ORNL/TM-11991
CESAR-92/01

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Terrain Following of Arbitrary Surfaces Using a High Intensity LED Proximity Sensor

J. E. Baker

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

OR ORNL/TM--11991

Engineering Physics and Mathematics Division DE92 007161

TERRAIN FOLLOWING OF ARBITRARY SURFACES USING A HIGH INTENSITY LED PROXIMITY SENSOR

J. E. Baker

DATE PUBLISHED — January 1992

Research sponsored by the
Office of Technology Development
U.S. Department of Energy

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE–AC05--84OR21400

CONTENTS

LIST OF FIGURES

ABSTRACT

Many robotic operations, e.g., mapping, scanning, feature following, etc., require accurate surface following of arbitrary targets. This paper presents a versatile surface following and mapping system designed to promote hardware, software and application independence, modular development, and upward expandability. These goals are met by: a full, *a priori* specification of the hardware and software interfaces; a modular system architecture; and a hierarchical surface-data analysis method, permitting application specific tuning at each conceptual level of topological abstraction. This surface following system was fully designed and implemented independently of any specific robotic host, then successfully integrated with and demonstrated on a completely *a priori* unknown, real-time robotic system.

1. INTRODUCTION

1.1 APPLICATION

Many robotic operations require accurate surface following of arbitrary targets. Such operations include scanning, terrain mapping/surveying, feature following, target recognition, and target monitoring. Furthermore, surface following in a background mode can provide secondary benefit in object avoidance, relative motion reduction, and increased "tool" effectiveness.

A surface following system provides demands which, when heeded, modify the robotic control commands to maintain a user specified distance between the end effector and the target. This ability requires the surface following system to predict the next target-offset distance based on the end effector's current velocity vector, the extrapolation of recent target points, and a precise characterization of: the sensor system, the input's accuracy, and the surface's inherent roughness.

To be generally applicable, a surface following system must minimize its assumptions about the target and yet to be worthwhile, it must be attuned to the target's particular surface character, i.e., the global applicability of local changes in slope. Hence, general applicability places tremendous demands on the surface following system, requiring both a reliable non-contact proximity sensor system, i.e., one unaffected by surface texture, color, orientation, size, composition, etc., and highly sophisticated data interpretation capabilities.

1.2 DESIGN MOTIVATIONS

The design of this surface following system is motivated by the following four considerations:

1) Hardware Independence – Since surface following is required for a variety of applications, the hardware system is designed as a self-contained, "plug-in" module. The hardware system includes its own sensory system, a dedicated CPU, memory space for all its local and shared variables, and all necessary, specialized interconnections.

2) Software Independence – Similarly, the software system is designed to maximize its compatibility with the application's software environment by employing its own CPU and restricting communication to predefined use of absolute addressed, shared memory locations. These absolute addresses are determined at run-time based on user defined parameter settings indicating the hardware configuration.

3) Application Independence – To enable the system to successfully follow an arbitrary surface: a robust sensor system is employed (capable of detecting most targets); the system is designed to operate in real-time, i.e., greater than the typical control system's loop rate (> 150 Hz); and the forecasting algorithms are designed to accommodate a wide variety of surface textures and modalities.

4) Upward Expandability – The system is designed to readily accommodate future technical enhancements by its modular design, shared memory communication, and hierarchical organization.

2 *INTRODUCTION*

The following section describes the hardware chosen for the system's initial implementation. Section 3 details the specific software algorithms used for surface mapping and terrain forecasting and the system's overall software architecture, while Section 4 presents general conclusions.

2. HARDWARE

2.1 SURFACE FOLLOWING SYSTEM

Conceptually, this surface following system consists of four hardware modules: a sensor system, an A/D converter, a CPU and externally accessible memory. These modules were configured and implemented as shown in Fig. 2.1.

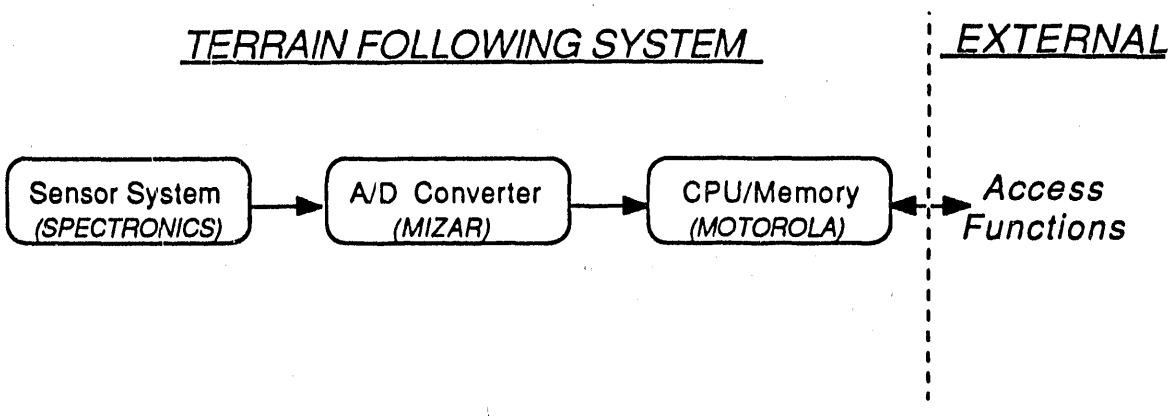

Fig. 2.1. Hardware configuration—development system.

The sensor system is a separate module responsible for measuring the absolute distance between the target's surface and the sensor's face, regardless of ambient noise (light/sound conditions) or surface character (color, texture, sheen, conductivity, etc.). Due to the unconstrained character of the environment, the sensor system must be extremely robust and accurate. After a careful survey of the available sensor technologies and systems, high intensity LED triangulation was determined to be the best sensor technology for this application. A sample sensor system was obtained, the Spectronics Model 204-4,[1] and its performance empirically analyzed.[2] This system was found to be extremely robust with respect to color, sheen, angle of incidence, ambient noise, and texture; and to be approximately an order of magnitude more accurate ($< \pm$ 0.015") than generally considered necessary for this application. The system's shortcomings lie in its restricted handling of glossy black surfaces (only visible between a $-10°$ and $+20°$ angle of incidence), its relatively small 4" field of view (only accurate for targets between 2" and 6" from the sensor) and its non-monotonic nature for targets closer than 2". This final attribute requires the system to be invoked only when the surface is known to be more than 2" away from the sensor and for the overall system to have a sufficient response rate to prevent inadvertent violation of this 2" lower bound.

This sensor system has both analog and digital RS-232 output ports. We chose to directly connect its analog port to an A/D board in order to greatly increase the data acquisition rate; RS-232 is updated at 60 Hz, whereas the analog signal is updated at ~250 Hz. This difference is extremely critical since the surface following system must be faster than the robotic control software's 100 Hz.

The A/D board chosen for this system is a MIZAR "MZ8605-2-00."[3] This board is VMEbus compatible with 16-bit addressing and provides 12-bit resolution on 8 differential input channels at approximately 25 KHz. As such, this board provides

more than enough speed to permit multiple samplings per cycle (employed for statistical signal-noise reduction, see Section 3.2.1) and capability to accommodate multiple sensor systems (permitting eventual expansion of the surface following system).

The CPU board chosen for the surface following system was a Motorola "MVME 133XT."[4] This 25 MHz board is VMEbus compatible with 32 bit addressing and contains an MC68020 microprocessor, an MC68882 Floating Point Coprocessor, and 4 megabytes DRAM. This board directly accesses the A/D board for sensor input and stores global information within its 4 Mb of externally addressable DRAM. This board was used for system development, testing and simulations.

2.2 SYSTEM INTEGRATION

The surface following system was integrated with an existing robotic control system for proof-of-principle testing and demonstration. Unfortunately, there were undocumented timing incompatibilities between the Motorola CPU board used in the surface following system and the FORCE CPU board (model "CPU-30 ZBE")[5] controlling the VMEbus in the demonstration system. These incompatibilities led to porting the surface following system to a FORCE-compatible CPU board.

The CPU board chosen for surface following in the demonstrated system was a FORCE "CPU-33 B/4."[5] This 25 MHz board is VMEbus compatible with 32 bit addressing and contains an MC68030 microprocessor, an MC68882 Floating Point Coprocessor, and 4 megabytes DRAM.

The FORCE "CPU-33 B/4" requires the support of a separate memory board to provide the necessary shared memory. (The surface following system's external communication interface requires absolute addressing of its shared memory. The 4 Mb of DRAM on the Motorola "MVME 133XT" could be easily partitioned to restrict its CPU to one portion of the memory while permitting absolute address accessing of the remainder. This feature was not available on the FORCE "CPU-33 B/4.") Hence, the configuration of the demonstrated surface following system was changed to that shown in Fig. 2.2.

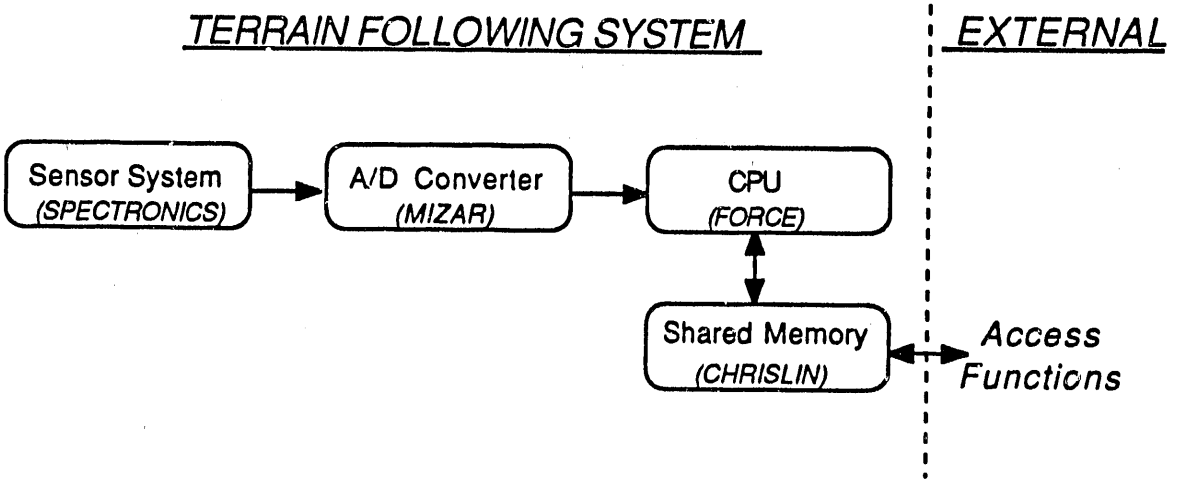

Fig. 2.2. Hardware configuration—demonstrated system.

On the demonstrated system, externally accessible shared memory was stored on a Chrislin "CI-VMEmory 8Mb."[6] As its name indicates, this board is VME compatible and has 8 Megabytes of DRAM. Furthermore, it has a 5 MHz typical access rate and 24/32 bit addressing modes, more than sufficient for this application. The switch to a separate memory board was completely indiscernible to the external accessing functions.

3. SOFTWARE

The surface following software system is responsible for: maintaining an internal map of the observed surface's topology and predicting the amount of vertical, end effector movement necessary to maintain a desired surface offset distance. The surface map is available offline and can be used for user display or additional target analysis. The required movement prediction is output in the form of a ΔZ demand on the end effector's desired position vector and hence, can be coupled with movement demands provided from other sources, e.g., object avoidance, reflex actions, path planning, etc.

This chapter is divided into three sections. The first two describe the surface mapping and surface forecasting algorithms, respectively, while the third presents an overview of the system's software architecture and functional components.

3.1 SURFACE MAPPING

3.1.1 Implementation

A long term surface map is maintained in shared memory for eventual display and analysis. This rectangular map is composed of individual square grids and is implemented as a one dimensional array of grid pairs. The particular grid associated with a given world coordinate is determined at run-time, based on data stored in a map header structure. This implementation permits run-time specification and alteration of the map's size, relative shape, resolution, and physical correspondence and permits reductions in both the map storage costs and access times.

Each grid stores two pieces of information: the average height of the corresponding surface area and the number of readings represented in this average. Although storing the average rather than the sum introduces both round-off errors and additional execution time, the memory space savings more than justify the cost. The round-off error is negligible, due to the raw data's high accuracy relative to the application's requirements. Furthermore, the execution time is only increased by a single multiply and divide operation per map insertion, whereas the memory space required for each grid is reduced by the size of its count field (25% in our implementation).

The average height for each grid is stored as a two byte, integer number of "milli-inches." This permits a 65 inch range of surface height values. A new average is determined by multiplying the current average value by the count, adding the new value, and dividing by the count plus one. The count field is stored as a single byte integer and thus permits counts of up to 255. After 255 samples are obtained which correspond to the same grid, each additional sample only affects the grid's "average height" field—the count remains 255. Thus, the impact of each sample on the "average" value is described by the following:

let $\quad \mathbf{Ave}^{(N)} =$ "average height" for a given grid after N samples,
$\mathbf{h_i} =$ grid height determined by sample $\mathbf{i}$, and
$\mathbf{a_i}^{(N)} =$ weighting of sample $\mathbf{i}$ on $\mathbf{Ave}^{(N)}$

then $\mathbf{Ave}^{(N)} = \sum_{i=1}^{N} (\mathbf{a_i}^{(N)} \mathbf{h_i})$ and the individual weights of the samples are

$$\mathbf{a_i}^{(N)} = \begin{cases} \frac{1}{\mathbf{N}} & \mathbf{N} \leq 256 \\ \frac{1}{256}\left(\frac{255}{256}\right)^{N-256} & (\mathbf{N} > 256) \wedge (\mathbf{i} \leq 256) \\ \frac{1}{256}\left(\frac{255}{256}\right)^{N-i} & (\mathbf{N} > 256) \wedge (\mathbf{i} > 256) \end{cases}$$

The effect of the one byte limit on counts, 256, is clear and this equation could easily be generalized for arbitrary count sizes. From this equation, we see that for values of N less than 257, the function yields a simple average with all samples having an equal 1/N weight. However, for values of N greater than 256, the weight of a single sample is a function of its age relative to the 256 threshold.

This averaging scheme provides many advantages: low execution time, low storage space requirements, simple averaging for all but the most heavily sampled grids, and slow data aging which gives greater impact to more recent samples while retaining significant historical inertia.

This implementation requires only three bytes of memory for each grid, regardless of the number of samples taken. However, since memory is only accessible in multiples of the individual data types' size, e.g., a two byte integer is only accessible on "even" byte boundaries, grids are stored as pairs—each six byte array element stores the information corresponding to two, adjacent grids.

The fielded system has 3 Mb of memory available for the surface map. This provides space for a one million grid map or nearly 70 square feet of surface coverage with a grid resolution of 0.01 sq. in.

3.1.2 Usage

To take advantage of the Spectronics sensor system's high degree of accuracy (<0.01 in.) and extremely small foot-print size (<0.00008 sq. in.), the surface map capability was designed to permit extremely high precision and high resolution mapping. These qualities require the surface to be relatively stable and the sensor's position data to be both accurate and precise. If either of these requirements fail, e.g., the surface changes or the position data is erroneous, then the map is defective and should be reinitialized. However, since high resolution mapping requires long-term map retention for sufficient topological data to be gathered to usefully characterize the overall surface, no automatic reinitialization of the map is provided. Rather, explicit user control functions for map reinitialization and storage are supplied: **MapReset**, **MapRefocus**, and **UpLoadMap**.

MapReset and **MapRefocus** allow the user to reinitialize the map and specify its physical correspondence, based on the user's "region of interest." These functions maximally exploit the data space available for the surface map. The user's region of interest is mapped at the finest resolution that both memory space and position accuracy will allow. (For this application, the finest resolution possible, based on available position accuracy, is expected to be 0.01 sq. in.) If additional memory space is available, the map is symmetrically expanded at this finest resolution.

MapReset and **MapRefocus** only differ in their initial "average height" values for areas which were scanned by the previous map. **MapReset** clears all previous map data from memory and is used whenever the previous data is considered corrupt, e.g., the topology has changed, the world coordinates or sensor data was incorrect, etc. **MapRefocus** permits the system to exploit previous height information for regions common to both maps and is used whenever the user's "region of interest" has changed. If this change permits an increase in the

resolution, the initial "average height" values for grids corresponding to previously mapped regions are taken directly from the earlier, coarser grid(s). If the resolution must decreased, these initial "average height" values are obtained by averaging the heights of the corresponding, finer grids, see Fig. 3.1. In both cases, the count field is set to zero and used as a indicator that the data was obtained under a different mapping scheme and should be replaced whenever new data is obtained.

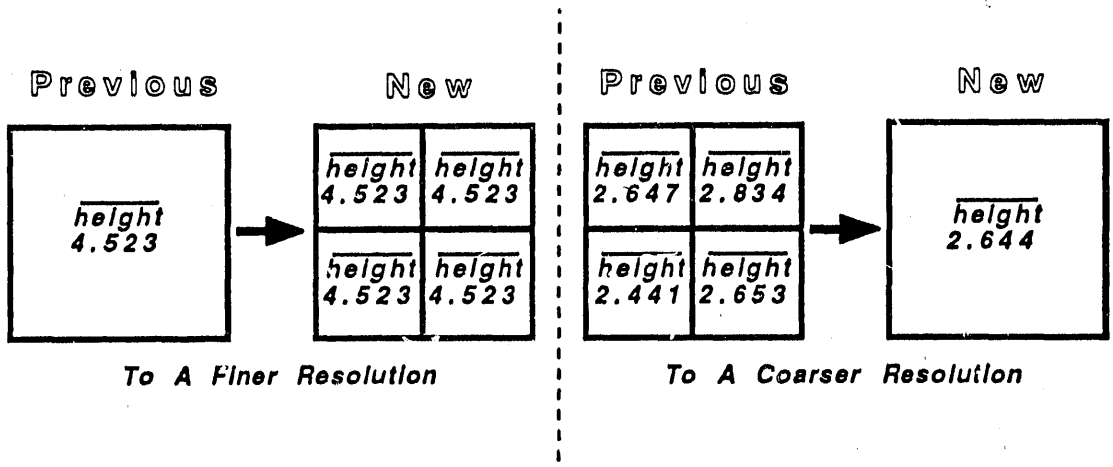

Fig. 3.1. MapRefocus—common grid initialization.

Note that since grid boundaries and resolutions are based on the user's "region of interest," new grids may physically overlap previous grids. In such a case, the new grid's initial height is based on the relative surface area contribution of the previous, overlapping grids, i.e.,

let $\mathbf{G_a^{(b)}}$ = grid **a** in mapping **b**
$\mathbf{h(A)}$ = height of grid **A**
$\mathbf{O(X)}$ = surface area of the region common to all grids in set **X**

then

$$\forall \mathbf{G_j^{(b+1)}},\quad \mathbf{h(G_j^{(b+1)})} = \sum_{\forall \mathbf{i}:\mathbf{G_i^{(b)}}} \mathbf{h(G_i^{(b)})} \frac{\mathbf{O(\{G_i^{(b)},\ G_j^{(b+1)}\})}}{\mathbf{O(\{G_j^{(b+1)}\})}}$$

Note that this general equation provides for both forms of grid initialization shown in Fig. 3.1.

The desire to maximally utilize the available memory space leads to an additional problem. The new map must be stored in the same memory space as the old map without overwriting any information necessary for the proper reinitialization of the map, regardless of any change in the "region of interest" or in the resolution. This "in-situ" reinitialization requires two passes of the map's data structure. During the first pass, the overlapped region is scanned from beginning to end and moved to the top of the data structure. The second pass scans this region backwards and moves the data to its correct location. If the resolution is becoming finer, "expansion" of the previous grids occurs during the second pass, while the data is being written to its final position, see Fig. 3.2. If the resolution is becoming coarser, "compression" of the previous grids occurs during the first pass, while the

data is being moved to the top of the data structure, see Fig. 3.2. This method ensures that the more compact version of the overlapped region is what is stored at the top of the data structure, thereby leaving sufficient memory space to avoid overwriting subsequently needed data

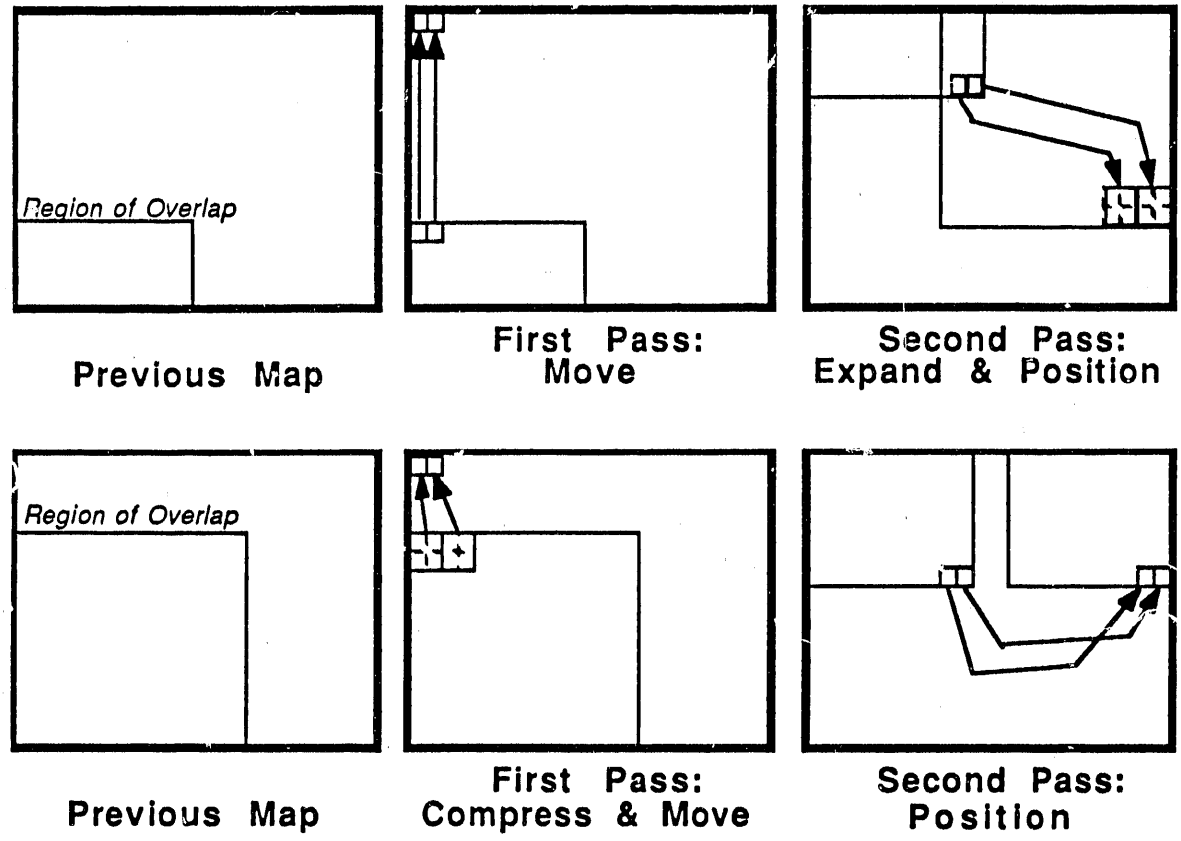

Fig. 3.2. MapRefocus—in-line reinitialization.

UpLoadMap provides long-term storage of maps to disk. This function writes the map header structure and all map data to a text file for subsequent analysis or display. Conceivably, this capability could be used to display composite, high resolution maps of larger surfaces or to reinstall a previous map when rescanning a surface.

3.2 SURFACE FORECASTING

Surface forecasting analyzes recent sensor data and end effector positions to characterize the target's surface, predict subsequent target points, and present necessary demands on the end effector's movement to maintain a constant sensor-to-surface stand-off distance. The system was designed to be generally applicable to many different types of surfaces from smooth, man-made targets, e.g., floors, containers, tools, etc., to rough, natural terrain, e.g., sand, gravel, turf, etc. However, since exploiting specific knowledge of a given target's relative smoothness and the global relevance of its local slopes can have a tremendous impact on a system's forecast efficiency, accuracy and responsiveness, the system permits inclusion of such surface characteristic knowledge through user definable tuning

parameters and employs a hierarchical data processing and representation scheme to isolate the handling and tuning of signal noise, textural sensitivity, local responsiveness, and forecast stability. Each of these hierarchical levels and their respective tuning parameters are presented in the following four sections.

3.2.1 Signal Noise

The surface following software system requires as input both the sensor's distance measurement and the end effector's position and orientation values. From this data, the absolute coordinates of the sensor's footprint are calculated and used for both surface mapping and surface forecasting. Inaccuracies in these input values due to signal noise can be reduced by averaging across multiple samples. Since these input values are from independent sources and available at different rates, separate tuning algorithms and parameters are employed.

Sensor data is available from the A/D board at ~25 KHz, see Section 2.1. Hence, we assume successive readings are measuring the same physical target point and can be directly averaged. The tuning parameter, **Sensor_Readings**, defines the number of successive readings across which to average. Based on a detailed empirical analysis of our particular hardware system's noise level, a value of seven (7) was chosen for the **Sensor_Readings** parameter.

The end effector's position and orientation values are provided by a completely external process at ~32Hz. Since the surface forecast must be updated at >100Hz, the system cannot wait for multiple readings to be available. Instead, the noisy readings are used with the smoothed sensor data (described above) to approximate the coordinates of the target point, and a rolling average of the more recent such coordinates is used as the current value. The sample size for this rolling average is set by the user's parameter **EE_Readings**. While this method effectively reduces signal noise, it also delays the system's response to end effector movements. **EE_Readings** should be set with this trade-off in mind. (Clearly, a better solution to this problem is to average across multiple encoder readings before calculating the end effector's position and orientation values. Unfortunately, that solution could not be implemented without access to the external process making those calculations.)

3.2.2 Textural Sensitivity

Due to the sensor's high degree of accuracy (<0.01 in.) and small footprint size (<0.00008 sq. in.), minor surface changes commonly considered texture or granulation are measurable and could be used to alter the topological forecasting. For the system to be useful for many different applications and surface types, the level of textural sensitivity must be adjustable.

Textural sensitivity is controlled by dividing the target space into "intervals" within which all data is averaged together, much like the grids in surface mapping. These "intervals" correspond to cylindrical volumes of the target space, with radius **Interval_Size** and axis parallel to the Z (height) axis. Thus, the user's specification of **Interval_Size** alters the perceived resolution of the physical world, see Fig. 3.3. (This implementation has the secondary benefit of greatly reducing the storage space requirements necessary to make meaningful forecasts at the end effector's physical scale.)

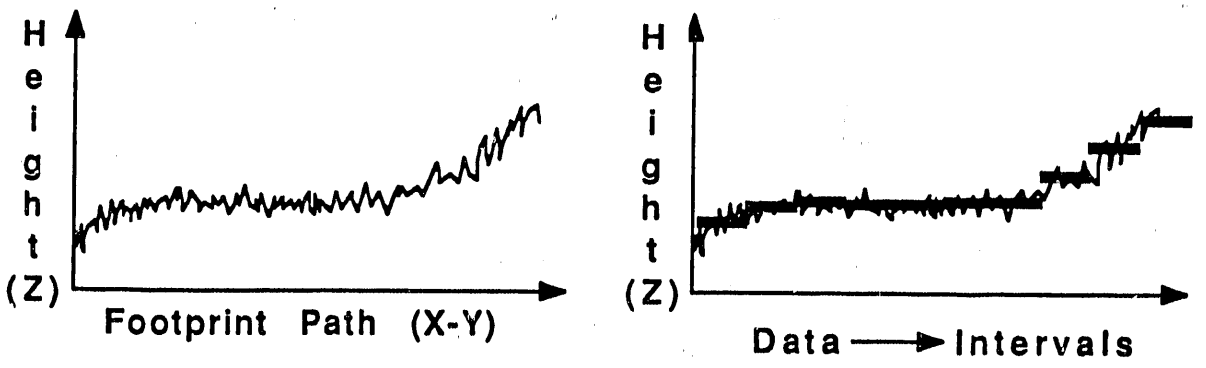

Fig. 3.3. Adjusting sensitivity—data averaged into intervals.

However, unlike the grids in surface mapping, the intervals are dynamically positioned based on the actual path of the sensor's footprint and intentionally overlapped to prevent possible oscillation between adjacent intervals. This dynamic positioning ensures the intervals have a relatively uniform sampling base and reflect a simi ar span of the footprint's travel path, see Fig. 3.4. This, in turn, permits the intervals to be used with equal weight when extrapolating the target's surface during surface forecasting.

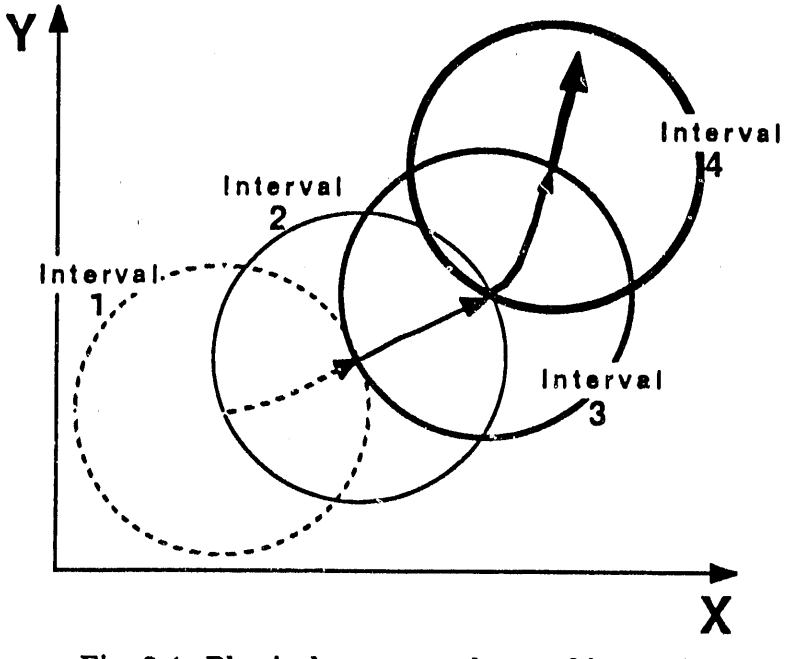

Fig. 3.4. Physical correspondence of intervals.

While the sensor's footprint remains within **Interval_Size** inches in the X-Y hyperplane from the center of the current interval, the target point is associated with that interval. However, whenever the path of the sensor's footprint exceeds **Interval_Size** inches from the center of the current interval, a new interval is created, centered at the current point, see Fig. 3.4. (The arrows indicate the path of the sensor's footprint and their boldness indicates the interval with which the data corresponds.)

If the next sensor reading occurs more than one interval from the previous interval's range, i.e., the current sensor's footprint exceeds 2 * **Interval_Size** inches from the center of the current interval, the unsampled, intervening path is interpolated and the appropriate intervals created and positioned along the route, see Fig. 3.5. This condition results whenever the sensor's footprint moves too swiftly, relative to the **Interval_Size**, or there is an intermittent loss of sensor data, e.g., when the target suddenly drops out of range. This path interpolation permits smooth forecasting across difficult target surfaces and sustains the constant positional relationship between successive intervals.

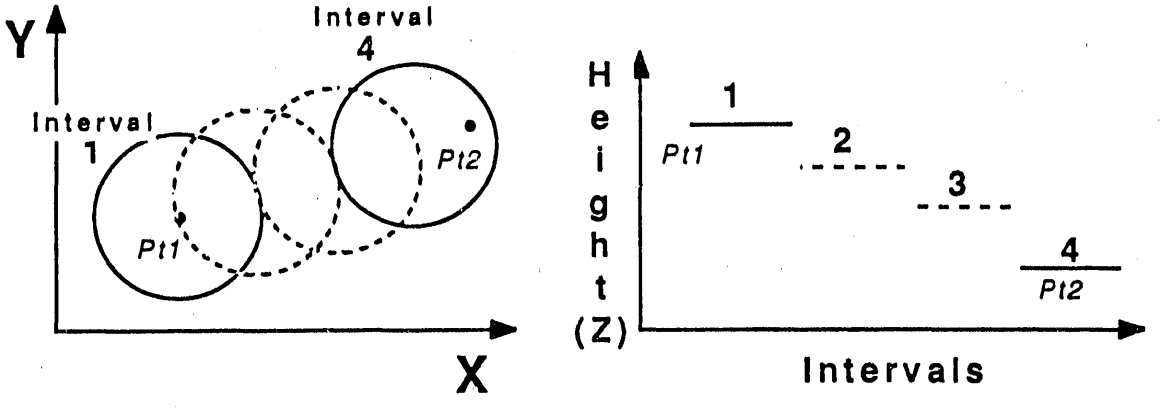

Fig. 3.5. Interpolation of intervals—position and height.

3.2.3 Local Responsiveness

Responsiveness at this level of the hierarchy refers to the degree of impact that individual interval heights have on the overall forecast. If the impact is too great, non-globally relevant changes in slope will erroneously alter the forecast. However, if the impact is too low, adaptations to significant surface changes will be excessively delayed. As with textural sensitivity, the desired level of responsiveness is dependant on the particular application involved. Hence, for the surface following system to be generally useful, the system's responsiveness to local changes in slope must also be adjustable.

Control of the system's responsiveness is provided by combining disjoint groups of contiguous intervals into "ranges," where each range consists of **Range_Size** number of intervals and has a height equal to the average height of its constituents, see Fig. 3.6. (Note the effect of **Range_Size** on the handling of the aberrant interval (fourth from the left) and the change in background slope.)

As new intervals are created, the ranges are dynamically redefined to use the most recent interval data. Hence, the current range always incorporates the most recent **Range_Size** intervals. This method ensures the use of the most relevant target data and a uniform sampling base for each range. For efficiency, all active intervals are stored in a single ring data structure with pointers indicating the head of each range. This implementation permits easy range shifting when new intervals are created.

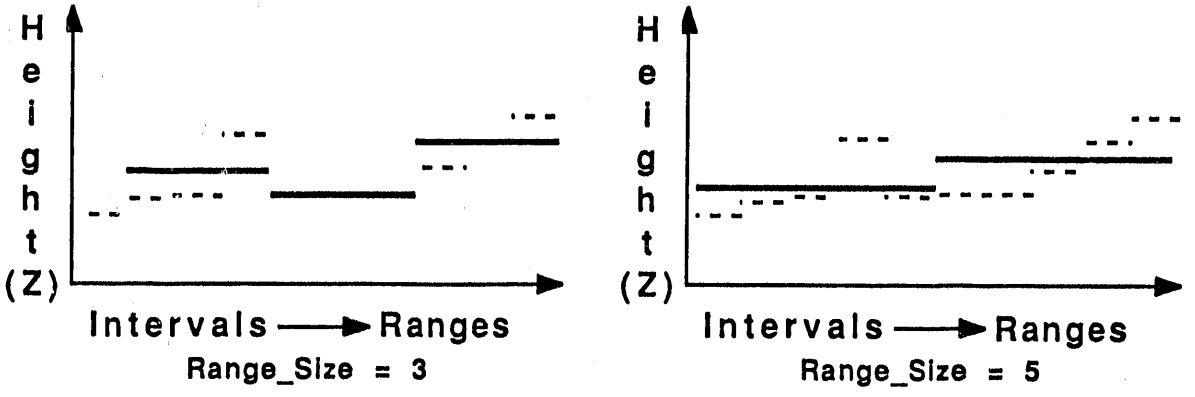

Fig. 3.6. Local responsiveness—intervals averaged into ranges.

3.2.4 Forecast Stability

Forecast stability, or non-volatility, is provided in three ways: by basing the forecast on numerous, widely spaced samplings, through the use of intervals and ranges; by masking the forecast output to permit user control of its precision; and by extrapolating the target's surface during periods of sensory depravation, and thereby increasing the system's overall target applicability. The following paragraphs individually detail these three stabilizing techniques.

First, surface forecasting linearly extrapolates the two most recent range heights to predict the target's underlying slope, Φ. This slope is used in conjunction with the most recent measure of the sensor's footprint's lateral speed to estimate the change in target height, ΔH, since the previous sensor reading. The vertical demand placed on the end effector, ΔZ, is then the sum of ΔH and the difference between the desired and previous actual surface offset distances.

Second, since for some applications one may prefer stability of end effector's movement over its extreme accuracy, a hysteretic function is placed on the ΔZ values actually output. The user parameter **Permitted_Variance** specifies the amount by which ΔZ must change before any demand is relayed to the end effector.

Third, forecast stability is provided during intermittent periods of sensor data loss, e.g., when the target is out of detection range. During these periods, ΔH is extrapolated based on Φ, the most recent lateral speed estimation, and the elapsed time.

Although this system uses two-point linear extrapolation to estimate Φ, it can readily be extended to higher order techniques and/or the use of additional ranges. Two-point linear extrapolation was determined to be sufficiently sophisticated for this application (soil), since the benefit of higher order extrapolation techniques requires the existence of a greater degree of local surface information content than is present in this application.

3.2.5 Usage

Four external access functions communicate with the surface forecasting algorithms: **SFReset**, **DevVector**, **RawDistance** and **EEPos**.

SFReset allows the user to reinitialize surface forecasting and the desired surface offset distance. This function writes its single parameter value (the desired surface offset distance) to shared memory and triggers the surface following system to reset all local surface forecasting variables, e.g., interval positions and heights, range heights, previous ΔZ value, etc.

DevVector returns a 6×1 vector indicating the surface following demand to be placed on the end effector. This vector permits inclusion of both positional and orientational demands, though only the ΔZ value is currently being prescribed. Returning the 6×1 vector rather than only a ΔZ permits direct expansion of the system to 3-dimensional surface following without having to modify the user-interface.

RawDistance returns the most recent sensor-to-target distance measurement used by the surface following system. This value is obtained by averaging multiple sensor readings to reduce the signal noise, see "signal noise" above.

An external process is responsible for calculating the end effector's most recent position and orientation data in world coordinates and passing these values as a parameter to **EEPos**. The external access function **EEPos** then updates shared memory accordingly.

3.3 ARCHITECTURE

The surface following system uses a modular, top-down approach to facilitate the individual development, verification, and extension of its capabilities. All complex and globally accessible data structures, physical hardware constants and user defined parameters are externally defined to permit their use by other processes, e.g., those executing the external access functions, and their rapid adaptation to hardware or application specific changes. This design permitted the previously described hardware configuration change, see Section 2.2, to be software realized in a matter of minutes.

The surface following software is designed to be a robust, stand-alone system, completely independent of the operating system environment used to control the actual robot. Furthermore, since the software must operate in real-time, it is executed on a dedicated processor rather than risk the unpredictable delays associated with time-shared environments. For these reasons, the system exclusively communicates through absolute addressing of shared memory and provides all shared memory management.

Conceptually, this system passes through three basic phases: initializing memory, responding to external requests, and processing input data, see Fig. 3.7. The primary functional components of each of these phases will be detailed in this section.

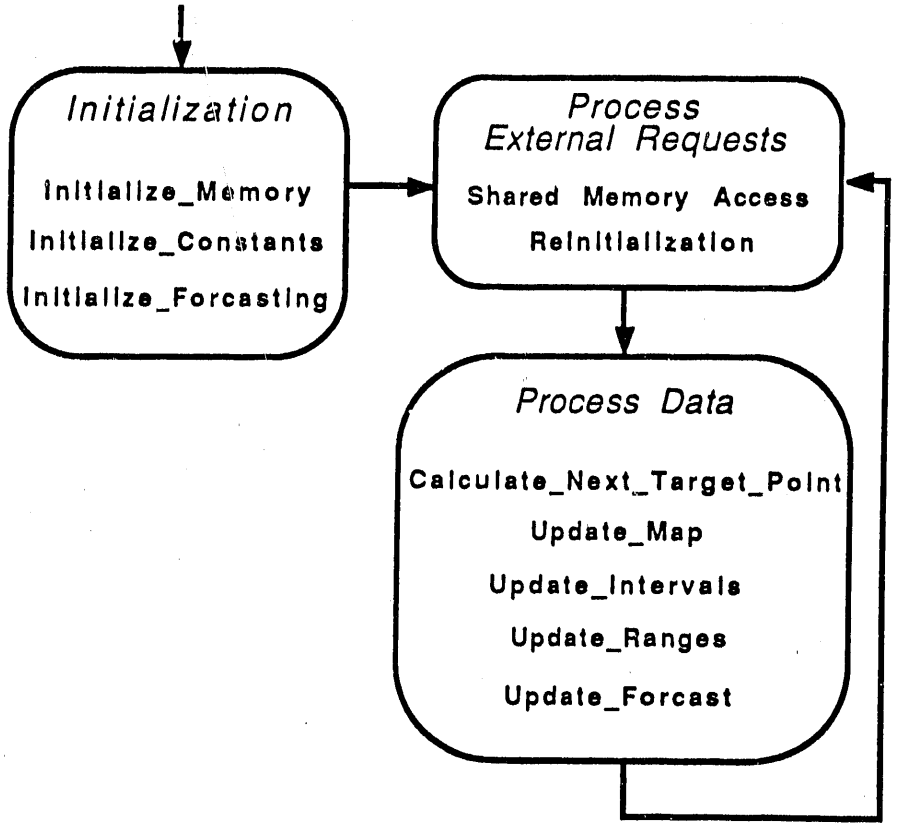

Fig. 3.7. Surface following—functional breakdown.

3.3.1 Initialization

This first phase of the system is responsible for partitioning and initializing the shared memory and creating and initializing all surface forecasting data structures. Initial values for physical hardware constants are defined in a single header file. These constants include the absolute starting address and size reserved for shared memory. Default values for user defined tuning parameters are encoded in a single function, **Initialize_Constants**, which permits their values to be calculated at run-time.

Three separate functions are responsible for initialization. **Initialize_Memory** partitions the shared memory space and sets-up absolute addressed pointers to each of its internal structures, including the surface map and data logging arrays, see Fig. 3.8. **Initialize_Constants** fills shared memory with the default values of the non-local variables, i.e., shared and user defined variables, and **Initialize_Forecasting** creates and/or initializes the local data structures used in surface forecasting, e.g., intervals, ranges, etc., based on position and sensor input data. (Note that the surface map is not automatically reset during initialization, see Section 3.1.2.)

After initialization is complete, the system infinitely cycles through the following two phases.

Map Header
Tuning Parameters
Input Data, e.g., sensor data, end effector position
Output Data, e.g., $\Delta Z, \Phi$
Sensor Position and Orientation w.r.t. end effector
Range Data, i.e., ring of forecast intervals
Surface Map Array
Data Log, i.e., input & output data with cycle numbers

Fig. 3.8. Shared memory—conceptual breakdown.

3.3.2 Process External Requests

There are two types of external requests handled by the surface following system, requests for reinitialization or for shared memory access. All external requests are communicated to the surface following system through specialized flags defined in the shared memory. The surface following system inspects these flags before processing the data for each cycle.

Reinitialization requests permit the user to invoke a controlled restart of either surface mapping or surface following. The user can restart surface mapping by calling either of the external access functions: **MapReset** or **MapRefocus**, see Section 3.1.2. The user can restart surface forecasting by calling **SFReset**, see Section 3.2.5. The **SFReset** function reinitializes all of the intervals, ranges, and local variables associated with surface forecasting and is, in fact, the same function executed during system initialization described above. These three access functions modify appropriate shared memory values, e.g., **MapRefocus** downloads the user's new "region of interest," before flagging the surface following system into action.

Memory access requests provide the user with exclusive access to the shared memory. This is necessary to ensure data integrity, since most of the data stored in shared memory cannot be accessed in a single cycle. Shared memory access is controlled and communicated through a four-step flagging protocol:

Flag Value	Interpretation
1	External request for access
2	System acknowledges and waits
3	External access completed
0	System acknowledges and proceeds

(Note that a four-step protocol is necessary since the communicating systems may be restarted independently and must then determine the status of shared memory access to prevent its corruption.)

3.3.3 Process Data

During each cycle in which sensor data is obtained, the world coordinates of the current target point are approximated (**Calculate_Next_Target_Point**) based on the most recent end effector position and orientation data and smoothed sensor data stored in shared memory. (Note that the sensor's position and orientation with respect to the end effector's origin is a constant, stored in shared memory during system initialization.) This target point is then used to update both the surface map (**Update_Map**) and forecasting data structures (**Update_Intervals**, **Update_Ranges**, **Update_Fore-cast**), see Fig. 3.7. These various functions are modularized to permit independent analysis of different approaches, e.g., alternative methods of combining intervals to control local responsiveness can be analyzed without effecting the other levels of the forecasting hierarchy. During cycles in which sensor data is unavailable, only surface forecasting is performed, see Section 3.2.4.

4. CONCLUSIONS

The surface following system described in the preceding chapters was successfully tested and demonstrated in the FY91 Hanford Demo. The target surface used in this demonstration was a natural, rolling terrain of finely crushed clay stones with randomly scattered, man-made objects and ramps. Although full integration with the robotic system at Hanford required several hardware and software changes, e.g., porting the system to a FORCE CPU and the use of a separate memory board, see Section 2.2; the extension of signal noise reduction methods to the end effector's position and orientation values; see Section 3.2.1; etc., the overall design of the surface following system permitted rapid compliance with those requirements. Furthermore, the system provided effective, real time surface forecasting for both the rough clay and smooth objects, despite their extremely different surface character. (Note that the drastic contrast in both the surface textures and the global relevance of local slopes prevented the use of optimal tuning parameter values within any given region, i.e., target inconsistency mandated the use of (sub-optimal) general purpose, parameter values.)

Superior performance in an unconstrained environment will ultimately require: 1) automatic adaptation to the topological surface characteristics currently addressed by user-defined, tuning parameters; and 2) incorporation of multiple sensor systems to provide orientational demands for 3-dimensional surface following. Development and validation of these new capabilities will require extensive empirical analysis of the system's behavior over a broad range of target types, slopes, and transitions. Hence, for convenience and efficiency, the surface following system must be implemented on a readily accessible robotic control system, e.g., HERMIES-III.[7]

ACKNOWLEDGMENTS

Research sponsored by the Office of Technology Development, U.S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

I wish to thank Alex L. Bangs for his immeasurable assistance in the hardware design and implementation and both he and Phil L. Butler's help with system integration for the FY91 Hanford Demo.

REFERENCES

1. Spectronics, Inc., 9655 SW Sunshine Court, #500, Beaverton, OR 97005.
2. J. E. Baker, *Empirical Characterization of a High Intensity LED Proximity Sensor*, ORNL/TM-11972, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, November 1991.
3. *MIZAR MZ 8605/8610 Analog Input/Output Modules*, MIZAR Publ. No. 4425151, May 1989, 1st ed., Mizar Inc. 1419 Dunn Drive, Carroliton, Texas.
4. *MVME133XT VMEmodule 32 Bit Monoboard Microcomputer User's Manual*, MOTOROLA Report MVME133XT/D1, April 1988, 1st ed., Motorola Inc., 2900 S Diablo Way, Tempe, Arizona.
5. FORCE Computers, 3165 Winchester Blvd., Campbell, CA 95008.
6. *Chrislin Industries CI-VMEmory VMEbus Dynamic Memory Technical Manual*, Rev. A.1, Version I, Chrislin Industries, 31332 Via Colinas, #106, Westlake Village, California.
7. Weisbin, C. R., et. al., "HERMIES-III: A Step Toward Autonomous Mobility, Manipulation and Perception," *Robotica* **8**, 7–12 (1990).

INTERNAL DISTRIBUTION

1. B. R. Appleton
2. G. A. Armstrong
3–7. J. E. Baker
8. A. L. Bangs
9. M. Beckerman
10. B. L. Burks
11. P. L. Butler
12. R. J. Carter
13. J. R. Einstein
14. C. W. Glover
15. W. R. Hamel
16. J. P. Jones
17. H. E. Knee
18. G. E. Liepins
19. R. C. Mann
20. E. M. Oblow
21–25. F. G. Pin
26. S. A. Raby
27. D. B. Reister
28. P. F. Spelt
29. F. J. Sweeney
30. M. A. Unseren
31. R. C. Ward
32–33. Laboratory Records Department
34. Laboratory Records, ORNL-RC
35. Document Reference Section
36. Central Research Library
37. ORNL Patent Section

EXTERNAL DISTRIBUTION

38. Office of Assistant Manager, Energy Research and Development, Department of Energy, Oak Ridge Operations, Oak Ridge, TN 37831
39. Dr. Peter Allen, Department of Computer Science, 450 Computer Science, Columbia University, New York, NY 10027
40. Dr. Wayne Book, Department of Mechanical Engineering, J. S. Coon Building, Room 306, Georgia Institute of Technology, Atlanta, GA 30332
41. Professor Roger W. Brockett, Wang Professor of Electrical Engineering and Computer Science, Division of Applied Sciences, Harvard University, Cambridge, MA 02138
42. Prof. John J. Dorning, Department of Nuclear Engineering and Physics, Thornton Hall, McCormick Rd., University of Virginia, Charlottesville, VA 22901
43. Dr. Bill Drotning, 1515 Eubank, SE, Organization 1414, Sandia National Laboratories, Albuquerque, NM 87123
44. Dr. Steven Dubowsky, Massachusetts Institute of Technology, Building 3, Room 469A, 77 Massachusetts Ave., Cambridge, MA 02139
45. Dr. Avi Kak, Robot Vision Lab, Department of Electrical Engineering, Purdue University, Northwestern Ave., Engineering Mall, West Lafayette, IN 47907
46. Dr. James E. Leiss, 13013 Chestnut Oak Dr., Gaithersburg, MD 20878
47. Dr. Oscar P. Manley, Division of Engineering, Mathematical, and Geosciences, Office of Basic Energy Sciences, ER-15, U.S. Department of Energy–Germantown, Washington, DC 20545
48. Wallace Masters, Spectronics, Inc., 9655 SW Sunshine Court, #500, Beaverton, OR 97005
49. Prof. Nevill Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green St., Urbana, IL 61801
50. Dr. Wes Snyder, Department of Radiology, Bowman Gray School of Medicine, 3005 Hawthorne Dr., Winston-Salem, NC 27103

51. Prof. Mary F. Wheeler, Rice University, Department of Mathematical Sciences, P.O. Box 1892, Houston, TX 77251
52–61. Office of Scientific Technical Information, P.O. Box 62, Oak Ridge, TN 37831

END

DATE
FILMED
3/17/92