N91-20037 p26

1990 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

ROBOT TRACKING SYSTEM IMPROVEMENTS AND VISUAL CALIBRATION OF ORBITER POSITION FOR RADIATOR INSPECTION

PREPARED BY:	Dr. Gregory Tonkay
ACADEMIC RANK:	Assistant Professor
UNIVERSITY AND DEPARTMENT:	Lehigh University Department of Industrial Engineering
NASA/KSC	
DIVISION:	Mechanical Engineering
BRANCH:	Special Projects (RADL)
NASA COLLEAGUE:	Mr. V. Leon Davis
DATE:	July 31, 1990
CONTRACT NUMBER:	University of Central Florida NASA-NGT-60002 Supplement: 4

ACKNOWLEDGEMENTS

I would like to thank NASA and ASEE for the chance to participate in this program for a second year. My professional knowledge of launch operations, shuttle support, and robot applications, has increased greatly based on my interactions with the robotics group at NASA and the employees of Boeing Aerospace Organization, the engineering services contractor for the RADL. A special thanks to Leon Davis, my NASA colleague for giving me the freedom to change my emphasis as new and more interesting problems appeared.

ABSTRACT

This report dealt with two separate topics: (1) improving a robotic tracking system and (2) providing insights into orbiter position calibration for radiator inspection. The objective of the tracking system project was to provide the capability to track moving targets more accurately by adjusting parameters in the control system and implementing a predictive algorithm. A computer model was developed to emulate the tracking system. Using this model as a test bed, a self-tuning algorithm was developed to tune the system gains. The model yielded important findings concerning factors that affect the gains. The self-tuning algorithm will provide the concepts to write a program to automatically tune the gains in the real system.

The section concerning orbiter position calibration provided a comparison to previous work that had been performed for plant growth. It provided the conceptualized routines required to visually determine the orbiter position and orientation. Furthermore it identified the types of information which are required to flow between the robot controller and the vision system.

SUMMARY

The RADL (Robotics Application Development Laboratory) has been working on the robotic tracking for several years. The application of tracking is based on using a robot to mate and de-de-mate an umbilical connection on the shuttle launch pad as it is swaying in the breeze. The tracking system has many parameters which must be set. Unfortunately, there are so many parameters that it has been impossible to manually tune them. This project developed a model of the tracking system and then implemented a self-tuning algorithm to tune the variables in the model. With the success of the program there is now incentive to apply the self-tuning algorithm to the real tracking system. This paper describes the model and self-tuning algorithm in detail.

The paper discusses types of error criteria and supports the use of a combination of minimizing maximum deviation and mean absolute deviation.

By observing the model self-tuning the control system, several important discoveries were made or verified:

Target velocity affects the gains

Optimal gains could be negative for short trials

Length of tracking trials significantly changes the optimal gains

Start up biases exist which affect gains

Error criteria affects gains

Next, the concepts required for the implementation of a predictive algorithm are discussed. Several important issues are raised with recommendations of how to proceed initially. Although no experimental analysis was performed, it would be feasible to modify the tracking model to give some insights into the questions.

Finally, a discussion is presented about the requirements for a vision system attached to a robot to determine the position of the orbiter. Two methods of location determination are discussed: triangulation and analysis of known features. The issues involved in an interface protocol are also explored. Finally, a proposed scenario is given for the orbiter orientation determination task.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I INTRODUCTION

1.1 ROBOTICS AT KENNEDY SPACE CENTER

The mission of Kennedy Space Center is to provide manpower and support for fast, efficient and safe preparation of launch vehicles. Robotics can be a key ingredient to satisfy this mission. Many of the operations performed at Kennedy Space Center are dangerous or repetitive which make them ideal candidates for robots. The design and servicing procedures of present space vehicles and launch procedures make it difficult to implement robotics. However, the next generation spacecraft will most assuredly be designed with robots in mind. This requires KSC personnel to have familiarity with robots and related hardware such as sensors and control systems. The RADL (Robotics Applications Development Laboratory) provides this experience to NASA and its contractors.

1.2 OBJECTIVE OF THIS RESEARCH PROJECT

The objective of this research project was to assist NASA personnel in two separate areas: 1) target tracking and 2) radiator inspection. The body of this report will be divided into two parts, each describing the background and results of one project.

The objective of the target tracking project was to provide the capability to track moving targets more accurately. The physical tracking problem is a robot tracking the movements of the external tank of a shuttle on the pad so that an umbilical connection can be mated. Success has been achieved at tracking targets and mating umbilical connections when moving several inches per second. The recommendations in this report should provide the capability to track faster targets with greater accuracy using two separate methods. The first method was to write a computer program to self-tune the software PID loop found in the computer generating the system moves. This PID loop will be described in more detail later. The second method was to recommend implementation procedures for a predictive algorithm to predict the position of the target.

The radiator inspection project involves a robot being designed to inspect the radiators on the inside of the cargo bay doors of each orbiter. This robot will travel on a 67 foot track to provide coverage for all the radiator panels. Since the orbiter is parked in a slightly different position each time it enters the OPF, a calibration must be performed to determine the position of the orbiter and the radiator panels prior to inspection. This report describes the procedure which should be used to determine the orientation of the radiator panels.

II IMPROVEMENT IN TARGET TRACKING PERFORMANCE

2.1 DESCRIPTION OF TRACKING SYSTEM

A block diagram of the target tracking system is shown in Figure 2-1. The major components in this system are an ASEA IRB-90 robot, a MicroVax computer, and a DataCube vision system. The camera for the vision system is mounted on the end effector of the robot. The entire system can be thought of as a control loop. The vision system takes a picture of the target. From this picture it can calculate the relative error in inches (units used by vision system) between the camera and the target. This error is converted to millimeters (units used by the robot) and fed to the MicroVax computer at the approximate rate of 30 error vectors per second. Two buffers are maintained in the vision system. One always contains a completed picture ready for transfer, while the other is processing a picture [1].

In the MicroVax computer, the error vector is fed into a software PID loop. The purpose of this loop is to allow fine-tuning of the system moves. The PID loop outputs the coordinates for a relative move to be executed by the robot. After the relative move coordinates are determined from the PID loop, the absolute move coordinates are calculated by summing the relative coordinates with the previous absolute coordinates. Next the absolute coordinates are converted to quaternions, the orientation notation system used by the robot. Finally, the move command is sent to the robot for execution. Because the communication link between the robot and the MicroVax is slow (9600 baud), a new command is always calculated and waiting in the robot controller for execution. This leads to a problem of sending the robot to a position that is based on data which is one move old and thus introduces a lag into the system. This is an area for improvement and will be discussed in more detail later in this report.

When the robot finishes executing a command, it sends an acknowledgement to the MicroVax and immediately starts executing the next instruction which is already in its buffer. When the MicroVax receives the acknowledgment of the previous move command, it calculates the next move command and sends it to the robot communications buffer. In this way, the next command will already be available for execution and the tracking system will not experience delays due to communications.

2.2 SOFTWARE PID CONTROL LOOP

As stated previously, the purpose of the software PID loop is to transform an error into relative move coordinates for the robot to execute. By properly setting the gains in the PID loop, the

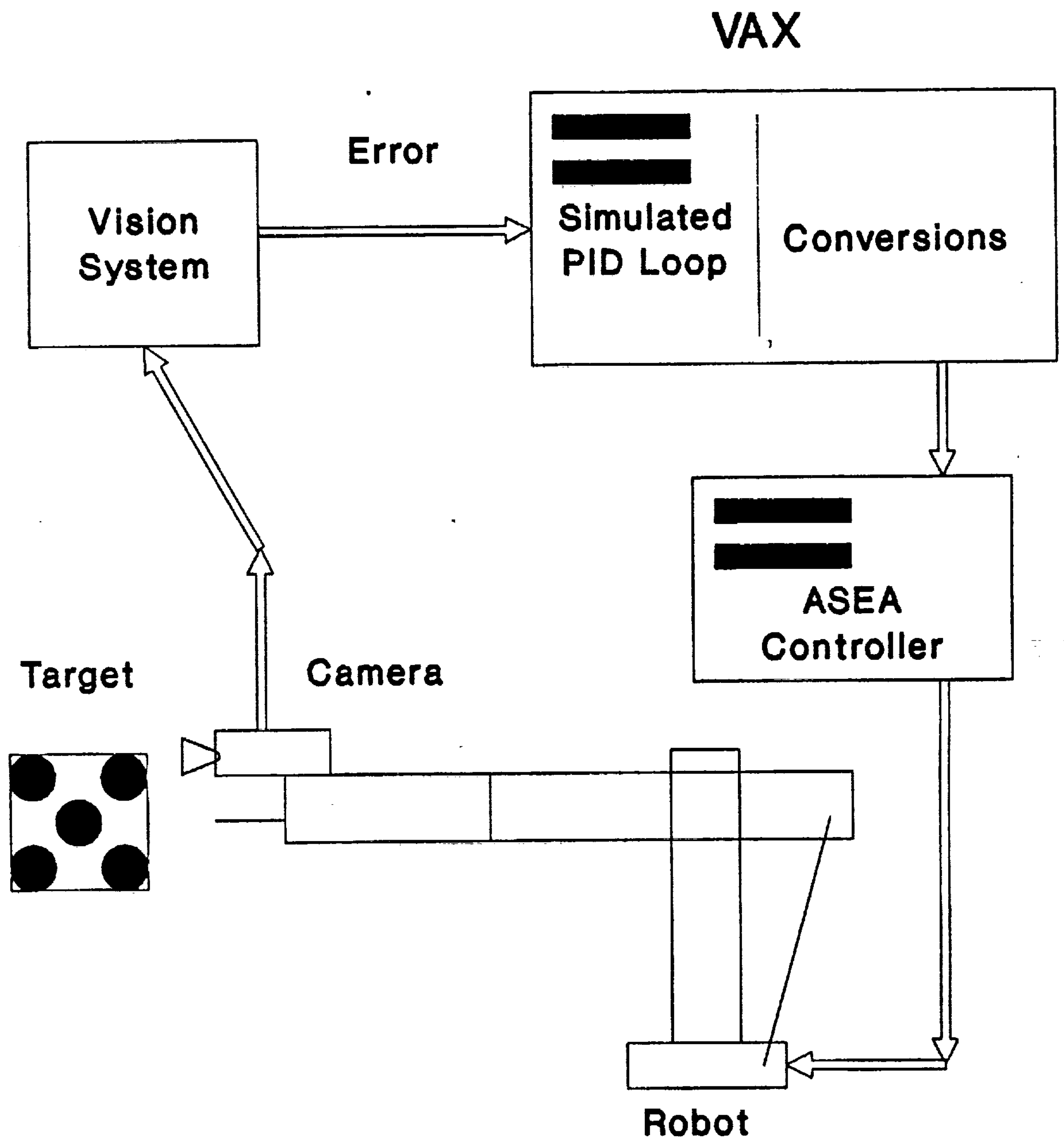


Figure 2-1. Tracking System Hardware

response of the plant (robot) can be substantially increased. There are two equations which represent the integral and derivative aspects of the PID controller, Equations (1) and (2) respectively. A third equation, Equation (3), provides the output of the PID loop.

$$I_i = I_{i-1} + e_i * T_v \tag{1}$$

$$D_i = \frac{D_{i-1} + K_4 (e_i - e_{i-1})}{1 + K_4 + T_v} \tag{2}$$

$$\Delta P = T_r * (K_p * e_i + K_i * I_i + K_d * D_i) \tag{3}$$

where:

I_i = integral error at time i
D_i = derivative error at time i
e_i = most recent calculated error by vision system at time i
K_p = proportional gain
K_i = integral gain
K_d = derivative gain
K_4 = exponential weighting factor in the derivative equation
ΔP = delta move coordinates for the robot from PID loop

The equations above represent a single axis of the system. There are actually 6 sets of equations as represented above corresponding to X, Y, Z, Roll, Pitch, and Yaw. The 18 equations have a total of 24 gain constants.

2.3 PROBLEMS TUNING THE PID LOOP

Tuning the software PID loop presents several problems. First, the optimal gain values depend on the operating parameters of the system, such as target velocity, pattern of target motion, time between vision updates, time between robot moves, maximum robot velocity, and robot acceleration. These parameters are subject to constant change as the system is upgraded. It would take a long time to optimally tune the PID loop manually, at least one man-week. Even if the PID loop were tuned, changing a single operating parameter would require the loop to be re-tuned. These problems necessitate an automatic method to tune the loop.

2.4 MODEL OF TRACKING SYSTEM

To develop an automatic method to tune the loop, valuable system time and manpower would be required during development, debugging, and testing. Therefore, a computerized model was developed to approximate the tracking system. By writing a program to self-tune the PID loop in this model before implementing it on the actual system, time and money would be saved. Furthermore, the tracking model could provide a graphical platform to explain the system and test hypotheses.

Several assumptions/limitations were made in order to reduce the complexity of the model. However, the operating principles of the actual system were used wherever possible. The first and greatest limitation is that of only modeling one axis of motion. Since the real-life target to be tracked is a shuttle on the pad blowing in the breeze, it is assumed that the major motion will be along one axis. Specifically, past research projects have dealt with a motion of 10 inches in the X direction, and 3/4 inch in the Y and Z directions. The same principles used to tune the single axis in this model could be used to tune all of the axes in the actual controller.

The second assumption is that the motion will be sinusoidal with time. The velocity will be at a minimum at the extremes of motion and a maximum through the mid-point. It should be noted that any pattern of motion could be simulated just by changing the motion generating function for the target.

The third assumption is that the robot controller is assumed to be perfect. No overshoot or instabilities result from the controller. While this may sound like a major limitation, it is believed that the time lags in the other systems cause instabilities which are orders of magnitude greater than those caused by the robot controller. The robot controller was modelled to ramp up to its required velocity at a given rate of acceleration. Furthermore, a variable was included to provide a delay between execution of robot commands. This would simulate the robot controller reading and setting up for the next move. The controller was also modeled to have an internal move buffer.

The fourth assumption is that there is no error in the vision system calculations. A possible way to model this error is to add a small random error to each calculation. The vision system was modeled with two buffers. When a robot move must be generated, the completed buffer is used even though the data in the buffer is already old.

Some thought was given to selecting a criteria for judging one set of gains against another. In the final program, the user must select one of the following three criteria:

Mean Absolute Error - This measure treats all errors as equal and attempts to minimize the mean. Arguments could be made against this measure because one extremely large error could be enough to shutdown the tracking system. However the effect of one large error on the overall mean would be insignificant.

Root Mean Square Error - This measure tends to magnify the larger errors (squares the error) and hence solves some of the problems of Mean Absolute Error. However, the effect on the mean of a single large error might still be insignificant.

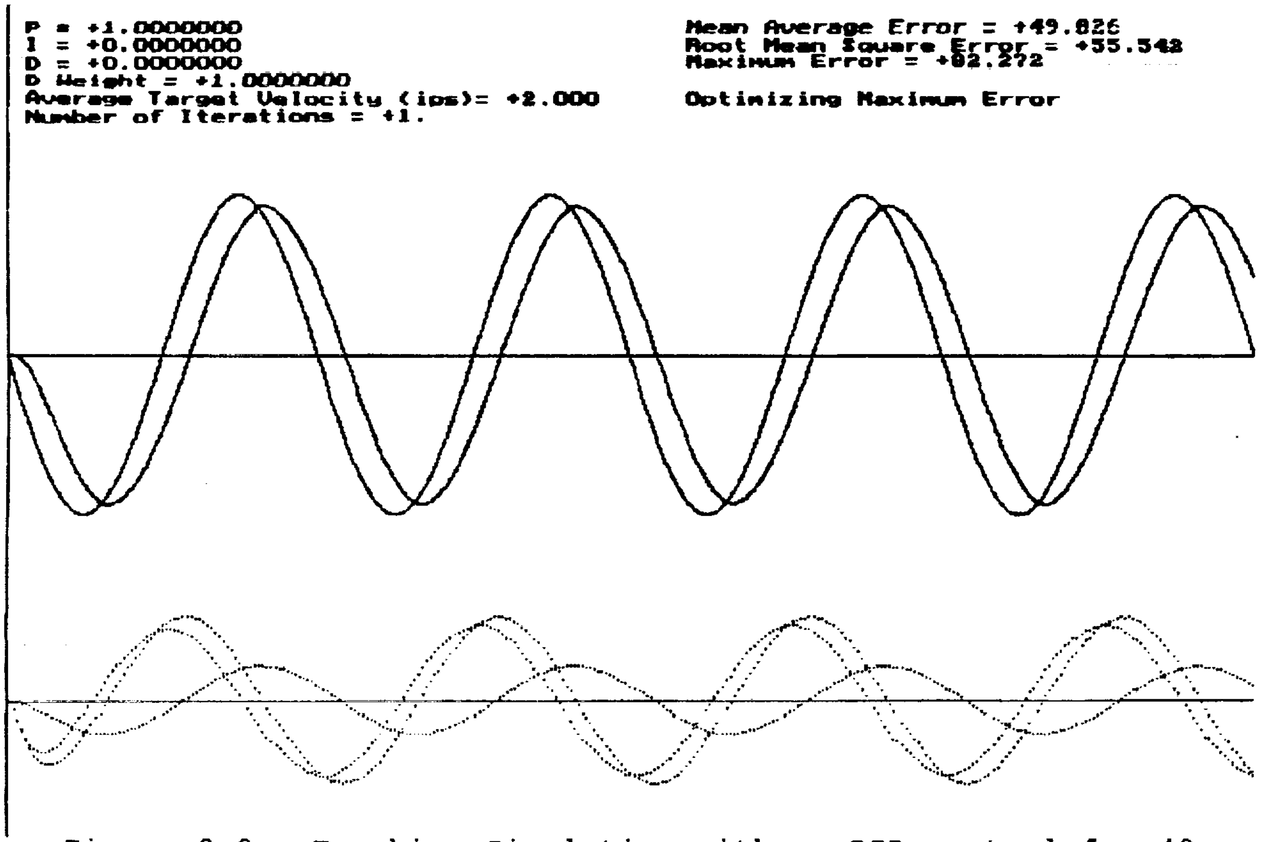
Maximum Absolute Error - This measure records the largest error in the trial but disregards the other errors. Statistically, it is better to use a measure that uses more of the data. However, for the tracking problem, this is a reasonable criteria. Note that in reality a user might be willing to sacrifice a little maximum error in exchange for better mean square error.

Another possible criteria would be a weighted function including terms for the maximum absolute error and the mean absolute error (or root mean square error). This would allow the user to use both measures to choose the optimal gains. Yet another possibility would be to minimize the mean absolute error subject to maintaining the maximum absolute error below some value. For example, if the system could track as long as the error was less than 3 inches, the optimizing routine could minimize the mean absolute error in trials which keep the absolute error less than 3 inches. These last two optimization criteria were not included in the tracking system model. They are mentioned as possible candidates for further work.

There are many variables which the user can change in the model. The most typically changed variables are the 4 gain parameters from Equations (1), (2), and (3), the speed of the target, the maximum velocity and acceleration of the robot, and the amplitude of target motion. Other variables can be changed to perform what-if scenarios. For example, "what would happen if the robot position could be updated 20 times per second instead of 7" or "what would happen if the buffer in the robot were eliminated and the robot had to wait to receive the move command before execution could start?" There are many possible configuration changes that can be quickly and easily tested.

2.5 OUTPUT FROM THE TRACKING SYSTEM MODEL

The model is programmed to graphically show the motion of the target and the robot in time. Figure 2-2 is an example of the tracking system with no PID loop to speed up the robot controller (proportional gain equal to 1 with no derivative or integral gain). Time is shown on the horizontal axis and position is shown on the vertical axis. The error components, e_t, I_t, and D_t from the PID loop, are shown on a second graph at the bottom.



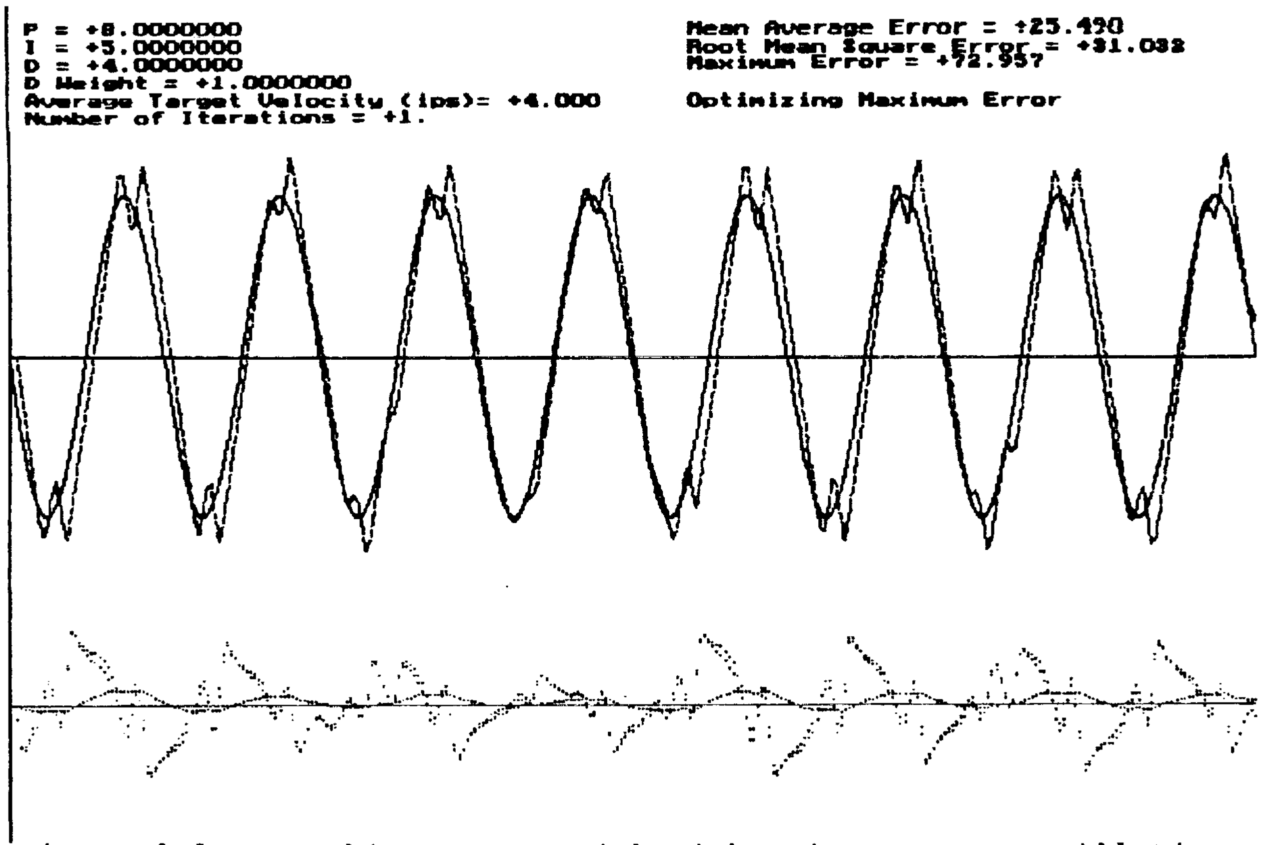
```
P = +1.0000000                                 Mean Average Error = +49.826
I = +0.0000000                                 Root Mean Square Error = +55.542
D = +0.0000000                                 Maximum Error = +82.272
D Weight = +1.0000000
Average Target Velocity (ips)= +2.000          Optimizing Maximum Error
Number of Iterations = +1.
```

Figure 2-2. Tracking Simulation with no PID control for 48 seconds

As expected from experience, the robot lags the target because of the buffers, delays, and inaccuracies in the system. As the gains are increased, the robot lags less. As the gains continue to increase, eventually the robot will overshoot when the target slows down and reverses direction. Some overshoot is acceptable, but increasing the gains further can cause oscillation as shown in Figure 2-3 and eventually system instability.

2.6 SELF-TUNING ALGORITHM

The system model as described above could be used to determine which projects should be attempted first in order to receive maximum payback in the form of increased tracking capability. However, a major goal of this project was to write a computer program to self-tune the gains in the PID loop. This was accomplished faster by first modeling the tracking system as described previously and then writing a self-tuning algorithm to alter the gains in the model to optimize one of the measures of performance. It also proved the concept before tying up expensive robot and technician time. Now that the self-tuning concept has been demonstrated, it can be implemented on the system hardware.



```
P = +8.0000000                                   Mean Average Error = +23.490
I = +5.0000000                                   Root Mean Square Error = +31.032
D = +4.0000000                                   Maximum Error = +72.957
D Weight = +1.0000000
Average Target Velocity (ips)= +4.000            Optimizing Maximum Error
Number of Iterations = +1.
```

Figure 2-3. Tracking System with High Gains Causes Oscillations

The approach used to self-tune the PID loop was a response surface, hill-climbing technique. If there were only two gains, the response surface would look like a mountain. The elevation, or Z axis, would be analogous to the optimization criteria. The two gains would be analogous to longitude and latitude coordinates, or X and Y axes. As the gains, or X and Y coordinates, are changed, the mountain climber will either ascend, descend, or stay at the same altitude. There are several hill climbing techniques that could be used to reach the top of the mountain. In the coded algorithm there are actually 4 gains to be changed rather than 2, but the concept is still the same.

The method implemented in the model is the "method of steepest ascent." The tracking error is known for the gains set at their current values. Then one at a time, each gain is incremented a certain amount while the other gains remain at their original values. If for any gain the error gets worse, the gain is decremented instead of incremented and another error calculation is performed. Thus with 4 gains, a minimum of 4 trials and a maximum of 8 trials are performed before the new gain values are selected. The new gain values will be the ones used in the trial which produced the greatest reduction in error.

One further refinement added to the algorithm was to adjust the increments added to the gains so that a course search was conducted. Once the gains were tuned with the large increments, smaller increments were used to refine the gains. Finally, even smaller increments were used to attempt to reach the pinnacle of the summit. Trials have shown that depending on the increments that are specified for the trials, it could take several hundred iterations to achieve a final solution.

In order for this coded algorithm to work with the tracking system instead of the simulation, a function must be defined which takes as its inputs the four gain values, runs the tracking system, and returns a value representing the measure of error. In addition, other modifications might be required in the tracking system for automatic operation.

Because the hill climbing technique is a heuristic, it may find a local optimal solution instead of a global optimal solution. To reduce this possibility, the algorithm could be run several times, each with different starting gains. While this will not guarantee a global optimal solution, it will increase the chances of finding a global optimal.

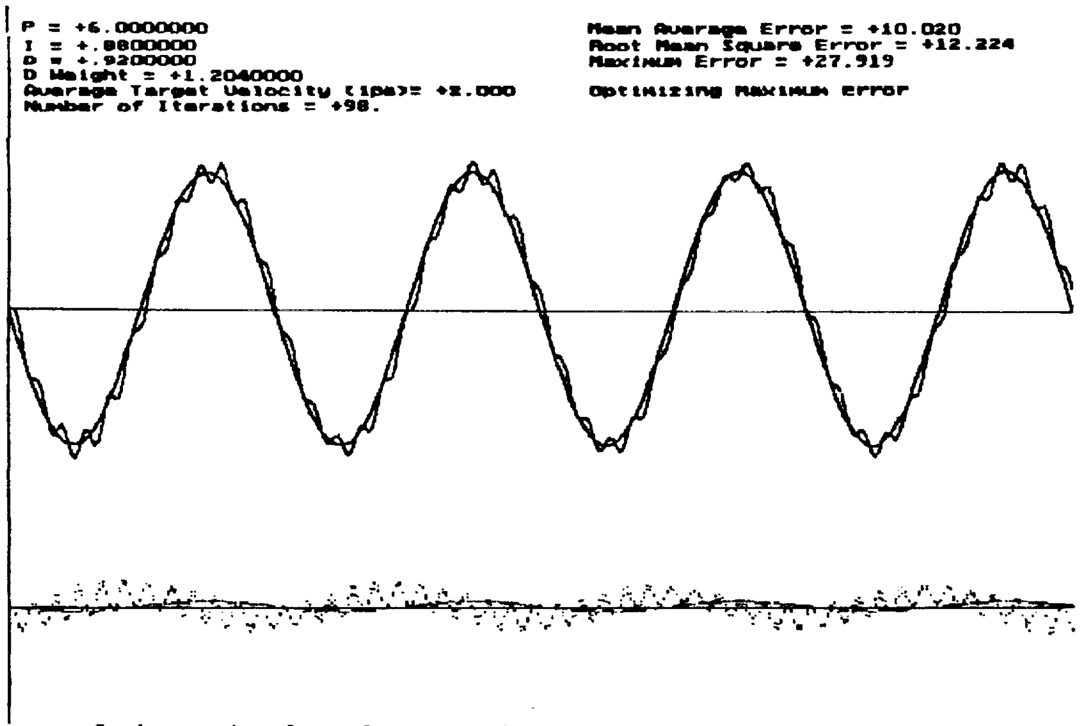
2.7 RESULTS OF SAMPLE TRIALS

After running several trials and comparing the results, several conclusions can be drawn.

Target Velocity is Significant. Changing the speed of target motion will change the resulting optimal gains. Figures 2-4 and 2-5 show the self-tuning results for average target speeds of 2 and 4 ips, respectively. Note also that the tracking errors are significantly worse with the target moving at 4 ips, almost twice as bad.

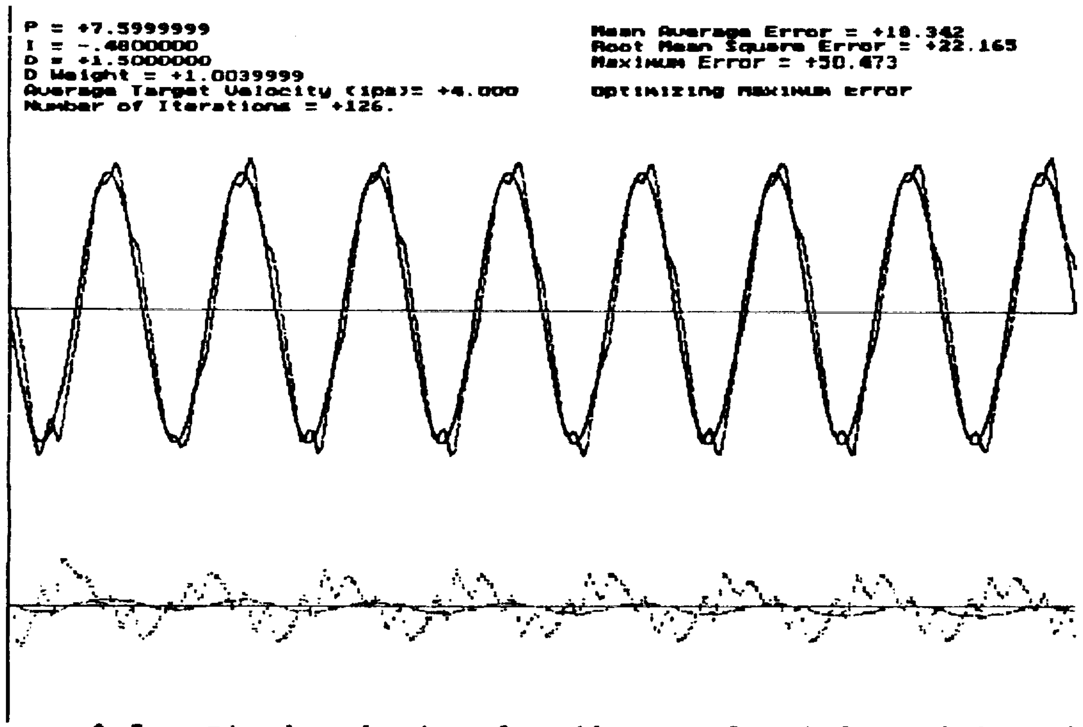
Length of Tracking Trials is Significant. The length of the tracking trials could significantly affect the selected gains. For short tracking trials the gains could be set such that the system is slightly unstable. This would not be noticeable unless longer trials were performed with the same gains. Figure 2-6 shows the solution when the gains were tuned using 48 second trials. If these same gains were specified in a trial that is 200 second as shown in Figure 2-7, the system became unstable after about 70 seconds.

Optimal Gains Could Be Negative. Figure 2-5 shows that the integral gain could be negative, depending on the tracking conditions. The parameter most affecting the negative gains is the time that the system tracks in each trial. For a system tracking for long periods of time, the gains should never be negative.



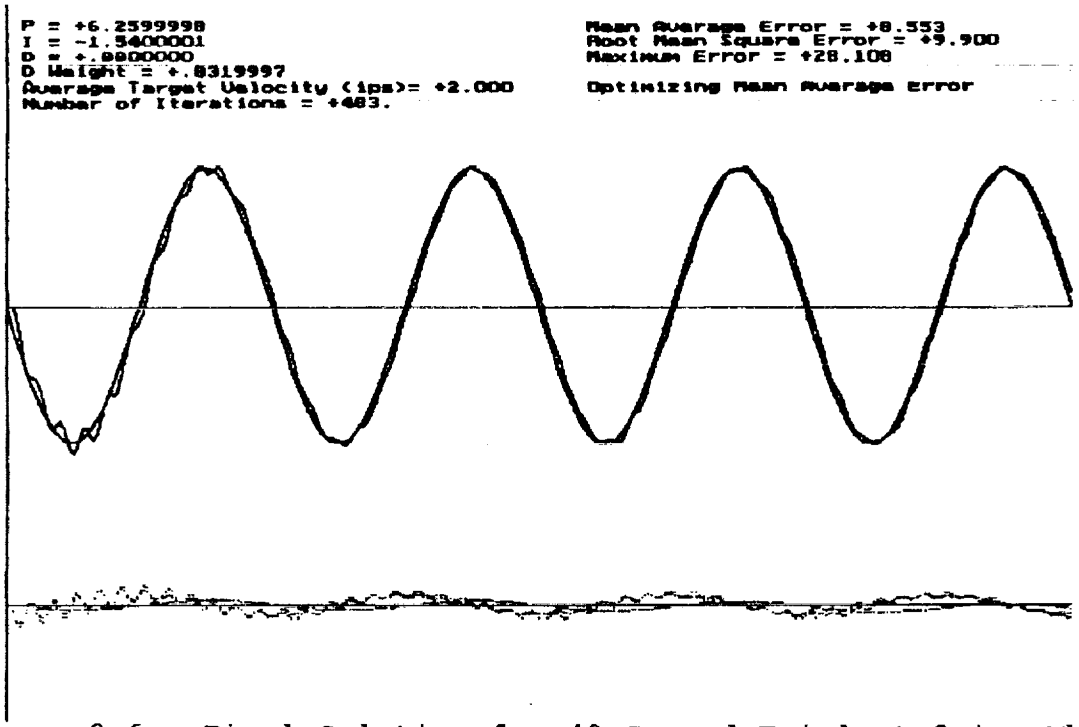
```
P = +6.0000000                                  Mean Average Error = +10.020
I = +.0800000                                   Root Mean Square Error = +12.224
D = +.9200000                                   Maximum Error = +27.919
D Weight = +1.2040000
Average Target Velocity (ips)= +2.000           Optimizing Maximum Error
Number of Iterations = +98.
```

Figure 2-4. Final Solution for 48 Second Trial at 2 ips When Optimizing Maximum Error with Start Up Bias



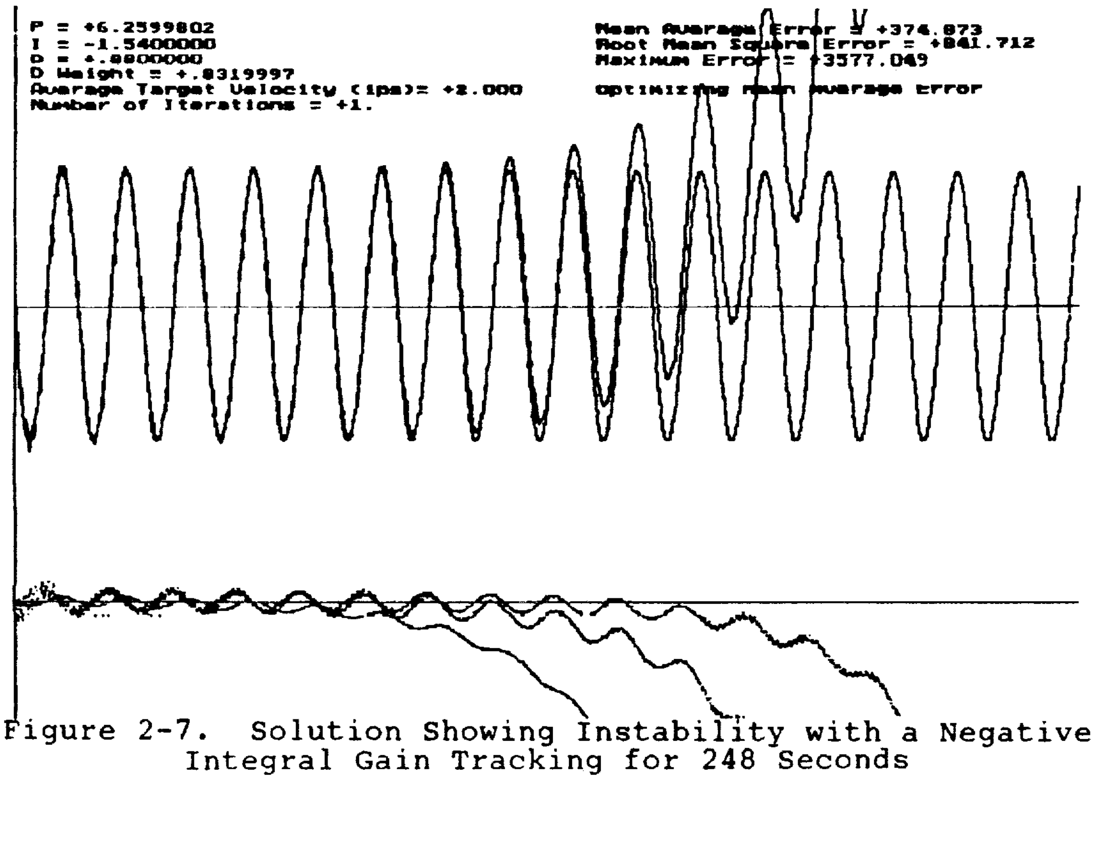
```
P = +7.5999999                                  Mean Average Error = +18.342
I = -.4800000                                   Root Mean Square Error = +22.165
D = +1.5000000                                  Maximum Error = +50.473
D Weight = +1.0039999
Average Target Velocity (ips)= +4.000           Optimizing Maximum Error
Number of Iterations = +126.
```

Figure 2-5. Final Solution for 48 Second Trial at 4 ips When Optimizing Maximum Error with Start Up Bias



```
P = +6.2599998                                    Mean Average Error = +8.553
I = -1.5400001                                    Root Mean Square Error = +9.900
D = +.8800000                                     Maximum Error = +28.108
D Weight = +.8319997
Average Target Velocity (ips)= +2.000             Optimizing Mean Average Error
Number of Iterations = +483.
```

Figure 2-6. Final Solution for 48 Second Trial at 2 ips When Optimizing Mean Average Error with Start Up Bias



```
P = +6.2599802                                    Mean Average Error = +374.873
I = -1.5400000                                    Root Mean Square Error = +841.712
D = +.8800000                                     Maximum Error = +3577.049
D Weight = +.8319997
Average Target Velocity (ips)= +2.000             Optimizing Mean Average Error
Number of Iterations = +1.
```

Figure 2-7. Solution Showing Instability with a Negative Integral Gain Tracking for 248 Seconds

ORIGINAL PAGE IS
OF POOR QUALITY

Start up Bias Exists. A start up bias exists which can significantly affect the gains. It is most apparent in trials which attempt to minimize maximum error. A large error occurs at the beginning of the trial while the system overcomes the lags. This single source of error quickly becomes the maximum error. In order to minimize maximum error, gains will be selected that reduce the start up bias at the expense of the rest of the tracking cycle. Figure 2-4 shows a trial with start up bias that tracks for 48 seconds. Figure 2-8 shows the same conditions except that tracking

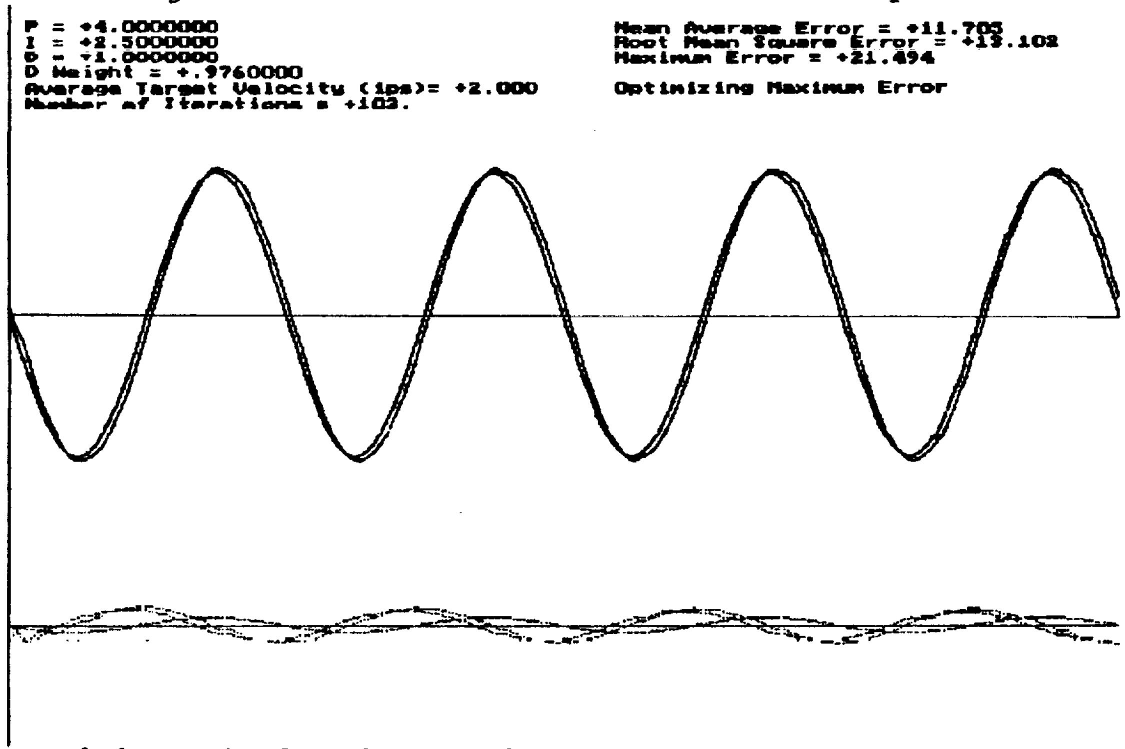


Figure 2-8. Final Solution for 248 Second Trial at 2 ips When Optimizing Maximum Error with No Start Up Bias

occurs for 248 seconds. The initial 200 seconds are not shown in the figure and errors are disregarded during this period. The results show a more stable system as the result of eliminating the start up bias. Measures using mean error are not affected as much by start up bias.

Optimization Criteria is Significant. The error criteria can play a significant part in the selection of gains. Figures 2-4 and 2-6 show the same trials except for a difference in error criteria. The difference will be more significant when the start up bias is not eliminated.

ORIGINAL PAGE IS
OF POOR QUALITY

III PREDICTIVE ALGORITHM

The use of a predictive algorithm is another attempt to track more accurately. If a certain type of motion is assumed, an algorithm can be formulated to predict where the target will be at the next robot move. However, if the motion significantly deviates from the assumption, the system will track poorly.

3.1 PREDICTIVE EQUATIONS

The equations used to predict the target position should be based on an assumed type of motion. This could include polynomial, exponential, or sinusoidal. Polynomials offer an added advantage of being quicker to calculate on the computer. In the case of polynomials, the predicted value is based on previous values. The number of previous values and the coefficients assigned to them can vary depending on the type of motion. However, unless there is a specific reason for doing otherwise, the coefficients should sum to 1. Examples of two equations derived by NASA and Boeing Aerospace Organization (BAO) are:

$$X_{n+1}=2X_n-X_{n-1} \tag{4}$$

$$X_{n+1}=\frac{5}{2}X_n-2X_{n-1}+\frac{1}{2}X_{n-2} \tag{5}$$

where:

X_{n+1} is the predicted value 1 period in the future
X_n is the value at the present time
X_{n-1} is the value 1 period in the past

Equations (4) and (5) should provide better response if the motion is linear and quadratic, respectively. Further derivations could be conducted for cubic motion. However, the increasing complexity of the calculations often does not provide proportional increases in prediction accuracy.

3.2 PREVIOUS ATTEMPT AT A PREDICTIVE ALGORITHM

A previous attempt was made to implement a predictive algorithm in the vision system. This approach did not use information about the robot or target position, velocity, and acceleration. Instead, it only used the error information contained in the vision system. Since the error information is dependent on both the target and robot trajectories, it is impossible to predict target motion without knowing something about the robot motion that produced the previous errors.

Furthermore, the problem is compounded because the error reference frame is continually shifting with the target and not constant with respect to the world or the robot. By choosing to implement the algorithms in a reference frame which is continually shifting, an important implicit assumption was made:

> If an error of "zero" occurs, the proper control strategy is being used and tracking <u>should continue</u> at the same velocity/acceleration to keep the robot on track. In more general terms it could be stated that any prediction about error that is made assumes that the current trajectory of the robot will continue. The predicted error is with respect to the current change in the position of the reference frame.

This can be illustrated with an example. Suppose the target is moving at a rate of 5 ips and the robot is tracking right on target with a velocity of 5 ips (highly unlikely because of lags in the system but okay for the purposes of illustration). Furthermore, it has been correctly tracking for the last several points so all of the past vision system errors are also "zero." With the current implementation of the predictive algorithm, the next predicted error would be "zero" and a command would be issued to the robot not to move during the next time frame. However, the target would continue to move and the robot would be left behind.

Based on the assumption listed above, what the robot should be doing when the error is "zero" is to continue moving in the same direction at the same velocity and acceleration. The reference frame must continue to move as it was moved when the previous errors were calculated. When the predicted error is "zero," it means it will be zero if the current trajectory is maintained. Therefore, a command should be issued to continue moving for the next segment at the same velocity in the same direction. In this example, the robot would be instructed to move at 5 ips and thus stay exactly on target.

In the more general case, the robot should move to a point which is the sum of the absolute position at the end of the current move plus the previous incremental move (to maintain the moving reference frame) plus the predicted error. A similar example could be presented for the general case where the predicted error is not "zero." However, to maintain brevity, it will be skipped at this time.

From this discussion it should be clear that there is a problem with the present implementation of the predictive algorithm. It could probably be modified to make it workable. However, it probably is not the best method of implementation.

3.3 PROPOSED IMPLEMENTATION OF THE PREDICTIVE ALGORITHM

There are several methods which could be used to properly implement a predictive algorithm. However, it is not clear which is the correct path to take for predictive algorithm implementation. It is not clear whether the prediction should be made in the MicroVax or in the vision system. Another important issue is where the results should be applied. Should an additional move component be calculated to sum to the robot position, or should the results of the predictive algorithm be fed into the PID loop for processing? Also, should the PID loop still remain in the tracking system? Since the PID loop is, in a manner of speaking, also doing some predictions, there is the possibility that the two will conflict. There are many questions which need to be answered. To gain some insight into these issues, the tracking model developed and presented in the previous sections could be altered. This would give the user some indication of the relative benefits of different methods.

Parallel to the implementation issues is the issue of calculating the prediction. The most straight forward approach is to determine the target path by combining the robot position with the vision system error. Thus, the predictive equations could be applied to positions as opposed to errors. If errors are still required as inputs to the PID loop, the calculated robot position at the end of the current move could be subtracted from the predicted target position to yield the predicted error. This should be the first attempt at implementation. With the PID loop intact, the controller will have the capability to speed up the response of the robot. However, the gains will probably require tuning because the error input will contain more information. If the system works properly, the gains should be able to increase, which will result in better response.

The predictive algorithm offers an opportunity for more accurate tracking. However, it must be reiterated that the predictions are based on an assumed type of target motion. If the target behaves in a random manner or a manner significantly different from the assumption, severely degraded performance could occur.

IV VISUAL CALIBRATION OF ORBITER POSITION FOR RADIATOR INSPECTION

4.1 RADIATOR INSPECTION ROBOT

NASA is currently in the process of design and construction of a robot to inspect the radiator panels on the orbiter. These panels are located on the inside of the cargo bay doors. They are inspected when the orbiter is pulled into the OPF and resting in the horizontal position. The cargo bay doors are opened to expose the radiator panels. Presently, the inspection occurs by workers in a bucket moving over the surface. The surface is divided into grids and defects are cataloged by location in the grid. Not all defects are repaired. Many are noted for future inspection to see if they are growing worse.

This is an ideal application for automated inspection. A robot is being constructed to move lengthwise beside the orbiter on a long track. It will be capable of inspecting the entire surface of the radiator panels. The system will be able to divide the radiator panels into smaller grids and thus provide better cataloging of defects.

The visual inspection system requires 1/8 inch accuracy in the X, Y, and Z direction. Therefore, the orientation of the orbiter will have to be determined with an accuracy better than 1/8 inch. For this part of the report, the task was to conceptually determine how the orientation could be performed visually, whether the Perceptics vision system in the RADL could perform the task, and what types of communication would be required between the robot controller and the vision processor.

4.2 PERCEPTICS VISION SYSTEM ROUTINES

The Perceptics vision system is a powerful high-level vision processor that runs in parallel with a MacIntosh computer. It can act as a color system if the appropriate hardware is included. Functions are available to snap pictures, perform thresholding, blob analysis, and other complex vision calculations [2].

The system was used in a project that identified wheat heads and moved the robot to point to them. While at first glance this seems much different than the project at hand, it really is very similar. A calculation must be performed to determine the position of the wheat head with respect to the robot. This is exactly the same task which must be used to determine the position of the orbiter.

Many routines were written in C to perform the vision tasks and robot move tasks. Of particular interest are the routines to determine the position of an object. There were two approaches to this problem. The first and most accurate was to use 4 fiducial points or dots arranged in a rectangle. By looking at these 4 points and using the vision system to measure the distance between them, the system could determine the orientation of the 4 points with 6 degrees of freedom. The accuracy achieved was about 1 mm which is well within the accuracy requirements. However, this method requires four dots to be laid out in a known rectangular pattern [3].

The second method used to locate objects was one of triangulation. Instead of a single picture as in the previous method, this method required two pictures to be taken a known distance apart. The same features must be located in each picture to determine the relative motion in the picture compare to the actual motion of the camera. This method provided accuracy that was barely acceptable in the axes of the vision system. In the third axis, the distance from the camera, the error exceeded that allowed for the system (5/32 inch error, 1/8 inch required accuracy). Better accuracy could be achieved by spreading the views apart. However, the previously described method would be preferred if implementable.

4.3 FEATURES TO USE FOR ORIENTATION

In order to determine the orientation of the orbiter, the position of 3 points on the surface must be known. In order to achieve maximum accuracy, the three points should be at the extreme ends of the object. On the orbiter, this might be three corners at the far extremes of the radiator panels. However, to achieve maximum accuracy the method using fiducial points is preferred. The corners of the panels do not offer this opportunity.

Another nearby feature is the set of bolts attaching the radiator panels to the cargo bay doors. In two of the corners, this is a 3 bolt pattern. In the other two corners, this is a 4 bolt pattern. By using these bolt holes and modified Perceptics vision system routines, the orbiter position could be calculated. A 3 bolt pattern will not provide a full 6 degree-of-freedom orientation like a 4 bolt pattern. However, it is not required in this case. By knowing the (X,Y,Z) position of 3 separate and known points on the orbiter, orientation of the orbiter can be calculated in 6 degrees-of-freedom.

4.4 PROPOSED SCENARIO

The proposed scenario to determine the orientation is:

1) Move to the first location. This could be either automatic or manual. The robot should be able to get close enough to perform the task automatically, although

initially manually controlled joystick motion might be preferred.

2) Take a picture and have the vision system send the appropriate coordinates with respect to the robot TCP.

3) The robot controller should calculate the location of the feature in OPF coordinates using the joint angles and kinematic transformations for the robot.

4) Perform tasks 1 to 3 for each of the remaining two features.

5) Determine the orientation of the orbiter using the robot controller. Pass any required information to the vision system performing the inspection.

4.5 INFORMATION EXCHANGE BETWEEN COMPUTERS

The communication link between the robot controller and the vision calibration system should require little information transfer. Since there are no processing speed requirements for the orientation task, a serial link should prove satisfactory. The robot controller should act as the master, issuing commands to the vision processor and waiting for responses.

The robot controller should be able to issue commands to the vision system to take pictures, process the pictures for specific features, and return the coordinates of the feature. The robot controller should also know the exact instant a picture is taken so that it can record its joint values. This will eliminate the problem of drift between the time the vision system takes the picture and the robot controller processes the data. However, it also requires that a single parallel signal be input to the robot controller to identify when a picture is taken. The vision system has no need to issue commands in this system.

V CONCLUSIONS

There were two distinctly different topics approached in this research project. The purpose of the first part of this research was to increase the tracking accuracy of a robotic tracking system. Two methods were explored: tuning the parameters in the system and implementing a predictive algorithm. A program was developed to self-tune the gains in the system by a heuristic "hill climbing" approach. The method implemented was the method of steepest ascent. Each of 4 gains in the system was incremented or decremented and the resulting reduction in error noted. The gain providing the best reduction in error was changed and the entire process was repeated until no further gain could be achieved.

To test this algorithm, a model of the tracking system was developed. This model included the lags inherent in the system due to buffering. It also modeled the acceleration rates of the robot controller. The model was used by itself and in conjunction with the self-tuning algorithm to better understand the tracking system. Several important conclusions were reported. The optimal gain values were found to be dependent on many factors in the system including the velocity of the target, the error criteria employed, the length of a tracking trial, and start up biases. The model was found to agree with most of the system responses observed by technicians and engineers over several years of trials.

Criteria for errors were discussed and although no trials have been performed, a measure was recommended for further study which included both the maximum absolute error and the mean absolute error. A possible implementation would be to use the maximum error as a constraint while minimizing the mean error.

Issues were raised and discussed in the implementation of a predictive algorithm. A previous attempt was discussed and the error was pointed out. It was proposed that the target position should be determined by summing the robot TCP position with the vision system error recorded at the same time. Questions were raised about where the predictive algorithm should be located and whether the PID loop would still be required. A recommendation was made to add these capabilities to the model to obtain some knowledge about the decisions which are required.

In the second topic of this report, the use of a vision system to determine the orientation of the orbiter in the OPF was discussed. The task of radiator inspection was defined. Next, the use of the Perceptics vision system located in the RADL was explored. Next, a scenario was proposed to determine the orientation. Finally, the communication between the robot controller and the vision system was discussed.

VI REFERENCES

[1] Davis, Virgil Leon, "Systems Integration for the Kennedy Space Center Robotics Applications Development Laboratory," MS87-482 SME Technical Report, 1987.

[2] User's Manual for the NuVision Image Processing Workstation, Perceptics Corporation, Knoxville, TN, 1989.

[3] Myjak, Michael; Sklar, Mike; Tharpe, Roy; Thomas, Mark; and Wegerif, Dan, Automated Plant Growth Development Project: FY89 Final Report, Advanced Technologies Branch of McDonnell Douglas Space Systems Company, Kennedy Space Center, 1990.