ON-LINE COLLISION AVOIDANCE FOR THE RANGER TELEROBOTIC FLIGHT EXPERIMENT

Bruce Bon Homayoun Seraji

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA '31109, USA

Abstract

This paper describes the approach and algorithms which have been developed for model-based *whole-arm* collision *avoidance for the NASA Ranger Telerobotic Flight Experiment.* Minimum distances *and nearest* points *between parts of the active dexterous manipulator and* potential *obstacles are computed by the Ranger obstacle detection software (described* in *a* companion *paper [1]). Obstacle data is then used to compute virtual repulsive forces* which *perturb the operator-commanded* manipulator Cartesian motions in *order* to avoid *collisions. The virtual forces* are *computed at the tool-tip for end-effector* position *perturbations, at the wrist. for end-effector orientation perturbations, and* on *the upper and lower arm* links *for arm angle perturbations. The* paper *describes the software development environment,* utilizing a *3-D graphical simulation and* providing a *graphical user interface for display and operator control. Results of several test runs, illustrating position, orientation and arm angle collision avoidance, are presented.*

1 Introduction

The NASA Ranger Telerobotic Flight Experiment [2, 3, 4, 5], led by the University of Maryland, is aimed at the development and demonstration of robotic technologies for executing manipulation tasks in space. Ranger incorporates two dexterous seven degree-of-freedom manipulator arms mounted on a free-flying base. These arms will be used, both individually and cooperatively, to perform a variety of manipulation experiments and servicing operations (Figure 1). The Ranger dexterous arms will be controlled using the Configuration Control approach developed at JPL [6, 7]. In this approach, the basic task of end-effector position and orientation control is augmented by additional user-defined tasks. For the Ranger implementation, the additional task is defined to be the control of the *arm angle*, which is the angle between the plane of the shoulder, elbow and wrist joints, and a reference plane through the shoulder and wrist joints and containing a "vertical" reference vector [8].

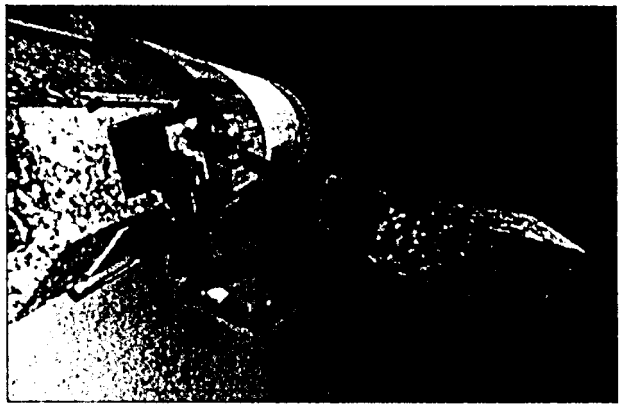

Figure 1: Ranger performing an on-orbit experiment

The Ranger baseline arm control system has no provisions for obstacle detection and collision avoidance. Therefore, erroneous operator commands can cause collision between the dexterous arms and the camera and grapple arms, the base, or the task board. Automated obstacle detection and collision avoidance will enable safe, collision-free motions of the arms throughout the workspace. It will also cause a reduction in the Ranger operation time, since possible motions with potential collisions will not be executed. This capability will increase the safety of the Ranger during the operation of the arms, a feature which is vital to the success of the Ranger mission.

Real-time collision avoidance has received considerable attention in recent years [9, 10, 11, 12], including a sensor-based position control system developed at JPL [13, 14]. The software described in this paper utilizes

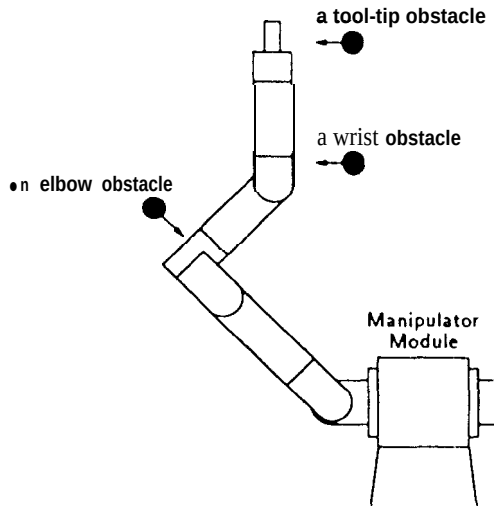

Figure 2: Top view of Ranger manipulator module and left dexterous manipulator in the home pose

the principles of the earlier JPL work, extending them to control six degrees-of-freedom of end-effecter Cartesian position anti orientation plus arm angle and utilizing model-based obstacle detection computations in place of sensor-based measurements.

Section 2 presents the virtual force formulation that is used to perturb position, orientation and arm angle for collision avoidance. Section 3 briefly describes the graphical user interface developed to achieve collision avoidance during test and demonstration. Section 4 presents the results of simulation runs to test avoidance perturbation in position, orientation and arm angle. Finally, Section 5 draws conclusions from this work.

2 Virtual Forces and Avoidance Perturbations

As shown in Figure 2, each dexterous arm has 7 joints and .3 links or arm segments, Obstacle detection provides data on the nearest potential obstacle in each of the following three zones: tool-tip, wrist and elbow. A tool-tip obstacle is one whose nearest point on the dexterous arm is on the tool link and is closer than a specified threshold to the tool-tip, A wrist obstacle is One whose nearest point on the dexterous arm is on the tool link and is further away than the specified threshold from the tool-tip. An elbow obstacle is one whose nearest point on the dexterous arm is either on the upper-arm link or on the lower-arm link.

The collision avoidance software computes the virtual force corresponding to the single nearest obstacle relative to each arm zone, thus limiting perturbation

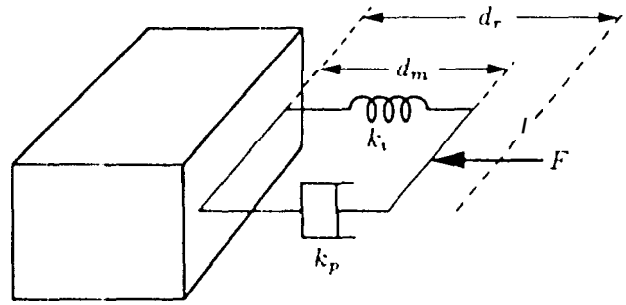

Figure 3: Spring and damper models used for virtual force generation

computations to no more than three obstacles during any iteration. A single' point obstacle near the tool link will perturb either the tool-tip position or the tool-tip orientation, depending on its distance from the tool-tip, while an extended obstacle may be detected as two point obstacles, one in the tool-tip zone and one in the wrist zone, thus perturbing both position and orientation. These collision avoidance policies are adopted for simplicity of concept, ease of implernentation, anti, most importantly, to minimize computational requirements while providing robust performance.

2.1 Virtual Spring and Damper Forces

For each of the three detection zones, collision avoidance generates a virtual force, denoted by a vector in Figure 2, to repel the arm away from the obstacle. This force is zero if there is no object within a user-specified *stand-off* distance, d_r, away from the surface of the arm. If the minimum distance to an obstacle, d_m, is iess than the stand-off distance, we say that the arm is within the obstacle's *avoidance zone* anti a virtual force, F is generated, comprised of a spring component, which is proportional to the incursion magnitude, and a damper component, which is proportional to the closing velocity between the arm and the obstacle (see Figure 3).

Tool-t ip virtual forces perturb the end-effector position coordinates to avoid collisions. Wrist virtual forces perturb the end-effector orientation coordinates to avoid collisions. Elbow virtual forces perturb the arm angle to avoid collisions.

Collision avoidance is accomplished by utilizing the virtual spring and damper forms to modify the operator-commanded arm motions. Entry into an obstacle's avoidance zone will be opposed by the virtual force, which is a function of the avoidance zone *incursion*, $c = d_r - d_m$. Let the virtual spring constant be denoted by k_s and the virtual damper constant by k_p.

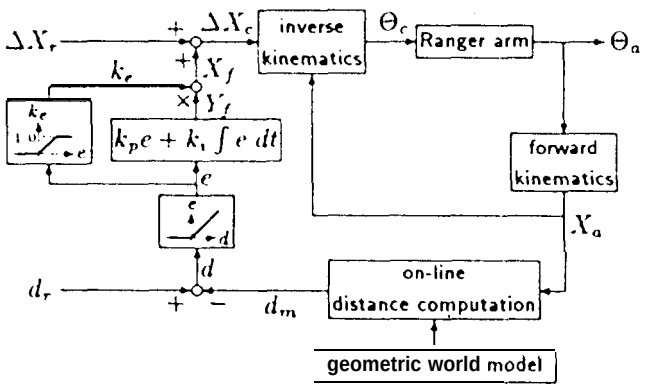

Figure 4: Control diagram for perturbing arm motion for collision avoidance

Then the virtual force for collision avoidance is:

$$F(t) = k_s e + k_p \frac{de}{dt} \quad (1)$$

where the sign of e (and, hence, of F) is chosen such that the force F will be applied in the opposite direction from the incursion. The terms in equation (1) correspond to a virtual spring force, $F_s(t) = k_s e$, and a virtual damper force, $F_d(t) = k_p\ de/dt$.

2.2 Hand Position Perturbations

The virtual force, F, can be considered as a velocity perturbation, v_f. Figure 4 illustrates how the perturbation is generated and used to modify the operator-commanded position, The raw position perturbation. Y_f, is given by:

$$Y_f = \int v_f\ dt = \int F(t)\ dt \quad (2)$$

The raw spring and damper position perturbations are given by:

$$Y_{fs}(t) = k_s \int_a^t e\ dt \quad (3)$$

$$Y_{fd}(t) = k_p\ e(t) \quad (4)$$

where a is the last time at which $e(t) = 0$. Numerically integrating using the trapezoidal rule yields:

$$Y_{fs}(t_i) = Y_{fs}(t_{i-1}) + \frac{1}{2} k_s \left[e(t_{i-1}) + e(t_i) \right] \Delta t \quad (5)$$

$$Y_{fd}(t_i) = k_p \left[e(t_i) - e(t_{i-1}) \right] \quad (6)$$

where $Y_{fs}(t_0) = 0$, $Y_{fd}(t_0) = 0$, and $\Delta t = t_i - t_{i-1}$

If used without modification, the raw position perturbation, Y_f, can cause stability problems at the boundary of the avoidance zone. This is caused by the abrupt zeroing of the perturbation when the arm leaves the avoidance zone, resulting in discontinuity in both velocity and position. [10] avoid this instability, the raw spring position perturbation is scaled to yield the final position perturbation, X_f. The raw damper perturbation is not scaled because it approaches zero over time and hence does not cause an exit discontinuity,

As shown in the far-left box in Figure -i, 'a scaling factor k_e is introduced as follows:

$$k_e(e) = \begin{cases} 0 & \text{if arm is outside of avoidance zone} \\ k_e / d_{k_e} & \text{if } 0 < e \le d_{k_e} \\ 1 & \text{if } e > d_{k_e} \end{cases} \quad (7)$$

where d_{k_e} is the value of e at which the full value of the raw spring perturbation, Y_{fs}, is applied. Multiplying the raw spring perturbation by k_e ensures that the effective perturb: tion, X_f, will never exceed the incursion, e, and hence prevents a discontinuity in commanded position that would otherwise occur when the arm exits the avoidance zone.[1] Thus the final, modified position perturbation for iteration i is computed from:

$$X_f(t_i) = k_e(e)\ Y_{fs}(t_i) + Y_{fs}(t_i) \quad (8)$$

where Y_{fs} and Y_{fd} are computed using equations (5)-(6). Finally, the perturbation $X_f(t_i)$ is applied by adding it to the current operator-commanded change in position Δx.

2.3 Hand Orientation Perturbations

Figure 5 illustrates one viable approach to perturbing the orientation of link WT, where W is the wrist and T is the tool-tip. Let 1' be the closest point on an obstacle to WT and Q be the closest point on WT to the obstacle. The objective is to leave the position of T unperturbed, but to rotate WT about T such that point Q will move away from the obstacle to zero out the virtual force $\vec{F}$. Following the derivation in 2.2, the desired displacement of point Q is $\vec{Q}_f$:

$$\vec{Q}_f = k_e(e) \int \vec{F}_{fs}\ dt + \int \vec{F}_{fd}\ dt \quad (9)$$

where $k_e(e)$ is defined by equation (7) and $\vec{F}_f = \vec{F}_{fs} + \vec{F}_{fd}$.

Let $\vec{r} = Q - T$ be the vector from T to Q, $\hat{r}$ the unit vector of $\vec{r}$ and r the length or $\vec{r}$. If the geometry

[1] For simplicity within Figure 4, k_e is shown as scaling the total raw perturbation, Y_f, but in actuality it is only used to scale the raw spring perturbation, Y_{fs}.

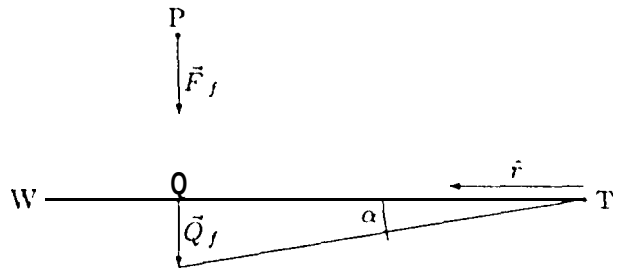

Figure 5: Example geometry for orientation perturbation

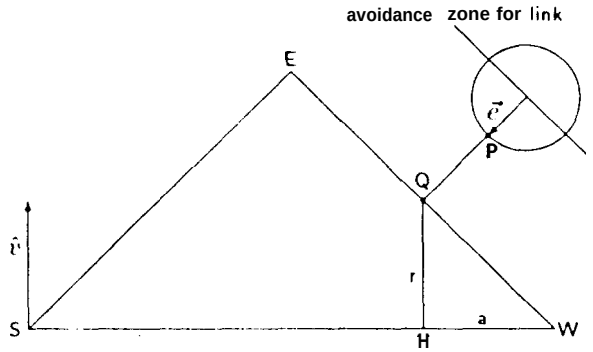

Figure 6: Shoulder-elbow-wrist geometry for collision avoidance

is known to be planar with $\vec{F}_f$ perpendicular to WT (as shown), then the desired orientation perturbation will be a rotation about T by the angle α, where:

$$\alpha = |\vec{Q}_f| \ / \ r \qquad (10)$$

In order to generalize for a rotation about an arbitrary axis through T, define the angular rotation perturbation vector, $\vec{R}_f$, to be a vector parallel to the desired axis of rotation whose magnitude is the desired angle of rotation and whose direction determines the direction of rotation by the right-handed rule. ? Then the rotation perturbation corresponding to the desired displacement perturbation, $\vec{Q}_f$, is given by:

$$\vec{R}_f = \left(\hat{r} \times \vec{Q}_f\right) \ / \ r \qquad 11)$$

or:

$$\vec{R}_f = \frac{\vec{r}}{\vec{r} \cdot \vec{r}} \times \vec{Q}_f \qquad 12)$$

Substituting equation (9) into equation (12), we obtain:

$$\vec{R}_f = \frac{\vec{r}}{\vec{r} \cdot \vec{r}} \times \left(k_r(c) \int \vec{F}_{fs} dt + \int \vec{F}_{fd}\, dt\right) \qquad (13)$$

The preceding analysis is valid providing that $\vec{r}$ is constant. However, considering that $\vec{r}$ varies over time as either the obstacle or the arm link changes position, we should bring the $\dot{r}$ term within the integrals:

$$\vec{R}_f = k_r(c) \int \frac{\vec{r}}{\vec{r} \cdot \vec{r}} \times \vec{F}_{fs} dt + \int \frac{\vec{r}}{\vec{r} \cdot \vec{r}} \times \vec{F}_{fd} dt \quad (14)$$

Substituting the definitions of $\vec{F}_{fs}$ and $\vec{F}_{fd}$ into equation (14):

$$\vec{R}_f = \int k_r(c) k_s \frac{\vec{r}}{\vec{r} \cdot \vec{r}} \times \vec{r}\, dt + \int k_d \frac{\vec{r} \times \dot{\vec{r}}}{\vec{r} \cdot \vec{r}} \times \frac{d\vec{c}}{dt} dt \quad (15)$$

[2]The components of $\vec{R}_f$ comprise roll, pitch and yaw for a fixed-angle rotation in space

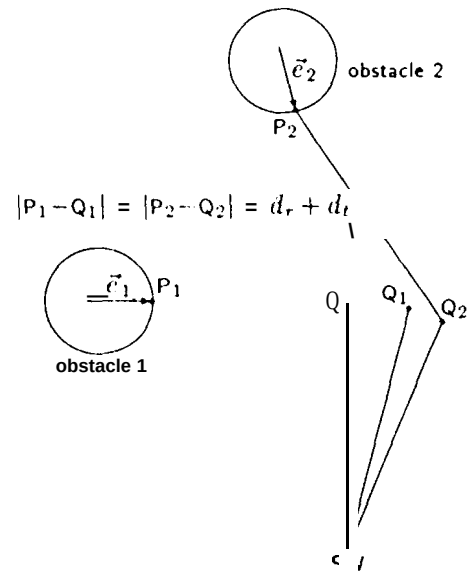

Figure 7: Edge-on view of shoulder-elbow-wrist geometry

Then a similar analysis to that presented in section 2.2 can be used to get iterative equations for computing the angular rotation perturbation, $\vec{R}_f$. In order to perturb the commanded motion in hand orientation, $\vec{R}_f$ is converted into a 3x3 rotation matrix, which is then multiplied by the 3x3 matrix representing the desired motion in hand orientation to get the achieved motion in hand orientation.

2.4 Arm Angle Perturbations

Potential collisions with the Ranger upper-arm and lower-arm links are avoided by perturbing the arm angle. Figure 6 illustrates the basic geometry, where points S, E and W are the shoulder, elbow and wrist center points, respectively, and $\hat{v}$ is a unit vector which, with the line SW, defines the reference plane. The arm

angle, ϕ, is measured from this reference plane to the arm plane containing S, E and W.

The point, P, is the nearest point, on the surface of any obstacle. to the surface of the upper-arm link. whose axis is SE, or the lower-arm link, whose axis is EW. The point, Q, is the point on axis SE or axis EW that is closest to P. The error, $\vec{e}$, is the vector whose direction is from P to Q and whose magnitude is the distance of incursion of point P on the surface of a collision object into the avoidance zone defined by the tool-link radius d_t and the stand-off distance d_r. Then:

$$|\vec{e}| = (d_t + d_r) - |Q - P| \tag{16}$$

and $\vec{e}$ is in the direction of (Q - P), i.e. from P to Q.[3]

Figure 7 illustrates an edge-on view of the shoulder-elbow-wrist geometry with two example obstacles. Both obstacles are the same distance from the link nearest point Q. The virtual force is always applied in a direction perpendicular to the SEW plane. Once the arm angle is perturbed to nullify the incursion into the avoidance zone, the link nearest point will be at Q_1 for obstacle 1 or Q_2 for obstacle 2. The arm angle must be perturbed more for obstacle 2, which is close to edge-on to the SEW plane, than for obstacle 1, whose incursion vector is perpendicular to the SEW plane, even though the component of the incursion vector $\vec{e}$ in the direction of required motion of Q is less for obstacle 2. The virtual force used for arm angle perturbation is a function of the incursion magnitude $|\vec{e}|$ to assure that edge-on obstacles such as obstacle 2 are successfully avoided.

The signed magnitude of $\vec{e}$, denoted e_m, is the basis for computing the arm angle perturbation. The sign of e_m is chosen to perturb Q away from the obstacle. If the virtual spring constant is k_s, and the virtual damper constant is k_p, then the scalar virtual force for collision avoidance will be:

$$F_f = k_s e_m + k_p \frac{de_m}{dt} \tag{17}$$

where the signs are positive to indicate a force in the direction opposing the incursion,

Let r be the length of line QH, the radius of revolution for the arm angle when the position of Q is perturbed. We need to find the signed scalar angular perturbation, ϕ_f, to arm angle ϕ which will nullify the avoidance zone incursion. Replacing R_f with ϕ_f in equation (15), and simplifying to scalars, we get the following equation for ϕ_f:

$$\phi_f = k_e(e_m)\, k_s \int \frac{1}{r}\, e_m\, dt + k_p \int \frac{1}{r} \frac{de_m}{dt}\, dt \tag{18}$$

[3]For ease of visualization, the avoidance zone in Figure 6 is shown surrounding the arm link rather than the obstacle - this is an equivalent model to the usual one, where the avoidance zone surrounds each obstacle

In order to numerically compute the arm angle perturbation. we compute $e'_m \equiv de_m/dt$:

$$e'_m(t_i) = \left[e_m(t_i) - e_m(t_{i-1})\right] / \Delta t \tag{19}$$

where $\Delta t = t_i - t_{i-1}$. Then, using the trapezoidal integration rule with $Y_{fs}(t_0) = 0$ and $Y_{fd}(t_0) = 0$ yields:

$$Y_{fs}(t_i) = Y_{fs}(t_{i-1}) + \frac{1}{2} k_s \left[\frac{e_m(t_{i-1})}{r(t_{i-1})} + \frac{e_m(t_i)}{r(t_i)}\right] dt \tag{20}$$

$$Y_{fd}(t_i) = Y_{fd}(t_{i-1}) + \frac{1}{2} k_p \left[\frac{e'_m(t_{i-1})}{r(t_{i-1})} + \frac{e'_m(t_i)}{r(t_i)}\right] dt \tag{21}$$

$$\phi_f(t_i) = k_e(e_m)\; Y_{fs}(t_i) + Y_{fs}(t_i) \tag{22}$$

where $k_e(e_m)$ is defined by equation (7) and serves, as discussed in section 2.2, to prevent disco ntinuities as the arm moves out of the avoidance zone.

3 Graphical User Interface

The software package for obstacle detection and collision avoidance is implemented in C 011 an SGI Indy under IRIX, but is designed to be portable for integration into the Ranger flight software running on a MIPS R3000 processor under VxWorks. A test and demonstration program provides a graphical user interface (GUI) for operator control and drives a 3-D graphical simulation provided by the Ranger project. Capabilities of this program are summarized below.

The 3-D graphics simulation displays the Ranger neutral buoyancy vehicle (NBV) and the two dexterous arms. The nearest obstacle to the currently active arm is identified by a colored line in the 3-D scene connecting the obstacle and the arm link that is closest to it. If the obstacle is within the detection threshold distance of the arm, the nearest arm link changes color to red and the connecting line changes from yellow to red. When collision avoidance is not active, commanding an arm into a collision with itself, the other arm or the spacecraft will result in the arm moving *inside* the obstacle, When collision avoidance is active, the arm will move as commanded until it bits the invisible avoidance zone and will then slide along the avoidance zone boundary until it is as close as possible to the goal state and the command runs out of time.

The GUI provides a main control panel with buttons to select the active arm, to bring up additional control panels for joint or Cartesian control. to turn obstacle

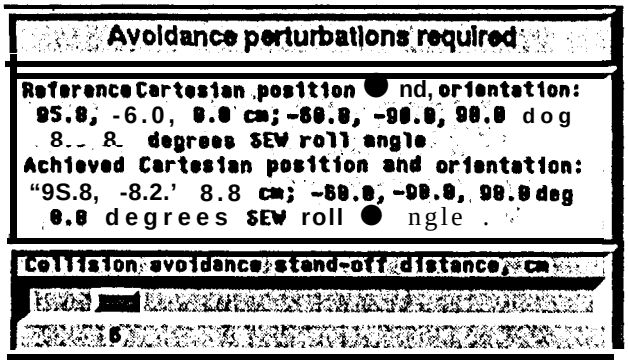

Figure 8: The JPL Ranger graphical user interface panel for avoidance

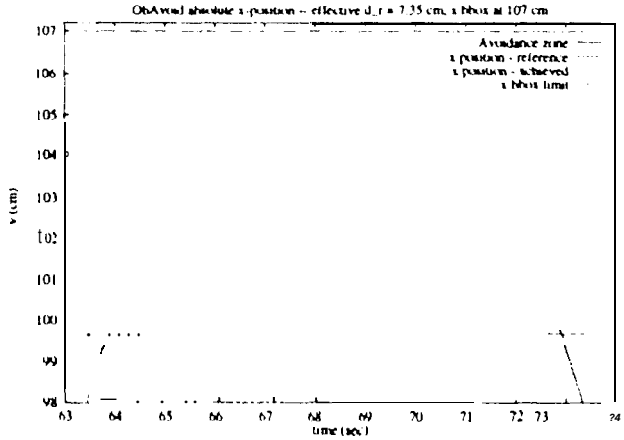

Figure 9: Position perturbation example: reference and achieved x trajectories

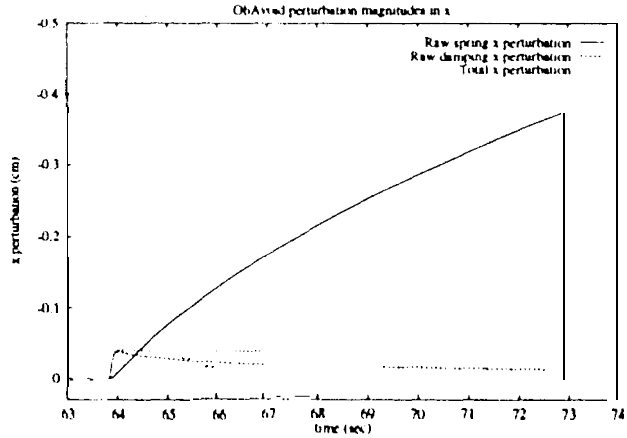

Figure 10: Position perturbation example: collision avoidance perturbations

detection and collision avoidance eon or off, to bring up a button panel for controlling tests and viewpoints, and to terminate execution. The Cartesian control pane] has sliders for operator control of the location and orientation of the end-effector of the currently active arm, as well as the arm angle. The Test Panel is comprised of buttons that allow the user to select from ten locations for viewing the simulation, as well as to setup and execute any of ten test and demonstration cases. Whenever detection or avoidance are active, the main control panel is expanded with section sfordat adisplay and control of these features.

Figure 8 is a screen snap.shotofthe GUI panel which controls the stand-off distance d_r and displays current avoidance data. The top-most label widget simply indicates whether or not avoid ance perturbations are currently required. The middle widget displays the reference and achieved Cartesian coordinates and artn angle (aka *SEW roll angle*). The slider along the bottom provides user control of the stand-off distance,

4 Simulation Results

Several tests are conducted on collision avoidance of the Ranger dexterous arms. We shall now present a typical set of results obtained in these tests. The obstacles used in these tests are *bounding box limits*. The *bounding box* is defined to be a virtual box enclosing the arms ●n(i centered on the Ranger manipulator module. Whena manipulator approachesany side of the bounding box, an obstacle is detected.

Figure 9 shows the results of a simulation run using the Ranger test program and the collision avoidance software to demonstrate hand position avoidance. For this run, the x component of end-effector position is commanded toward a bounding box limit obstacle at $x = 10\bar{7}$ cm. The avoidance zone boundary, taking into account the radius of the link, is at $x = 99.65$ cm, and the *reference* trajectory is the trajectory whir-h would have been followed if collision avoidance were not active. As shown in the Figure, the *achieved* trajectory enters the avoidance zone by less than 0.5 cm before settling down to an avoidance zone incursion of less than 0.1 cm. The shape of the trajectory and the magnitude of avoidance zone incursion can be adjusted by modifying the parameters k_s and k_p.

Figure 10 shows the perturbations used to modify [he trajectory in Figure 9. As shown , the raw spring perturbation continues 10 increase over time, while the r a w damper perturbation decreases toward zero. Because the incursion is always less than the spring per - turbation scaling parameter, d_{k_s}, the scaling factor, k_s, reduces the effect of the raw spring perturbation. The net result is a total perturbation which approaches the

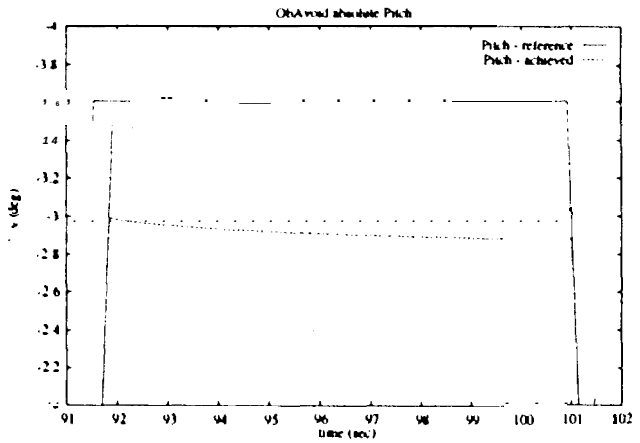

Figure 11: Orientation perturbation example: reference and achieved pitch trajectories

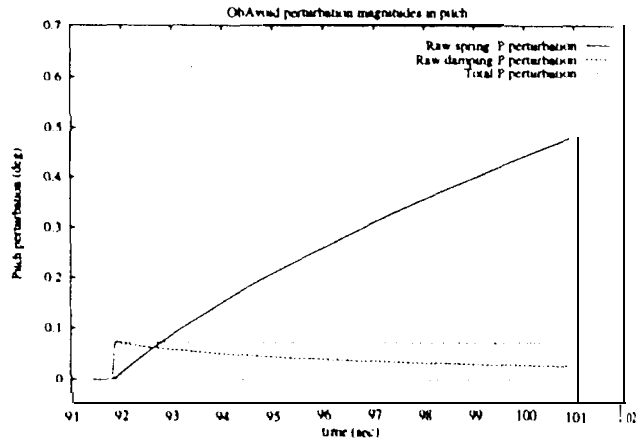

Figure12: Orientation perturbation example: collision avoidance perturbations in pitch

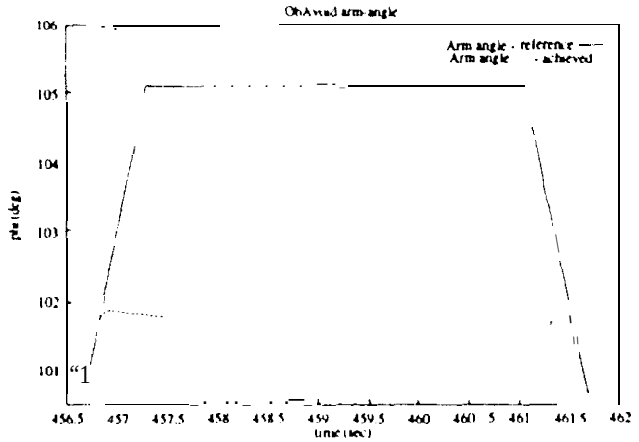

Figure 13: Arm angle perturbation example: reference and achieved ϕ trajec tories

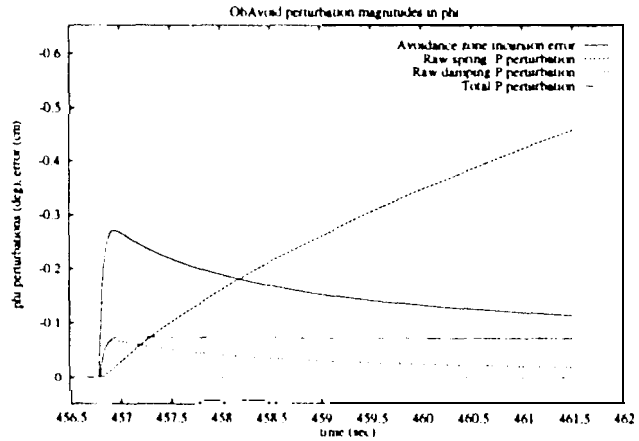

Figure 14: Arm angle perturbation example: collision avoidance perturbations in ϕ

value corresponding to the maximum move allowed per iteration, or 0.04 cm.

Figure 11 shows the results of a simulation run demonstrating hand orient ation collision avoidance, and Figure 12 shows t be corresponding avoidance per- t urbations. The obstacle is a positive-z bounding box limit at 10 cm, resulting in an avoidance zone boundary at 3.65 cm. For the particular configuration of the manipulat or arm used in the test, this avoidance zone boundary corresponds to a limit of about '2.9 degrees in pitch. The reference pitch trajectory would have been achieved if avoidance had been turned off. When the reference pitch trajectory moves out of the avoidance zone, t he achieved t rajectory again coincided wit h t he reference. Note that the perturbations shown in Figure 12 are virtually identical in form with those of of figure 10, showing position perturbations.

Figure 13 shows the results of a simulation run to demonst r ate arm angle collision avoidance. Figure 14 shows the corresponding avoidance perturbations and the avoidance zone incursion error $|\vec{e}|$ used for compu - ing the perturbations. The maximum incursion into the avoidance zone is approximately 0.28 cm. The avoic - ance zone is such that ϕ is limited to 101 .6 degrees. The reference trajectory is the trajectory that ϕ would have achieved had avoidance been turned off. When the reference ϕ trajectory moves out of the avoidance zone, the achieved trajectory again coincides with the reference. Generally, t he avoidance results for this run are very similar to the end-effector test run results discussed above.

Perturbation computation times per iteration, not including the time required for obstacle detection or forward and inverse kinematics computations, are mea-

sured to be less than 0.2 msec on a MIPS R4600PC processor running at 100 MHz. Total computation time. including obstacle detection and forward and inverse kinematics computations, are measured to be about 2.5 msec. The Ranger flight computer is expected to be a MIPS R3000 processor, which may take 2-4 times as long for these computations.

5 Conclusions

A complete on-line collision avoidance software package for the Ranger Telerobotic Flight Experiment has been presented. All algorithms described shove are documented in detail **[15],** implemented and extensively tested in a simulation environment, using, geometric models corresponding to the latest design of the flight hardware. Tests show the software to be very effective in computing perturbations to avoid collisions, while maintaining smooth manipulator trajectories. Performance is fast enough that the perturbation computation can easily run within a real-time control loop to provide continuous on-line collision avoidance for Ranger operations.

Acknowledgements

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (the Code X Telerobotics Program). The 3-D graphical simulation program was provided by the Space Systems Laboratory of the University of Maryland.

References

[1] B. Bon and H. Seraji: "Obstacle detection for the Ranger Telerobotic Flight Experiment," submitted to International Robotics and Automation Conference, 1996.

[2] D. Akin and R. Howard, "The ml] of free-flight in space telerobotic operations," International Con - ference on Automation and Robotics, 1993.

[3] D. Akin and P. Churchill. "Robotics architectures for telerobotics flight operations." A IAA Space Programs and Tech nologies Con feren ce, Huntsville, September 1993.

[4] C. Carignan, "A decoupling inverse kinematics algorithm for a seven joint manipulator," SSL publication, [university of Maryland, 1992.

[5] J. Graves, "Ranger Telerobotic Flight Experiment program update," Proc. Conference on Space Manufacturing, Princeton, May 1995.

[6] H. Seraji: "onfiguration control of redundant manipulators: theory and implementation," IEEE Trans. on Robotics and Automation, Vol. 5, No. 4, pp. 472-490, 1989.

[7 H. Seraji, M. Long and T. Lee: "Motion control of 7-DOF arms: the configuration control approach," IEEE Trans. on Robotics and Automation, Vol. 9, No. 2, pp. 125-139, 1'393.

[8] K. Kreutz-Delgado, M. Long, and H. Seraji: "Kinematic analysis of 7-DOF manipulators," International Journal of Robotics Research, Vol. 11, No. 5, pp. 469--181, 1992.

[9] E. Cheung and V. Lumelsky: "Proximity sensing in robot manipulator motion planning: system and implementation issues, " IEEE Trans. on Robotics and Automation, pp. 740-751, 1989.

[10] C. Boddy and J. Taylor: "Whole-arm reactive collision avoidance control of cinematically redundant manipulators, " Proc. IEEE Int. Conf. on Robotics and Automation, pp. 382-38?, Atlanta. May 1993.

11] J. Feddema and J. Novak: "Whole arm obstacle avoidance for teleoperated robots, " Proc. IEEE Int. Conf. on Robotics and Automation, pp. 3303-3309, San Diego, May 1994.

12] H. Ding and W. Schiehlen: "On controlling robots with redundancy in an environment with obstacles," Proc. Symp. on Robot Control, pp. 771-776, Capri, Italy, September 1994.

[13] H. Seraji: "Sensor-based control of manipulators: theory and experiments," submitted for publication, 1995.

[14] H. Seraji, R. Steele and R. Ivlev: "Experiments in sensor-based collision avoidance," submitted for publication, 1995.

[15] B. Bon, "Distance servoing and collision avoidance algorithms for Ranger." internal JPL document, July 1995.