing data is a preprocessing technique for enhancing the appearance of images by diminishing the effects of noise. It is a form of spatial integration. Although there are many benefits to be gained by removing image noise, smoothing tends to blur the original image and therefore to de-emphasize sharp edges and contours [ROSEN82]. There are many methods used for smoothing data: neighborhood averaging, edge preserving smoothing, low pass filtering, shrinking and expanding, pyramiding, etc. [BALLA82, GONZA77, ROSEN82]. These methods are discussed in this section.

A7.4.1. Averaging

Averaging is a technique for reducing spurious noise in an image. It can be considered to be a special case of low-pass spatial filtering (see sec. 3.3.4). Averaging can be done as either temporal integration over successive images or as spatial integration in a single image [ROSEN82]. Processing in the temporal domain is useful when there are multiple instances of the same scene, i.e. a stationary scene, and where the noise values present in the images are independent of each other and have a mean value of 0. For example, if there are n images of a single scene, $I_1, I_2, \ldots I_n$, each pixel in the averaged output image G is computed as:

$$G(x,y) = (I_1(x,y) + I_2(x,y) + \ldots + I_n(x,y)) / n$$

The noise values in the input images will be blurred (the degree of blurring depends on the number of input images averaged) while the objects in the image remain unchanged. Averaging images in a moving environment produces more blur.

Averaging in a single image involves a local operation over neighborhoods in the image.

A neighborhood of a point is defined as those points surrounding the point (fig. A4).

a	b	c
d	f(x,y)	e
f	g	h

Figure A4. Points in the Neighborhood of Point f(x,y).

For every pixel g(x,y) in the spatially integrated output,

$$g(x,y) = \left(\sum_{i=1}^{n} f_i(x,y)\right)/n \quad \text{where}$$

n = the total number of points in the neighborhood of f(x,y)

$f_i(x,y)$ = grey level of all points in the eight neighborhood of f(x,y) [GONZA77].

Neighborhood averaging is an effective method of reducing fine-grained noise but, depending on the size of the neighborhood being averaged, can result in an image where boundaries, as well as noise, are blurred. Averaging over larger neighborhoods produces greater blurring. This blurring effect can be reduced by combining the averaging operation with a thresholding operation. A point which differs by less than a specified threshold T from its averaged neighbors is left unchanged. Thus

$$\text{if } (f(x,y) - (\Sigma f(n,m))/M > T)$$

$$g(x,y) = (Sf(n,m))/M$$

else

$$g(x,y) = f(x,y)$$ [GONZA77].

A7.4.2. Edge Preserving Smoothing

Edge preserving smoothing is a technique which performs a local blurring on an image to suppress noise without blurring any edges that might be present in the image [ROSEN82, SINGH87]. Noise values are suppressed only at selected points. Implementation of this scheme is based on detecting edges and determining edge directions in the image and then performing an averaging operation on only non-edge pixels. This operation can be iterated to weaken noise without affecting edges.

A7.4.3. Median Filtering

Another smoothing technique which does not blur or smooth edges is median filtering. Rather than averaging the points in a neighborhood around a point f(x,y), the output value

g(x,y) is set equal to the median value of the points in the neighborhood. Because edge points are not weakened by this operation, the operation can be iterated a fixed number of times to reduce noise values to an acceptable level.

A7.4.4. Low-Pass Filtering

Low-pass filtering is a technique that uses information from the frequency domain to enhance information in the spatial domain. In the frequency domain, an image is grouped into different frequency band widths, each of which contains unique information. The Fourier Transform maps the spatial domain onto the spatio-frequency domain. It is defined as:

$$\Im [f(x,y)] = \iint f(x,y) \exp[-j2\Pi(ux + vy)]\, dx\, dy$$

and its inverse

$$\Im^{-1}[F(u,v)] = \iint F(u,v) \exp[j2\Pi(ux + vy)]\, du\, dv \quad \text{where}$$

x,y = image coordinates

u = frequency in the x direction

v = frequency in the y direction

The primary advantage of frequency domain processing is that any arbitrary frequency response is easy to implement. Because of the reversibility of the transform, many properties that are inherent to the frequency domain can be used in the spatial domain which is a more natural and intuitive representation. In the image representation, local operators can be applied to the image to attenuate or completely suppress information in all other frequencies [GONZA77]. Since edges and other sharp transitions contribute heavily to the high frequency portion of an image's Fourier Transform, the smoothing operation can be performed by attenuating a specified range of high frequency components. A low-pass filter is one which filters out high frequency information (edges) and passes low frequency information. This results in a blurred or smoothed image. A Gaussian convolution applied over all points in the image is an example of a low-pass filter which reduces noise in those parts of an image where there are no strong edges. A side effect of this operation is that portions of the image containing a large intensity gradient are also blurred.

A7.4.5. Binary Edge Smoothing

Noise removal in a binary image is a more simple operation than grey scale image smoothing. Because a thresholded image consists only of objects and background values, noise can be misinterpreted as "object" and therefore must be removed. One method of noise removal involves a shrinking and expanding operation [ROSEN82]. The shrinking operation examines the neighborhood of each black point in the image and changes its value to white if any neighbors are white.

For all values n,m in the neighborhood of f(x,y)

if (f(x,y) == black) && (f(n,m) == white))

g(x,y) = white

else

g(x,y) = black.

The expanding operation performs the reverse operation:

if (f(x,y) == white) && (f(n,m)== black)

g(x,y) = black

else

g(x,y) = white.

This process completely removes any noise value that is smaller than two pixels wide, and the expanding step restores larger objects without restoring the noise.

Framegrabbers that convert grey scale information into binary information often contain additional hardware that can filter noise in an image as it is being digitized. The removal of noise and the width of the noise to be suppressed is a user option sent to the framegrabber when the image is to be read.

A7.4.6. Multi-Resolution Processing

Multi-resolution processing or image pyramids offer additional methods for image enhancement. A pyramid is an "image data structure consisting of the same image at several successively decreasing levels of resolution" [BALLA82]. The image at each level of the pyramid is formed by replacing each neighborhood at the n^{th} level of the pyramid with a single pixel at the n+1 level (fig. A5). The resultant levels of images, each of which is one quarter the size of its next lower level, resemble a pyramid (fig. A6).

The multi-resolution method of smoothing an image involves averaging each 2 x 2 neighborhood of the image at level n and placing the value of the neighborhood average in the level n+1 image. This operation can be repeated for two or three levels of the pyramid. The smoothed image is restored to full resolution by expanding, i.e. mapping each pixel at level n+1 into four pixels at level n, and interpolating the results to remove "blocking" effects. This operation is repeated until the full resolution image is restored

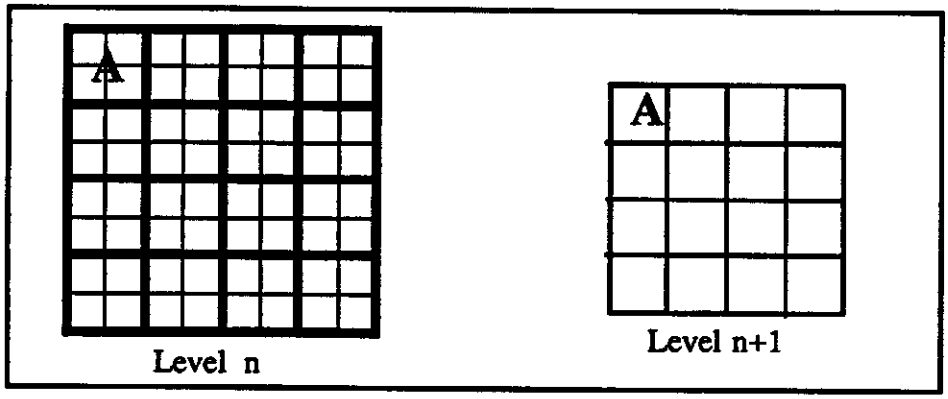


Figure A5. Forming Levels of a Pyramid.

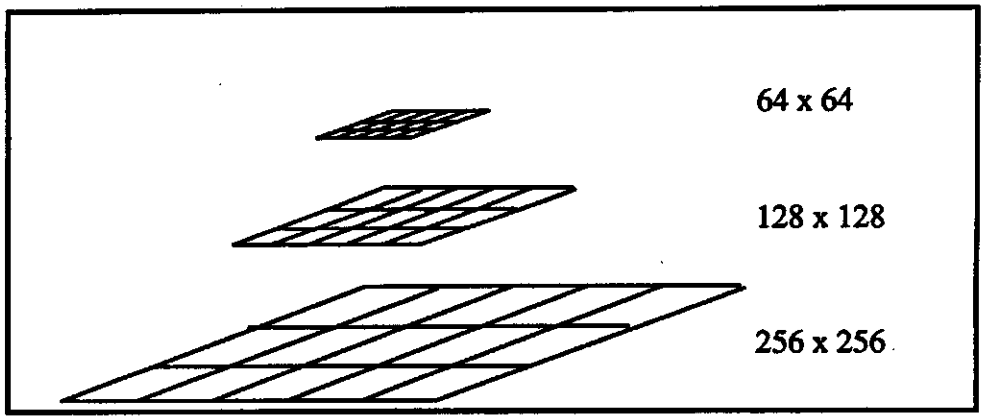


Figure A6. Three Levels of a Pyramid.

A7.5. Sharpening

The purpose of sharpening an image is to emphasize edges or areas of intensity discontinuities. Since the goal of this operation can be considered opposite to that of smoothing, the techniques involved in sharpening images are theoretically opposite to those used in smoothing: smoothing entails integration and sharpening entails differentiation. Two types of image sharpening techniques are discussed in this section: differentiation and high-pass filtering.

A7.5.1. Differentiation

The computation of gradients is the most common method of differentiation of an image. The gradient operation not only extracts the local edges which represent areas in the image where grey levels are changing rapidly, but also can be used to provide information about the direction and the magnitude of the rate of increase of intensity at each point on the edge. This filtering method is described in detail in Appendix B, section B1.1.

A7.5.2. High-Pass Filtering

The use of high-pass filters is based on the distribution of information in the frequency domain as computed by the Fourier transform. Edges and other sudden changes in grey level are associated with high frequency components. Thus image sharpening involves "attenuating the low frequency components in the Fourier transform without disturbing high frequency information" [GONZA77], and in effect removing contrast information of the image while emphasizing edges.

The Laplacian operator (fig. A7) is an example of a high-pass filter applied in the spatial

domain. The result of convolving an image with the Laplacian mask is that areas of the

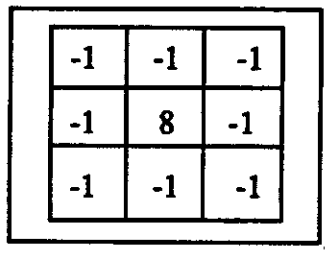


Figure A7. Laplacian Operator.

original image containing edge information will map into a bright and a dark edge adjacent to each other in the filtered image, while low frequency information is mapped into a mid-grey intensity [ROSEN82].

Unsharp masking is a local technique for sharpening an image by subtracting its Laplacian transformation from the original blurred image. This operation emphasizes edges while preserving the grey level information in the non-edge portions of the image.

8. Appendix B: Segmentation Techniques

Segmentation of an image occurs in the sensory processing module of Level 1. The image data is broken up into components or features which classify pixels by distinct categories [JARVIS84]. The features are extracted from filtered or enhanced images and classify pixels in an image based on similarities or differences. The classification of data points produces compressed information; the information reduction process is irreversible [BROWN86]. Two basic approaches to segmentation accomplish boundary (gradient) extraction and surface patch (region) extraction. The goal at this level in the sensory processing system is to operate directly on pixel data to measure important spatial or spectral properties in the image.

B8.1. Boundary Extraction

Methods for extracting boundaries in an image rely on detection of discontinuities in intensity. The grey level at an edge changes abruptly at the border of two adjacent regions. A local edge operator measures this change detects this change over a small spatial extent using a mathematical operation. There are basically three main classes of edge operators: mathematical gradient operators, template matching, or parametric model fitting. These boundary features take the form of either edges or corners. Corner detection will be discussed in the Level 2 Perception Processing document. The following sections provide more detail about these methods.

B8.1.1. Mathematical Gradient Operators

Gradient operators respond strongly to places in an image where the grey level changes rapidly. Digital approximations made to either the first or second partial derivatives respond numerically to intensity changes. The first order partial is a directional derivative which enables calculation of magnitude and direction of the change. The second order partial is not sensitive to direction and also responds to corners as well as edges.

The first order partial derivatives, $\delta f/\delta x$ and $\delta f/\delta y$, measure the rate of change of a function f in perpendicular directions. The direction of the rate of change is a linear function given by:

$$\frac{\delta f}{\delta x'} = \frac{\delta f}{\delta x}\cos\theta + \frac{\delta f}{\delta y}\sin\theta \qquad [1]$$

The direction which has the largest rate of change is :

$$\mathrm{dir}(f) = \arctan \frac{\left(\frac{\delta f}{\delta x}\right)}{\left(\frac{\delta f}{\delta y}\right)}. \qquad [2]$$

and the magnitude of that change is:

$$\nabla f = \sqrt{\left(\frac{\delta f}{\delta x}\right)^2 + \left(\frac{\delta f}{\delta y}\right)^2} \quad [3]$$

These continuous operations can be approximated using discrete difference operations. They measure horizontal and vertical changes in f across a pixel located at (x,y) in the image by:

$$(\Delta_x f)(x,y) \equiv f(x+1, y+1) - f(x,y) \quad [4]$$

$$(\Delta_y f)(x,y) \equiv f(x,y+1) - f(x+1,y). \quad [5]$$

Some of the most historical edge operators are numerical masks that are convolved with the image such as:

0	1
-1	0

-1	0
0	1

(a) Roberts

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

(b) Prewitt

-1	0	1
-2	0	2
-1	0	1

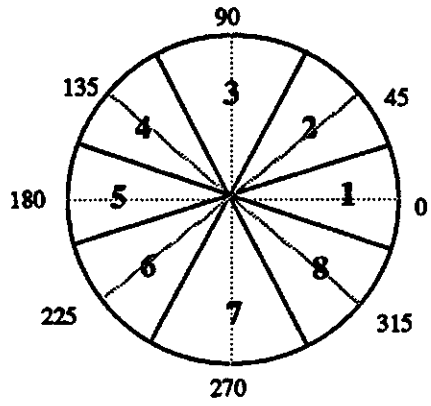
1	2	1
0	0	0
-1	-2	-1

(c) Sobel

Using a 3x3 operator instead of a 2x2 operator enables greater local averaging to reduce noise. The Sobel operator includes a weighted average to combine the pixel values, which increases the response of sharp edges.

Because these gradient operators measure a response across multiple pixels, the edge detection process produces responses on both sides of the edges even if the edge is perfectly sharp. Since edges are usually slightly blurred, the operator generally produces responses with a thickness of several pixels in the gradient direction. In subsequent applications, it is often necessary to have one pixel wide edges, so non-maximum suppression is used to eliminate multiple responses in the gradient direction. Edge responses are quantized into one of

eight directions, such as in the figure below:



Each of these directions implies checking the edge response in a different direction within a local neighborhood to see if its gradient is a maximum. For example, an edge whose gradient direction is 45 degrees is retained if its gradient magnitude is a maximum among itself, its northeast and its southwest neighbors in an 8-connected neighborhood of pixels.

After passing an image through a bandpass filter at multiple resolutions, described by Crowley and Parker [CROWL84], a difference of low-pass transform images are formed. Peaks and rides are detected in the resulting images; a peak corresponds to a local positive maxima or negative minima in a two dimensional 8-connected neighborhood of pixels, and a ridge is similarly a maxima or minima in one dimension.

Canny [CANNY86] presents similar measures for good edge detection. He defines detection and localization criteria for edges and derives mathematical forms for these criteria. In addition, he adds the constraint that the operator must provide a single response across the width of a single edge. Good detection of a noisy step edge correspond to low probabilities of either failing to mark an existing edge point or falsely marking a non-existent edge point. This criterion is met by maximizing the signal to noise ratio:

$$\Sigma = \frac{A}{n_0} \frac{\int_{-\infty}^{0} f(x)dx}{\sqrt{\int_{-\infty}^{+\infty} f^2(x)dx}}$$

[6]

where A is the amplitude of the input step edge. The localization of the marked edge points to their true position is given by:

$$\Lambda = \frac{A}{n_0} \frac{f'(C)}{\sqrt{\int_{-\infty}^{+\infty} f'^2(x)dx}} .$$

[7]

To meet both objectives, the two functions are multiplied together and maximized. The probability of marking multiple edges is reduced by constraining the distance between adjacent

maxima in the response. Combining these constraints, the solution becomes:

$$f(x) = a_1 \exp(\alpha x) \cos(\omega x + \theta_1) + a_2 \exp(-\alpha x) \cos(\omega x + \theta_2) - \frac{\lambda_3}{2} .$$

[8]

When the values of the constants are solved for, the solution can be approximated by the first derivative of a Gaussian function. To expand this solution to two dimensions, a Gaussian is also used to project the direction of two dimensional slope to one dimension. These two operators are convolved together. Since the edges can be approximated by linear segments, highly directional operators at several orientations are used. Varying widths of the operators to cope with varying signal to noise ratios in the image. The results are integrated into a single description.

The one dimensional edge operator described by Canny provides similar results to the zero-crossing of the Laplacian operator described by Marr and Hildreth [MARR75]. The Laplacian operator, shown below:

-1	-1	-1
-1	8	-1
-1	-1	-1

is a digital approximation to the second partial derivative of $\delta^2 f/\delta x^2 + \delta^2 f/\delta y^2$ in the same way that the gradient methods discussed previously are approximations to the first partial derivatives. The Laplacian operator, though, does not provide useful directional information and doubly enhances the noise in an image. The work by Marr and Hildreth advocates filtering an image using four Gaussians which have different bandpass characteristics. The filtered images are then convolved with the Laplacian operator, and places where changes in sign occur correspond to edges in the original image.

B8.1.2. Template Matching

In template matching, an edge pattern is centered on each pixel in an image, and the closeness of their correspondence is measured. Since these templates often represent second differences of step edges, the operators are similar to those difference operators in section 8.1.1. The Prewitt and Sobel operators can be generalized to eight masks corresponding to eight edge orientations. The Kirsch operator is related to the edge gradient by:

$$S(x) = \max [1, \max \sum | f(x_k) - f(x)|]$$

[9]

where $f(x_k)$ are the eight surrounding pixels of x. The corresponding masks are shown be-

low:

5	5	5
-3	0	-3
-3	-3	-3

5	5	-3
5	0	-3
-3	-3	-3

5	-3	-3
5	0	-3
5	-3	-3

-3	-3	-3
5	0	-3
5	5	-3

-3	-3	-3
-3	0	-3
5	5	5

-3	-3	-3
-3	0	5
-3	5	5

-3	-3	5
-3	0	5
-3	-3	5

-3	5	5
-3	0	5
-3	-3	-3

In practice, the operator is sensitive to the magnitude of f(x), so that templates with larger spans offer the advantage of being less sensitive to noise. However, larger templates have difficulty resolving the detail of fine texture. Marr [MARR81] present methods for choosing the appropriate span. Using these ideas, Nevatia and Babu [NEVAT80] use six 5x5 masks that correspond to an ideal step edge at various 30^{o} orientations:

-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100

-100	32	100	100	100
-100	-78	92	100	100
-100	-100	0	100	100
-100	-100	-92	78	100
-100	-100	-100	-32	100

100	100	100	100	100
-32	78	100	100	100
-100	-92	0	92	100
-100	-100	-100	-78	32
-100	-100	-100	-100	-100

100	100	100	100	100
100	100	100	100	100
0	0	0	0	0
-100	-100	-100	-100	-100
-100	-100	-100	-100	-100

100	100	100	100	100
100	100	100	78	-32
100	92	0	-92	-100
32	-78	-100	-100	-100
-100	-100	-100	-100	-100

100	100	100	32	-100
100	100	92	-78	-100
100	100	0	-100	-100
100	78	-92	-100	-100
100	-32	-100	-100	-100

Another template-based method, used by Frei and Chen [FREI77], chooses orthogonal 3x3 masks as a basis for expansion. This expansion yields a space of 3x3 masks with which

local neighborhoods in the image can be compared. Given these templates, an edge response is measured by determining the generalized correlation measure between grey level values in the template and those in the image window. Define the mean, α, and the variance, σ, in the template as:

$$\alpha = \frac{1}{n^2} \sum_{l=-k}^{+k} \sum_{m=-k}^{+k} p_{l,m} \qquad [10]$$

$$\sigma(p) = \frac{1}{n^2} \sum_{l=-k}^{+k} \sum_{m=-k}^{+k} (p_{l,m} - \alpha)^2 \qquad [11]$$

(where $p_{l,m}$ is a pixel in the template at l, m and the template size is odd or n = 2k+1) and the mean, $\beta_{i,j}$, and the variance, $\sigma_{i,j}$, of an n x n window as:

$$\alpha_{i,j} = \frac{1}{n^2} \sum_{l=-k}^{+k} \sum_{m=-k}^{+k} q_{i+l,j+m} \qquad [12]$$

$$\sigma(q)_{i,j} = \frac{1}{n^2} \sum_{l=-k}^{+k} \sum_{m=-k}^{+k} (q_{i+l,j+m} - \beta_{i,j})^2 . \qquad [13]$$

Then the generalized correlation measure between the image and the template at pixel i,j is:

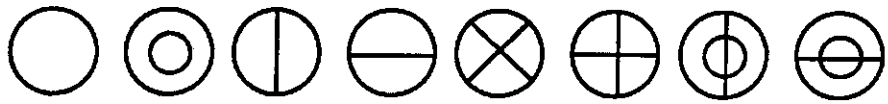
$$\delta_{i,j} = \frac{\frac{1}{n^2} \sum_{l=-k}^{+k} \sum_{m=-k}^{+k} (p_{l,m} - \alpha)(q_{i+l,j+m} - \beta_{i,j})}{\sigma(p)\, \sigma(q)_{i,j}} . \qquad [14]$$

Close correlation between the template and the window indicates an edge of the type depicted in the template.

B8.1.3. Parametric Edge Modeling

Parametric edge models provide more information than the magnitude and direction of the gradient as discussed for previous edge detection methods. This approach involves expanding the image and the step edge functions in terms of a set of orthogonal basis functions. Hueckel [HUECK71] proposed analyzing the frequency behavior by observing the zero-

crossings of the following eight basis functions defined on a disk:



By minimizing the sum of the squared error between the image and the edge model in the circular neighborhood, measures of the slope of the step edge and the average intensity values on either side of the step edge can be obtained. Other uses of parametric edge models include models by Nevatia [NEVAT77] who used a subset of Hueckel's basis functions and O'Gorman [O'GORM78] and Mero and Vassy [MERO75] whose bases were defined on squares. Parametric edge models determine more about an edge's structure, but they are also more computationally expensive than other methods of edge detection.

B8.2. Region Extraction

The class of segmentation methods which label pixels according to similarities is termed region or surface patch extraction. These methods can be looked at as the opposite of edge extraction. Region based methods group pixels which share some intensity based property and which provide spatial continuity. These variations can occur on a large scale, where the classification is based on shading, or on a small scale, where the differences are based on texture.

B8.2.1. Intensity and Color

When viewed objects are uniform in color or intensity, labeling pixels according to these characteristics is a natural way to segment the image. The classification of images can be accomplished by labeling pixels based on a number of criteria. One of approach labels pixels by comparing them to a threshold value and another method of labelling is based on comparing the connectedness of adjacent pixels. Each of these approaches are described in more detail.

Weszka [WESZK78] describes numerous global, local, and dynamic methods to choose threshold values. Global threshold values separate peaks of an image's histogram into two or more categories. However as grey level subpopulations become less distinct, reliable threshold selection becomes more difficult. Local threshold techniques label each pixel based on the properties of its surrounding neighbors. These methods are susceptible to minor variations in intensity but have the advantage of being applied in parallel. A dynamic method, designed to operate on low quality images, uses the statistical variance in a local neighborhood to select a threshold. The methods described can be used to perform binary or multilevel thresholding using grey levels or multispectral images.

B8.2.2. Texture

Textured patterns are regions of uniform brightness that have many internal edges. Consequently, methods that apply to smooth region extraction (Appendix B8.2.1) cannot be used to classify pixels in a textured region [ROSEN88]. Texutred regions can be segmented by

information determined from individual pixels, local features, or larger regions. Measurements can be made on pixels or local features using statistical relationships, which characterize the distributions and relations of pixels or regions. The structural methods describe primitives and the patterns used to generate a texture. Both of these classes of methods are described in more detail.

Uniformly spaced elements of similar shape in an image produce texture. Both the autocorrelation function of an image and its Fourier transform measure the spatial frequency that characterizes a pattern. Autocorrelation indicates how each pixel in an image influences surrounding pixels. It is a linear model that describes the frequency of light transmitted when the intensity of a pixel is compared to its neighbor. The maxima and minima of this two dimensional function indicate the size and separation of the texture primitives that compose the image. The coarseness of the pattern is indicated by the slope of the central peak in any given window of the image. The periodicity of the peaks also characterize the frequency of the texture pattern. The auotocorrelation function is given by:

$$Y(\Delta_x, \Delta_y) = \frac{\sum_{i,j} f(i, j)\, f(i + \Delta_x, j + \Delta_y)}{\sum_{i,j} [f(i, j)]^2}$$

[15]

where i and j can lie within a window and Δ_x and Δ_y represent the shift between a pixel and its compared neighbor. The autocorrelation function does not provide good discrimination for natural textures, since coarseness tends to not be distinct.

B8.3. Optical Flow

Optical flow is defined as the motion of object points across an image resulting from the relative motion between a camera and objects in the scene. It is calculated from local temporal and spatial variations in sequences of grey level images. The optical flow, or instantaneous velocity field, assigns a two dimensional "retinal velocity" to every point in the visual field. The results of this measurement are used as input for higher level methods which compute camera motion, depth maps, and surface normals.

There are two general classes of methods for extracting optical flow from sequences of images: gradient based methods and correlation based methods [HONG89]. The first method uses the spatial and temporal derivatives of pixel brightness; the second tracks features in small regions of images over time. Level 1 processing includes the gradient method of optical flow extraction. The assumption of gradient based techniques is that pixel intensity in an image is constant over time, and thus any change in intensity at a point in the image is due to camera motion The optical flow is defined as (u,v) :

$$u = (1/z)(x_i v_z - v_x f) + \alpha (x_i\ y_i)/f + \beta (x_i)^2/f + \beta f - \gamma y_i$$

[16]

$$v = (1/z)(y_i v_z - v_y f) + \beta (x_i y_i)/f + \alpha (y_i)^2 /f + \alpha f - \gamma x_i$$

[17]

where $(x_i,y_{i)}$ is the position of a point in the image, f is the focal length of the camera, $V = (v_x,v_y,v_z)$ is the translation velocity of the camera, and α,β,γ represent the rotational velocity of the camera about the X, Y, and Z axes respectively [HONG89].

The Horn and Schunck optical flow algorithm [HORN81] uses the ratio of spatial and temporal image derivatives described above over two frames of image sequences to measure pixel velocity normal to the gradient direction. Analysis of the results of their method by Kearney, et al. [KEAR87] indicate that large errors occur where the image is highly textured or where motion boundaries exist due to depth discontinuities. In an effort to overcome these shortcomings, methods have been developed which use a large number of frames sampled closely together in time [DUNC88, WAXM88]. Errors are reduced by extracting the optical flow from the second derivative of the Gaussian temporally smoothed image [MARR81].

B8.4. Evaluation

Errors are often made when segmenting pixels based on local information as described in the previous boundary or region methods. One way to improve the reliability of labelling is by adjusting the measurements made based on measurements of adjacent pixels. This method detects and corrects local inconsistencies in the pixel labels and is called relaxation. [DAVIS80]

There are two types of relaxation methods: discrete and fuzzy. A discrete method checks adjacent label values and may adjust a label's value based on this comparison. Fuzzy labelling associates a likelihood value with each label and uses this to determine the appropriate value.

A relaxation process is specified by two things: a neighborhood model and an interaction model. The neighborhood model specifies which pairs of pixels contribute to the relaxation process. The choice of which pixels communicate depends on the goal of segmentation. A directional neighborhood model may be specified for edge detection, while the positional information may not be important in a region extraction method. The interaction model determines the criteria for changing a pixel's label. Interaction models need to represent the relationships between labels and the mechanism by which labels are modified. The interactions can be represented by relational knowledge or by logical statements.

T

W

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NIST/TN-1260	2. Performing Organ. Report No.	3. Publication Date June 1989

4. TITLE AND SUBTITLE

Visual Perception Processing in a Hierarchical Control System: Level I

5. AUTHOR(S)

Karen Chaconas, Marilyn Nashman

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
(formerly NATIONAL BUREAU OF STANDARDS)
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as Item #6

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This document describes the interfaces and functionality of the first level of the visual perception branch of a realtime hierarchical manipulator control system. It includes a description of the scope of the processing performed and the outputs generated. It defines the interfaces and the information exchanged between the modules at this level, as well as interfaces to a camera, a human operator, and to higher levels of the system. The reader should be familiar with ICG Document #001, NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM).

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

camera processing; execution module; hierarchical control system; image processing; job assignment module; perception; planner module; real-time; sensory processing; task decomposition; temporal-spatial processing; world model

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

49

15. Price