Three-Dimensional Position Determination from Motion

Marilyn Nashman and Karen Chaconas

National Institute of Standards and Technology, Gaithersburg, MD 20899

ABSTRACT

The analysis of sequences of images over time provides a means of extracting meaningful information which is used to compute and track the three-dimensional position of a moving object. This paper describes an application in which sensory feedback based on time-varying camera images is used to provide position information to a manipulator control system. The system operates in a real-time environment and provides updated information at a rate which permits intelligent trajectory planning by the control system.

1. Introduction

The NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM)[1] describes a hierarchical framework to control complex robot systems. It has been partially implemented in various versions of the Real-time Control System (RCS) at the National Institute of Standards and Technology (NIST) including the Automated Manufacturing Research Facility (AMRF), the Army Field Material-handling Robot (FMR), the NBS/DARPA Multiple Autonomous Undersea Vehicle project (MAUV), and the Army TEAM (Technology Enhancement for Autonomous Vehicles) project. The hierarchy decomposes plans spatially and temporally to meet system objectives, monitors the environment with system sensors, and maintains the status of system variables.

The control system is composed of three parallel systems, task decompostion, world modelling, and sensory processing, that cooperate to perform telerobot control. The task decomposition system breaks 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 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 support processes in each of the three systems of the control system. 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 1. 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.[3] The manipulator branch of the tree supports processes which are responsible for planning and executing manipulator trajectories.[4,10] The two branches decompose tasks independently and communicate via the global data system.

The world modeling support modules communicate asynchronously with the task decomposition and sensory processing systems[7]. Information 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 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

Commercial equipment and materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the materials or equipment identified are necessarily the best for the purpose.

Proc. of SPIE Conference on Advances in Intelligent Systems, Boston, MA, November 1990.

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.

In this paper we discuss an initial task performed in our laboratory involving an implementation of the lowest two levels of the NASREM control hierarchy. Our telerobot system consists of a manipulator, a gripper, and a camera. The demonstration successfully integrates the sensory processing, world modeling and task decomposition modules, thus closing the sensory feedback loop. The discussion in this paper focuses on the algorithms that execute within the modules highlighted in figure 1. In the following sections we present an overview of the system in our lab. We introduce the role of the operator interface for initializing the world model. We then proceed to describe the algorithms that we implemented.

2. System Description

In the experiment, a ping-pong ball is released at the top of an inclined board which contains randomly placed pegs, as shown in figure 2. As the ball rolls down the board, it is tracked by the single-camera vision system and then caught by the manipulator at the bottom of the board. Our manipulator is a Robotics Research seven degree-of-freedom arm. The vision processing is performed by a real-time image processing machine, the Pipelined Image Processing Engine (PIPE)[2]. A VME interface board is used to map memory locations between PIPE and the host system, a 68020 processor. The software is written in Ada.

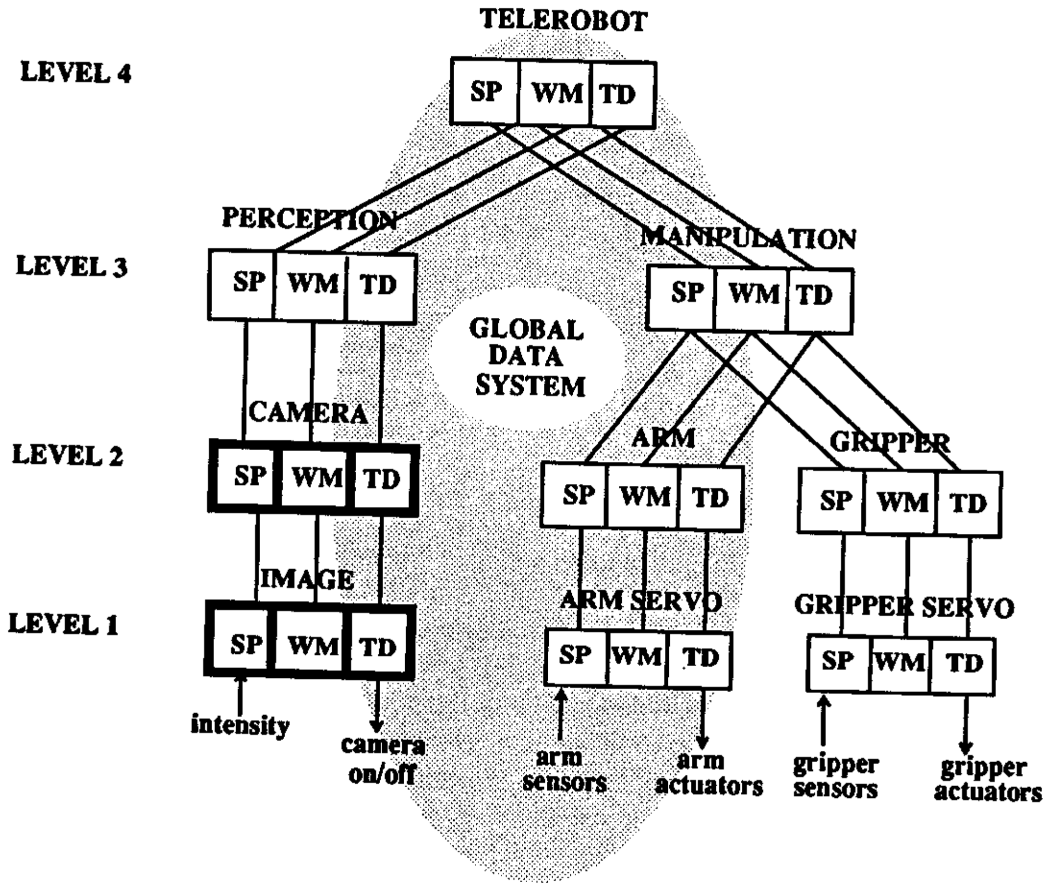


Figure 1. A NASREM Subsystem

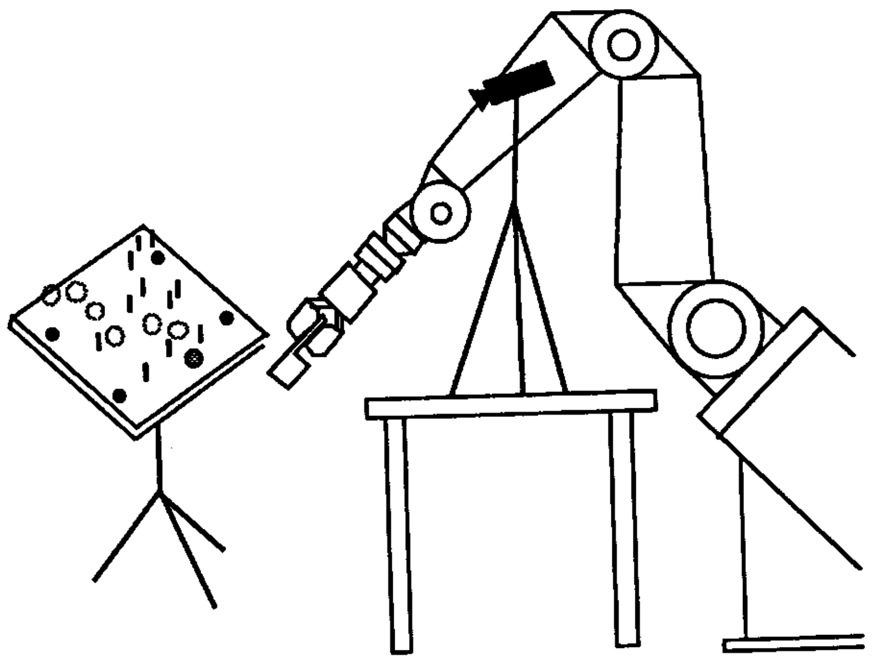


Figure 2. Ball catching task setup

This paper details the algorithms used by the sensory processing and world modeling modules to provide feedback to the manipulation modules. The NASREM implementation is based on the concept of cyclically executing modules which serve as the computational units for the architecture[6]. After initialization, all computations are performed by cyclically executing processes which communicate via global read-write interfaces. Each unit acts as a process which reads inputs, performs computations, and then writes output. Such a process always reads and executes on the most current data; it does not wait for new data to arrive since reliable cyclic execution requires that a module be able to read or write data with minimal delay. Reading and writing involves the transfer of data between local buffers and buffers in global memory. System software has been written to prevent data corruption during these transfers.

3. Perception Initialization

The perception processing modules perform initialization of the board's position with respect to the camera. The camera is arbitrarily positioned at the start of the task; the only constraint on its placement is that the entire face of the board must be in the field of view. World modeling then must record the camera's location in the world. The plane of the board is denoted by four black circles with a known configuration. We determine the transformation from the board to the camera using a four coplanar point algorithm.[11] In addition to the camera-to-board transformation, the board's pose in world coordinates must also be known. The position and orientation of the board is identified in world coordinates by placing the manipulator's end effector at three locations on the board and calculating the plane which contains all three points. Now, x, y, z locations produced with respect to the camera's frame of reference can be transformed to the board's reference frame and from there to a world coordinate system.

In our implementation, the operator interface facilitates initialization of the system in the absence of a complete world model. The operator interface provides a means by which human operators can observe, supervise, and directly control the system.[1] The operator interface is used to locate the position of the four circles in the camera image. By moving the keyboard mouse, the operator is able to isolate each of the four black circles within a "window of interest."

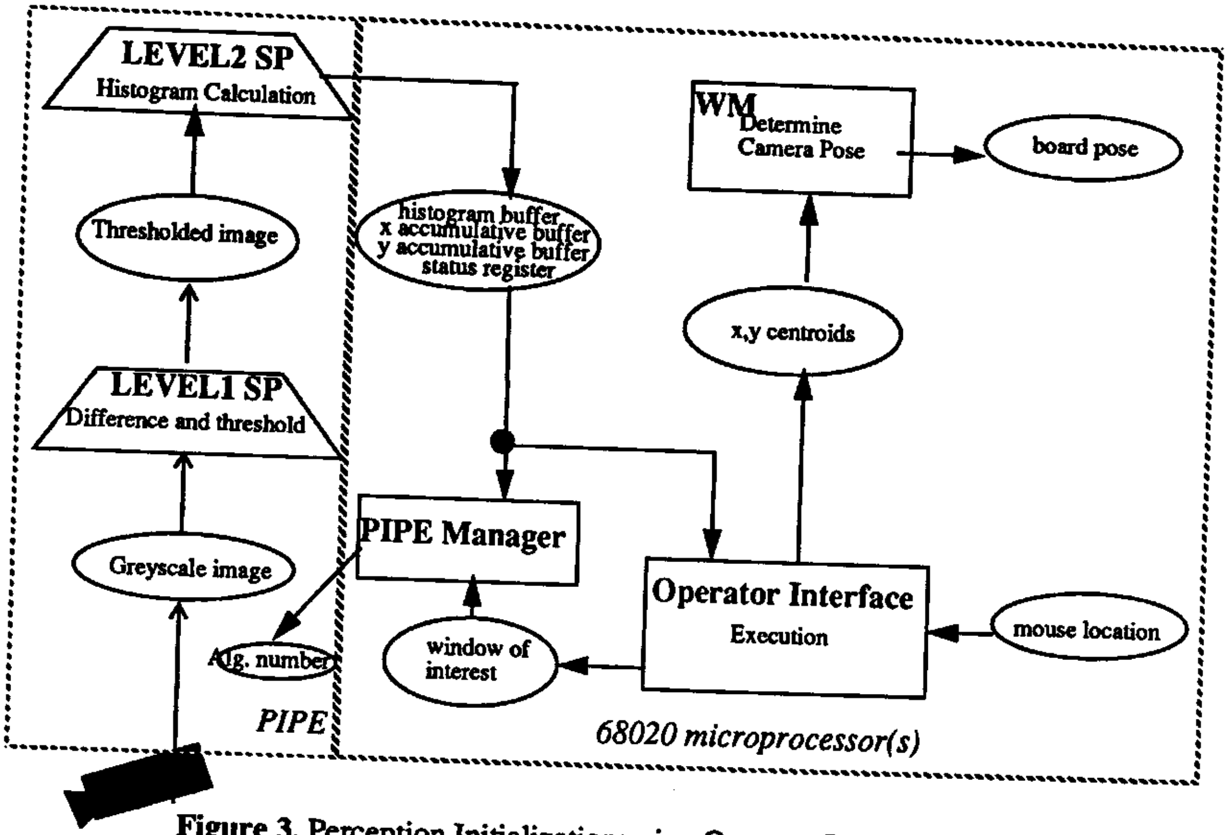


Figure 3. Perception Initialization using Operator Interface

Once the centroids of the four circles are found, world modeling computes the transformation from the board to the camera.

The operator interface performs the initialization as shown in figure 3. In this figure, the processes that are pertinent for initialization are shown using rectangles. The ovals represent the interfaces between processes and detail the data that is passed. The trapezoids show the pipelined processes that perform image processing. The operator moves a keyboard mouse which causes a cursor on his screen to move. The operator interface process reads the position of the cursor on the screen in a scale that corresponds to image coordinates. The mouse position information is sent to PIPE where it is used to determine the location of a mask, or "window of interest", in the image. The mask is centered on the position of one of the four circles in the camera image of the board. The purpose of the mask is to blank out all information other than that in the area of interest.

A camera view is digitized by PIPE, and the input image is thresholded to produce a binary (black/white) view of the scene. The threshold value is determined a priori based on knowledge of the scene. The thresholded image passes through PIPE's Iconic to Symbolic Mapper (ISMAP) which computes frequency and accumulative histogram information. The frequency histogram produces the total number of occurrences in the thresholded image of each value from 0 to 255. The first moments are computed using the histogram accumulation mode to sum the values of the x and y addresses in the binary image corresponding to non-zero grey levels. Since the image is binary and only the non-zero values are of interest, only one location is read from each histogram. The centroids are computed by dividing the frequency histogram by the accumulative histogram value.

The number of pixels contributing to the centroid computation can change. The intensity value of a pixel can fluctuate due to sampling noise or variations in the ambient lighting. Any change in measured grayscale of pixels whose intensity is close to the threshold value used can cause the resulting value to vary. In order to obtain as accurate a centroid calculation as possible, 10 histogram readings are read by the operator interface process. The grayscale histogram reading with the largest area and the corresponding accumulative histograms are used for the centroid calculation.

The centroid of each of the four circles is computed in the same manner and is stored in global memory for use by world modeling. All four centroid positions are used by the Level 2 world model process to compute the camera's pose with respect to the board. Once the pose is known, two-dimensional image positions corresponding to points on the board can be transformed to three-dimensional positions which are used by the manipulation processes.

4. Vision Processing

The output produced by the sensory processing system is a feature that will be used to identify the three-dimensional location of the ball at each sampled point in time. There are several criterion that the sensory processing algorithm must meet. Foremost among these is that the algorithm run in real-time and produce feedback for the manipulator at a meaningful rate. The Primitive module for the manipulator updates the commands it sends to the Servo module at 5ms intervals.[10] Our algorithms are implemented as parallel programs running on two tightly-coupled multiprocessors, an Aspex PIPE for low-level processing and a 68020 microprocessor for high-level processing. PIPE is a specialized image processing machine designed to operate on 64K bytes of image data every sixtieth of a second. In order to minimize the bottleneck effect of the image processing algorithms on the manipulator system, it is necessary to use algorithms that optimize update rates as opposed to throughput rates. (Throughput rate is defined as the time interval between digitizing an image and outputting results dependent on that image. Update rate is defined as the time interval between successive outputs.) Since PIPE is a pipelined processor, this criterion can be met. The world model compensates for throughput delays by providing predictions of the current position based on past readings. The output rate must be spaced at a constant interval to improve the validity of the world model predictions.

A second criterion that we consider in choosing an algorithm is that the selected algorithm be robust in an unstructured environment. Our laboratory contains the robot arm, the PIPE machine, 68020 microprocessor racks, and several Sun workstations. We have standard overhead fluorescent lighting, and we do not use special backdrops to separate the camera field-of-view from the remainder of the laboratory. In addition, the ball that is used in this demonstration is similar in intensity to the planar surface upon which it is rolled.

A third criterion imposed on our choice of algorithm is that it take advantage of the high temporal sampling rate available with PIPE. We prefer an algorithm that produces approximate output samples frequently as opposed to an algorithm that produces infrequent but very accurate outputs. The world model module filters and smooths the outputs produced by the sensory processing algorithm and thus the system accommodates for faster but less accurate algorithms.

Several algorithms were written and tested in an effort to meet the above conditions. The goal of all the algorithms was to segment the image and extract the ball from the scene. At that point, information about the ball could be used to extract its two dimensional centroid. Traditional connected component algorithms were rejected because of the difficulty in extracting the ball from such a complicated scene. Also, a connected component algorithm cannot currently be performed on PIPE and therefore processing would have to be done on the 68020 boards. This restriction conflicts with our real-time requirements. (Aspex Corporation is currently working on adding a connected component capability to PIPE.) Simple thresholding of the input scene is not feasible because of the experimental setup; we are using a white ping-pong ball on a white planar surface. An additional restriction on the choice of algorithm is a result of the ISMAP design. The centroid calculation is based only on grey level information, not on whether those grey levels are contained in only one connected blob. Thus it is necessary for the sensory processing algorithm to encode only the moving ball at a unique grey level. Any other objects (or parts of objects) that are encoded with that same grey level will result in corrupted centroid information.

For these reasons, we decided to use an algorithm that segments the image based on motion rather than working with static images. By considering a dense stream of images over time, we can extract the moving ball from the static background. Among the algorithms we considered were the optical flow computation proposed by Horn and Schunk[7], a differencing of an initial static image from a subsequent stream of images, and an algorithm that detects motion as a change of intensity between consecutive images. These methods are briefly described below, and the rationale for our choice of the third method is explained.

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 temporal and spatial variations in sequences of grey level images. The optical flow is represented as a two-dimensional image velocity at every point in the visual field. The algorithm proposed by Horn and Schunk[7] uses a gradient-based method for extracting the optical flow from a sequence of images. It computes the ratio of temporal and spatial image derivatives over two frames of an image sequence to measure pixel velocity normal to the gradient direction. These calculations are easily implemented on PIPE since they are based on local operations. However, the results obtained from this algorithm were not suitable for our application. The velocity of the moving ball varied at different points in its path, and when the motion was greater than one pixel between frames, the velocity computed at that point was lost. Efforts to overcome this problem result in a longer program and thus interfere with the requirement for real-time processing.

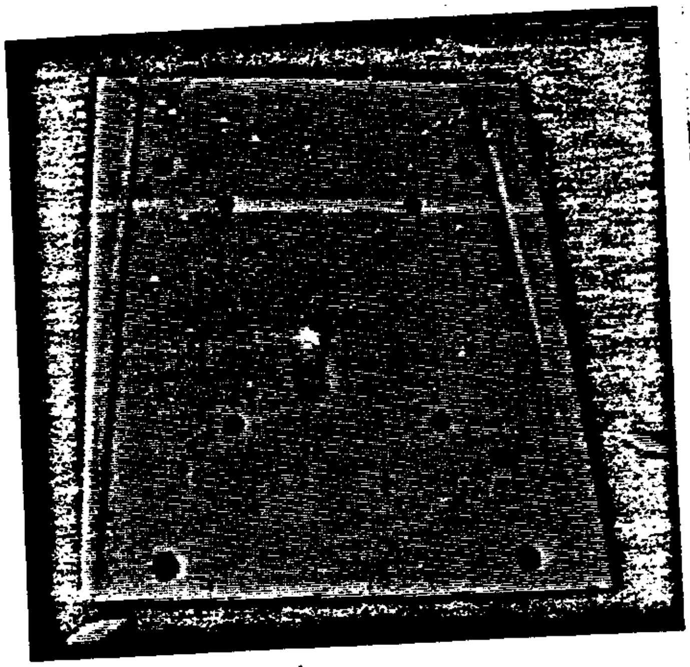
The second algorithm implemented is based on extracting information due to changing intensity values between image frames. In our experimental scene, the only moving object visible to the camera is the ball moving down the inclined plane. We expected to extract an image of only the ball by differencing an initial static image without the ball present from a subsequent stream of images where the ball is in motion. This algorithm, too, is easily implemented on PIPE. The differenced image is converted to a binary image and information about the ball is extracted from ISMAP. The problem encountered with this algorithm was caused by slight differences in the stream of images caused by flickering fluorescent lights or by vibrations of the inclined plane or camera caused by closing doors. The differencing scheme was extremely sensitive to these variations, and as a result, image noise often appeared in the final image corrupting the computed centroid.

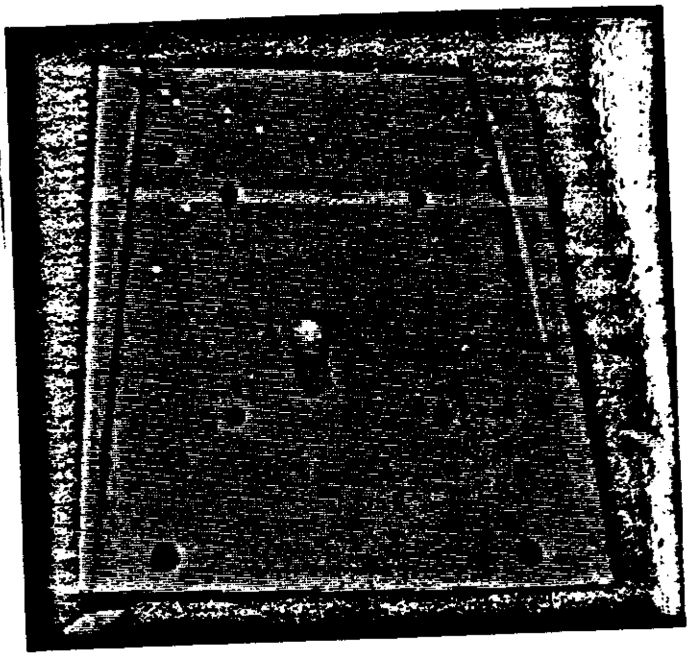
The vision algorithm finally chosen for the experiment determines the location of the ball using an algorithm based on the difference of sequential images. The scene is segmented due to changing intensity values between successive images. PIPE acquires images at a video rate of 1/60 second, but in order to ensure that consecutive images have the same camera field sampling, images are input every other PIPE cycle. This limitation will be removed in future implementations due to a modification of the camera hardware which will permit us to digitize images in a non-interlaced mode. An example of two images taken of the ball in motion and separated by 1/30 second are shown in figures 4a and 4b. Incoming images are smoothed using a Gaussian convolution to diminish the effects of spurious noise in the image.[3] Two consecutive smoothed images are subtracted from each other in order to detect any change in intensity due to motion between the frames. All non-moving features in the field of view "disappear" in this difference image since the grayscale value of a pixel in the second frame is being subtracted from the identical grayscale value in the first frame. The result of the difference image is thresholded to produce a binary image in which white areas refer to points where intensity has changed due to motion and black areas refer to stationary intensity points. This result is shown in figure 4c. The thresholded difference image is passed through ISMAP on PIPE and the relevant histogram and accumulative histogram values are computed. The 68020 processor polls ISMAP to determine when the histogram information is current and available. At the appropriate time, it reads the histogram values and passes them to the Level 2 sensory processing module which completes the centroid computation. Figure 5 describes the algorithm as it is programmed on PIPE. The algorithm operates on two sets of difference images in parallel. The implementation of this algorithm has a throughput time of 100.2 ms and an update rate of 66.8 ms.

The remaining NASREM perception processes run on a 68020 board. This includes the Level 1 and Level 2 modules, the PIPE manager which acts as an interface between PIPE and Level 1, and the Level 2 world model process. Figure 6 describes the decomposition of the perception branch of the NASREM hierarchy for this demonstration.

5. World Modeling

World modeling performs several functions in support of the calculation of a three-dimensional position.[8] The image differencing algorithm described above is useful for tracking a ball moving on a planar surface when the ball is the only object which moves in the field-of-view. In that case, the centroid successfully approximates the location of the ball. However, if the integrity of the scene cannot be guaranteed, world modeling must intervene. One method of isolating the motion of the ball is for world modeling to provide a window of interest around the expected location of


4a


4b

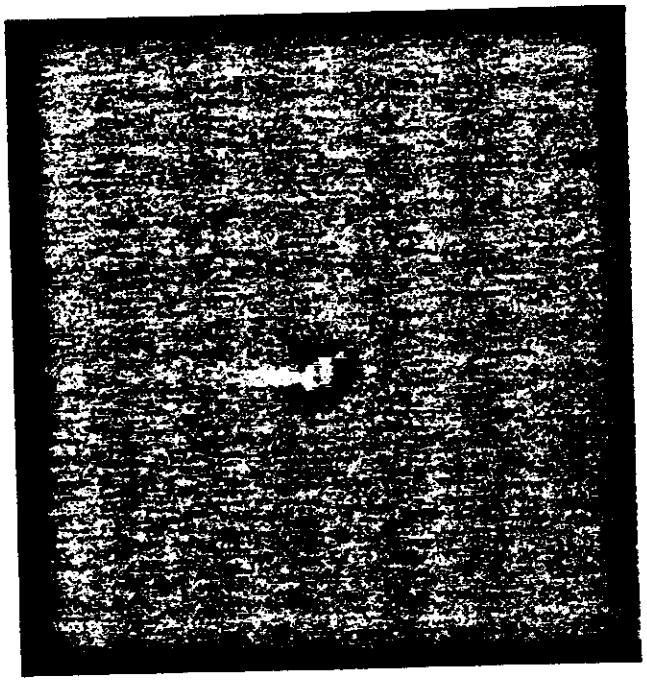

4c

Figure 4. Intermediate Results of Sensory Processing Algorithm

(a) Input image at time t=n (b) Input image at time t=n+1/30 (c) Result of motion detection

time →

Image Set A	input image		input image	subtract & threshold accum x histogram	accum y histogram	graylevel histogram		
Image Set B	accum y histogram	graylevel histogram			input image		input image	subtract & threshold accum x histogram

Figure 5. PIPE Implementation Optimized for Update Rate

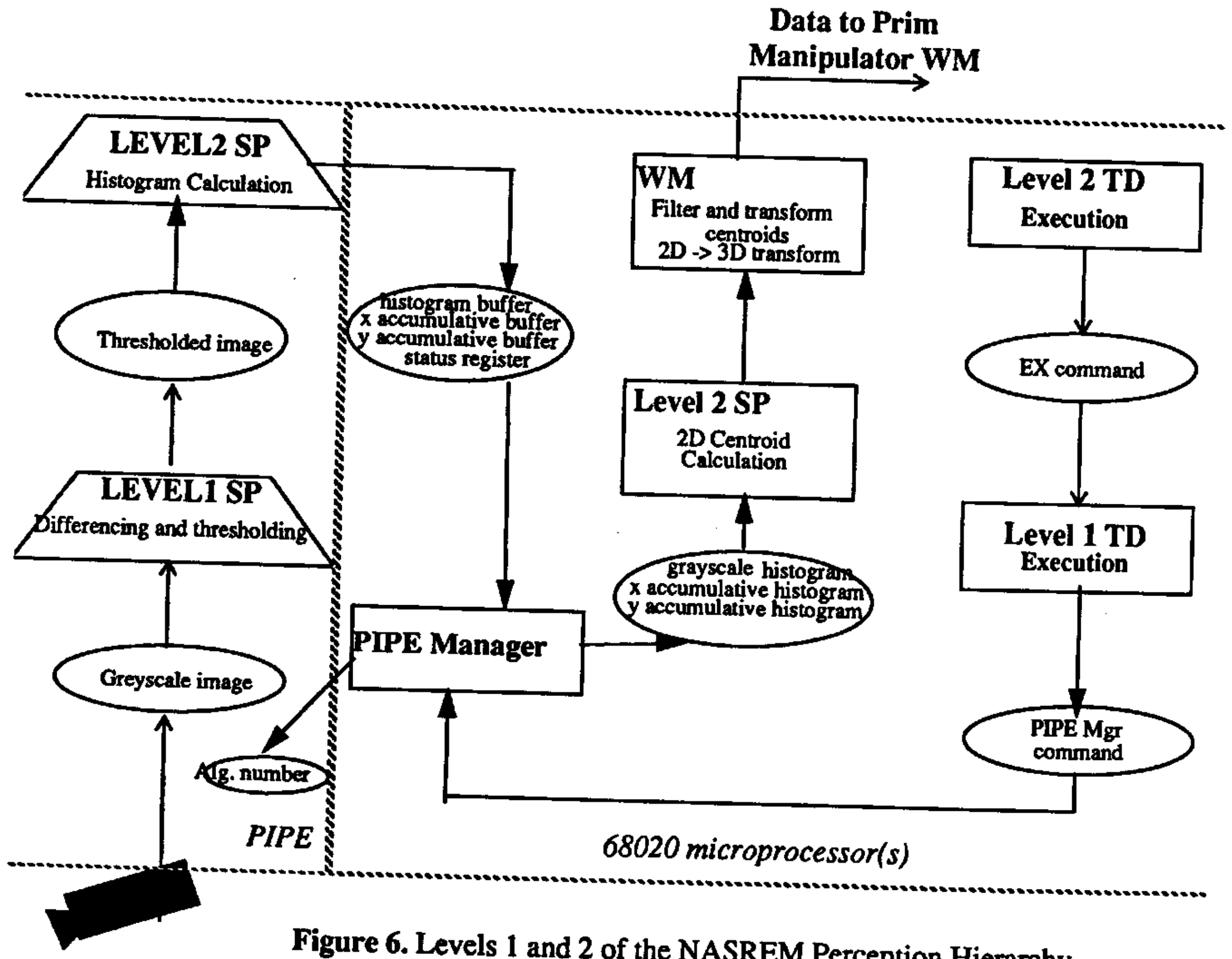


Figure 6. Levels 1 and 2 of the NASREM Perception Hierarchy

the ball in each image. The window locations would be based upon previous reading(s) and knowledge about the movement of the ball between sampling instants. An alternate method, the one which we employ currently, is to maintain a history of centroids which world modeling uses to filter out spurious readings. In our current implementation, world modeling performs all filtering after transforming the image centroid to 3^D space.

The centroid value, as supplied by sensory processing, is a 2^D centroid in image space; the manipulator Level 2 module (Prim) requires a 3^D position with respect to a known reference frame. World modeling performs several computations to transform the 2^D centroid data to a value usable by Prim. First the location of the ball's centroid in the image is represented as a 3^D point on the camera's image plane. The ball's position on the inclined board is computed by projecting a ray from the camera's optical center through the point on the image plane. The intersection of the ray with the inclined board approximates the ball's location in 3^D space.

Specifically, first the point representing the origin of the camera's optical axis and the point giving the ball's position on the image plane are represented with respect to the board's frame of reference. They are transformed to the board's frame of reference via the transform computed during initialization using the four coplanar point algorithm. Then the equation of the line containing the two points is given by:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1} \qquad (1)$$

where (x_1, y_1, z_1) is the ball position and (x_2, y_2, z_2) is the camera position. The equation of the plane of the board can be represented by the general plane equation:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1} = t \qquad (2)$$

where the variable *t* parameterizes the plane. The line through the two points (the origin of the camera and the ball's point on the image plane) intersects the plane of the board at z=0. Actually, a more precise value for z at the point of intersection is (z = half the diameter of the ball), not (z=0). The non-zero value for z reflects the volume, or depth, of the ball; the value generated by the sensory processing algorithm, representing the centroid of the moving ball, is affected by the ball's depth. Using the known z location of the ball, we solve equation (2) for *t* and then determine the x, y location of the ball.

After determining the 3^D location of the ball, world modeling filters out spurious or invalid data points corresponding to locations beyond the boundaries of the inclined board[9]. World modeling also filters out locations which are displaced from the previous reading by more than a predefined limit. We set the limit roughly based upon the physical setup (e.g., incline angle of the board) and the time delay between samples. World modeling transforms the point to the world coordinate system using the transform between the board and the base of the robot manipulator (${}^{board}T_{world}$) that is stored during initialization. The actual position of the ball, as computed from the image, is of little use to Prim. In order to catch the ball, the manipulator must move toward predicted locations of the ball. The manipulator Prim algorithm for generating manipulator goal states based upon the ball position and velocity is discussed by Fiala and Wavering [6].

6. Conclusion

This paper has described a real-time motion-detection algorithm used to determine three-dimensional position. The algorithm is implemented within the perception hierarchy of the NASREM control system. We have described the sensory processing requirements, the image processing algorithm chosen, and the algorithms used by the world model to initialize the system and to produce three-dimensional feature points. The system provides real time position data quickly enough for a manipulator to track and catch a ball randomly rolling down an inclined board.

7. References

[1] J. S. Albus, H. G. McCain, R. Lumia.,"NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", NIST Technical Note 1235, Gaithersburg, MD, July, 1987.

[2] Aspex, Inc., "PIPE--An Introduction to the PIPE System", New York, 1987.

[3] K. Chaconas, M. Nashman, "Visual Perception Processing in a Hierarchical Control System", NIST Technical Note 1260, Gaithersburg, MD, March, 1989.

[4] J. Fiala, "Manipulator Servo Level Task Decomposition", NIST Technical Note 1255, Gaithersburg, MD, October, 1988.

[5] J. Fiala, "Note on NASREM Implementation," NIST Internal Report 89-4215, Gaithersburg, MD, December, 1989.

[6] J. Fiala, A. Wavering, "Implementation of a Jacobian Transpose Algorithm," NIST Internal Report 90-4286, Gaithersburg, MD, April, 1990.

[7] B. K. P. Horn, B. G. Schunk, "Determining Optical Flow", Artificial Intelligence, Vol. 17, pp. 185-203, 1981.

[8] L. Kelmar, "Manipulator Primitive Level World Modeling," NIST Technical Note 1273, Gaithersburg, MD, October, 1989.

[9] L. Kelmar, R.Lumia, "World Modeling for Sensory Interactive Trajectory Generation," to be published at the Third International Symposium on Robotics and Manufacturing (ISRAM), Vancouver, B.C., July, 1990.

[10] A. Wavering, "Manipulator Primitive Level Task Decomposition", NIST Technical Note 1256, Gaithersburg, MD, October, 1988.

[11] P. S. Yeh, S. Barash, E. Wysocki, "A Vision System for Safe Robot Operation," Proceedings of the 1988 IEEE Int'l Conference on Robotics and Automation, pp. 1461-1465, April 24-29, 1988.