1993 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

CONTROL OF A SERPENTINE MANIPULATOR WITH COLLISION AVOIDANCE

PREPARED BY: Dr. Robert M. Byers

ACADEMIC RANK: Assistant Professor

UNIVERSITY AND DEPARTMENT: University of Central Florida
Department of Mechanical and
Aerospace Engineering

NASA/KSC

DIVISION: Mechanical Engineering

BRANCH: Special Projects

NASA COLLEAGUE: Bill Jones
Gabor Tamasi
Todd Graham

DATE: August 10, 1993

CONTRACT NUMBER: University of Central Florida
NASA-NGT-60002 Supplement: 11

ACKNOWLEDGMENTS

The author gratefully acknowledges the support of the University of Central Florida and the NASA personnel who made the Summer Faculty Fellowship Program a productive and enjoyable experience. Especially noteworthy are the efforts of Dr. Ray Hosler and Kari Stiles, who rode herd on the Faculty Fellows and kept everything running smoothly. In addition, I wish to thank Gabor Tamasi, Bill Jones, and Todd Graham for their encouragement and patience. I look forward to working with them in the future.

ABSTRACT

The robotics lab at the Kennedy Space Center is investigating the possibility of using a "serpentine" manipulator for Shuttle inspection and payload processing. Serpentine manipulators are characterized by a large number of degrees of freedom giving them a high degree of redundancy. This redundancy allows them to be used to reach confined areas while avoiding collisions with their environment. In this paper, the author describes a new approach to controlling the joint rates for an n degree of freedom robot such that it moves its end effector to a desired position while simultaneously avoiding collision of any part of the robot arm with obstacles. Joint rates which move the end effector toward the target are found via a Lyapunov stability function. The gradient of an obstacle cost function indicates the direction toward obstacle collision in the joint space. The component of the end effector joint rates orthogonal to the obstacle gradient becomes the commanded joint rates. A notional eleven DOF model is used to numerically demonstrate the efficacy of the control law.

TABLE OF CONTENTS

I. INTRODUCTION

II. MANIPULATOR KINEMATICS

III. INVERSE KINEMATICS

IV. LYAPUNOV STABILITY FOR END EFFECTOR TRAJECTORY

V. OBSTACLE AVOIDANCE

VI. LIMITATIONS ON JOINT RATES AND DEFLECTIONS

VII. CONCLUSIONS AND RECOMMENDATIONS

VIII. APPENDIX: *MATHEMATICA* PROGRAM FOR ROBOT SIMULATION

XI. REFERENCES

I. INTRODUCTION

The range of motion achievable by a robot manipulator's end effector is a function of the number and type of joints or degrees of freedom it possesses. Any degrees of freedom in excess of the minimum number required to reach an arbitrary end effector position and orientation within the workspace are considered "redundant". Commercial manipulators typically possess six or fewer DOF for primarily "anthropomorphic" tasks such as industrial assembly and are therefore not redundant.

There are some tasks for which such standard manipulators are not well suited, such as those requiring an extended reach in a confined workspace. For that reason, so-called "serpentine" manipulators have attracted interest. Their designation and appearance (Fig. 1) suggest the long reach and dexterity associated with snakes or tentacles. They achieve this snake-like ability by possessing a high degree of redundancy. This redundancy allows them, theoretically, to "wriggle" an end effector into a confined or difficult to reach point while allowing the robot arm to be configured in such a way as to not contact the surrounding environment.

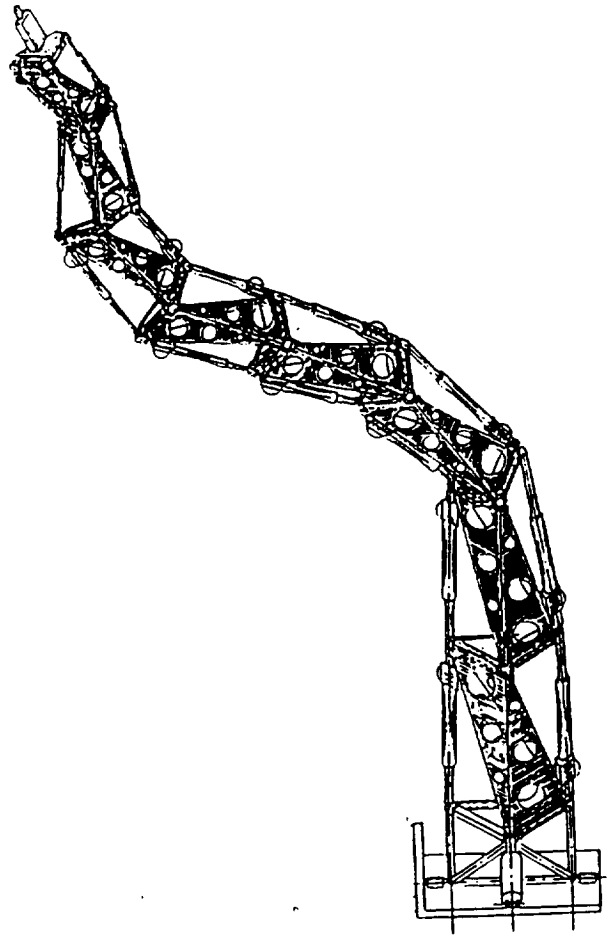

Figure 1
Serpentine Manipulator

One possible application is the inspection and processing of shuttle orbiter payloads in the Payload Changeout Room. During final launch preparations, tasks such as connecting/disconnecting umbilicals, removal of lens covers, or visual inspections must be carried out on Shuttle payloads. It is difficult and sometimes treacherous for technicians to see or reach many of the points at which these processes must take place and the payload itself may be put at risk. A serpentine robot is currently under development at KSC to study the feasibility for its use for such tasks [1].

There are two traditional approaches to controlling robot motion: to determine the dynamical equations of motion for each of the joints and generating the required torque for desired end-effector motion, or to control the joint velocities in response to the robot kinematics.

The complexity of serpentine motion, coupled with collision avoidance requirements, typically dictate relatively slow motion. This usually renders the dynamics of the robot arm negligible. Therefore, only the kinematics of the serpentine motion need be addressed.

Several approaches to controlling redundant manipulators for collision avoidance have been suggested. Maciejewski and Klein [2], Nakamura [3] and Wegerif, et al [4] make use of the pseudo-inverse and some variations of null-motion. Sciavicco, and Siciliano [5] make use of Lyapunov stability and an augmented configuration space to track a prescribed trajectory and incorporate obstacle avoidance. Alternatively, Pasch [1], and Asano [6] prescribe an end effector path and cause each joint to follow it in a "follow-the-leader' mode. All of these methods require that at least the end effector's trajectory and velocity be prescribed. This presumes that a clear path for the end effector is easily determined. Only [4] allows for the end effector to deviate from the prescribed path but only as an emergency measure.

In this paper, the author presents an alternative method for determining an acceptable robot trajectory which allows the end effector's path, as well as the entrained link's to be free to move around obstacles.

II. MANIPULATOR KINEMATICS

The position and orientation of the end effector $\underline{r} \in R^m$ is described, using the standard Denavitt-Hartenberg convention as a function of the generalized vector of joint displacements $\underline{q} \in R^n$ for a manipulator with n joints.

$$\underline{r} = \underline{r}(\underline{q}(t), t) \tag{1}$$

At present, only the end effector's position is of interest, giving $m = 3$. The Denavitt-Hartenberg convention, (described by Craig [7]), permits a single degree of freedom for each joint. The rotation matrix which relates the i_{th} link frame to the $(i\text{-}1)_{th}$ frame is given by:

$$ {}^{i-1}_{i}R = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix} \tag{2}$$

The angle α_i reflects the rotation of the i_{th} joint frame about the local x axis with respect to the $(i\text{-}1)_{th}$ frame. The angle θ_i is the rotation angle of the i_{th} joint and corresponds to a component of the vector $\underline{q}$. The end of the i_{th} link is located by the vector

$$\underline{r}_i = \underline{r}_{i-1} + {}^{i-1}_{i}R \begin{bmatrix} l_i & 0 & 0 \end{bmatrix}^T \tag{3}$$

so that the end effector is located by:

$$\underline{r} = \sum_{i=0}^{n} \underline{r}_i \tag{4}$$

Serpentine manipulators are characterized by a high degree of redundancy and by a distinctive structure. Revolute joints are essentially universal joints possessing two degrees of freedom. This permits any link to have an arbitrary direction in space. Two degrees of freedom are modeled in the D-H convention by incorporating links of zero length. Following the D-H convention, the link frames are alternately rotated about the link $\pm x$ axes with every other link having zero length. Equation (2) becomes:

$$\alpha_i = \begin{cases} -\pi/2, & i \text{ even} \\ \pi/2, & i \text{ odd} \end{cases} \Rightarrow {}^{odd}_{even}R = \begin{bmatrix} \cos\theta_i & 0 & -\sin\theta_i \\ \sin\theta_i & 0 & \cos\theta_i \\ 0 & -1 & 0 \end{bmatrix}, \quad {}^{even}_{odd}R = = \begin{bmatrix} \cos\theta_i & 0 & \sin\theta_i \\ \sin\theta_i & 0 & -\cos\theta_i \\ 0 & 1 & 0 \end{bmatrix} \tag{5}$$

The location of the link ends and the end effector is then found by Eq. (3). In this paper, for simulation purposes, a notional 11 DOF manipulator is used. It consists of five links each of unit length, each with 2 DOF and a base that can translate in the base frame x direction. Numerical simulations and their graphical presentation were facilitated by *Mathematica* [8] , a symbolic mathematics software package.

III. INVERSE KINEMATICS

The velocity of the end effector is calculated by :

$$\frac{d\underline{r}}{dt} = \left(\frac{\partial \underline{r}}{\partial \underline{q}}\right)\frac{d\underline{q}}{dt} \tag{6}$$

where the coefficient matrix ,$(\partial \underline{r} / \partial \underline{q}) = J$, is the *Jacobian* matrix.
To achieve a desired trajectory for the end effector, when $n>3$, the differential equation for the generalized joint displacements may be solved:

$$\dot{\underline{q}} = J^{\#}\dot{\underline{r}} \tag{7}$$

where $J^{\#}$ is the Moore-Penrose pseudo inverse given by:

$$J^{\#} = J^{T}\left(JJ^{T}\right)^{-1} \tag{8}$$

Eq. (7) yields the minimum norm solution for $\dot{\underline{q}}$. This solution assumes an appropriate scaling metric as discussed by Doty, et al [9]. In fact, because in a redundant manipulator, $n>m$, there are infinitely many solutions to Eq. (6). The joint rates are a function of those rates which cause the end effector to move, $\dot{\underline{q}}_R$ and "null rates", $\dot{\underline{q}}_N$ which do not. That is, $\dot{\underline{q}} = \dot{\underline{q}}_R + \dot{\underline{q}}_N$ where $\dot{\underline{q}}_R$ is the minimum norm motion given by Eq. (7) and $\dot{\underline{q}}_N$ is given by:

$$\dot{\underline{q}}_N = (E_n - J^{\#}J)\underline{\mu} \tag{9}$$

where E_n is the $n\times n$ identity matrix and $\underline{\mu} \in R^n$ is an arbitrary vector. It is a simple matter to confirm that $\dot{\underline{q}}_R$ and $\dot{\underline{q}}_N$ are orthogonal vectors by taking their inner product. Alternatively, multiplying Eq. (8) by J yields the null vector.

The selection of $\underline{\mu}$ generates one of an infinite number of joint rate combinations which move all of the links but do not cause motion in the end effector. There have been several control laws suggested which make use of the null motion to avoid obstacles. Several of these are recapped by Nakamura [3] in some detail. A major shortcoming of these methods is the requirement to prescribe the end effector path and velocity. Not only can this be a difficult task in itself for a complex workspace, but, in some situations, it proscribes joint motion which could avoid collisions.

IV. LYAPUNOV STABILITY APPROACH TO END EFFECTOR TRAJECTORY

In contrast to the pseudo-inverse approach, the author has adopted a Lyapunov stability measure similar to that in [5]. However, whereas the end effector in [5] tracks a prescribed end effector path, here only the final end effector position is required. The error vector $\underline{e}$ is defined:

$$\underline{e} = \underline{r}_T - \underline{r} \tag{10}$$

where $\underline{r}_T \in R^3$ is the vector locating the target with respect to the robot base frame. For a fixed target

$$\dot{\underline{e}} = -\dot{\underline{r}} \tag{11}$$

The Lyapunov scalar function v is defined

$$v = \tfrac{1}{2}\underline{e}^T\underline{e} \tag{12}$$

Because v is a positive scalar (related to the error's magnitude squared), then, if the time derivative of v is negative, $\underline{e}$ will go to zero as time approaches infinity. Taking the time derivative gives:

$$\begin{aligned} \dot{v} &= \underline{e}^T\dot{\underline{e}} \\ &= -\underline{e}^T\left(J\dot{\underline{q}}\right) \end{aligned} \tag{13}$$

An obvious selection for $\dot{\underline{q}}_R$ is to make its elements proportional to the elements of $J^T\underline{e}$. One such solution is given by

$$\dot{\underline{q}}_R = M\left(\frac{J^T\underline{e}}{\left\|J^T\underline{e}\right\|}\right) \tag{14}$$

where M is a positive definite matrix of dimension $n \times n$. The computational simplicity of Eq. (14) contrasts starkly with the complexity of computing the motion using the pseudo-inverse approach in Eq. (7). In addition, since $\dot{\underline{q}}_R$ is a unit vector scaled by M and no matrices must be inverted, the control law works well even in the vicinity of joint singularities.

The importance of an appropriate metric must be emphasized. Doty, et al [9] show that results may be obtained which are non-invariant with respect to choice of reference frame or dimensional units used to describe the problem. In Eq. (14) $\dot{\underline{q}}_R$ has the dimensions of *radians/time* for revolute joints and *length/time* for prismatic joints. On the right hand side of Eq. (14), J has dimensions of *length/radians* and *length/length* for revolute and prismatic joints, respectively, while the error vector has the units of *length.*

Clearly the units of $J^T \underline{e}$ are incompatible with $\dot{\underline{q}}_R$. Normalizing $J^T \underline{e}$ as a unit vector renders it dimensionless and the matrix M serves to give the appropriate unit metric. In addition, the selection of the magnitude of the elements of M may be used to emphasize the motion of some joints over that of others. This aspect will be discussed later as a means of avoiding excessive joint rates or deflections.

Figure 2 shows three simulations of robot motion for different values of a diagonal matrix M. For all the maneuvers, the manipulator has an initial end effector position $\underline{r}(0)=[3, 2, 0]$, shown in Fig. 2(a), and moves to a final end effector position of $\underline{r}(t_f) = [-3, 2, 0]$, indicated by the dot in the upper left portion of the workspace. In Fig. 2(b) the final configuration and the end effector path are shown for M equal to the identity matrix. That is, all of the joint rates are equally weighted. The end effector trajectory resembles a damped sinusoid.

In Fig. 2(c) the motion of the last two revolute joints is given a weight of ten times greater than the other nine joints. This causes the manipulator to attempt to reach the target primarily by moving these two joints, which at one point causes a near singularity. This is evidenced by the abrupt direction change of the end effector. Because of a fairly large step size in the *Mathematica* program, the final conditions are not satisfied exactly.

In Fig. 2(d) the motion of the translating base and the first revolute joint are emphasized by a factor of ten. The motion to the target is accomplished almost exclusively by the motion of these two joints.

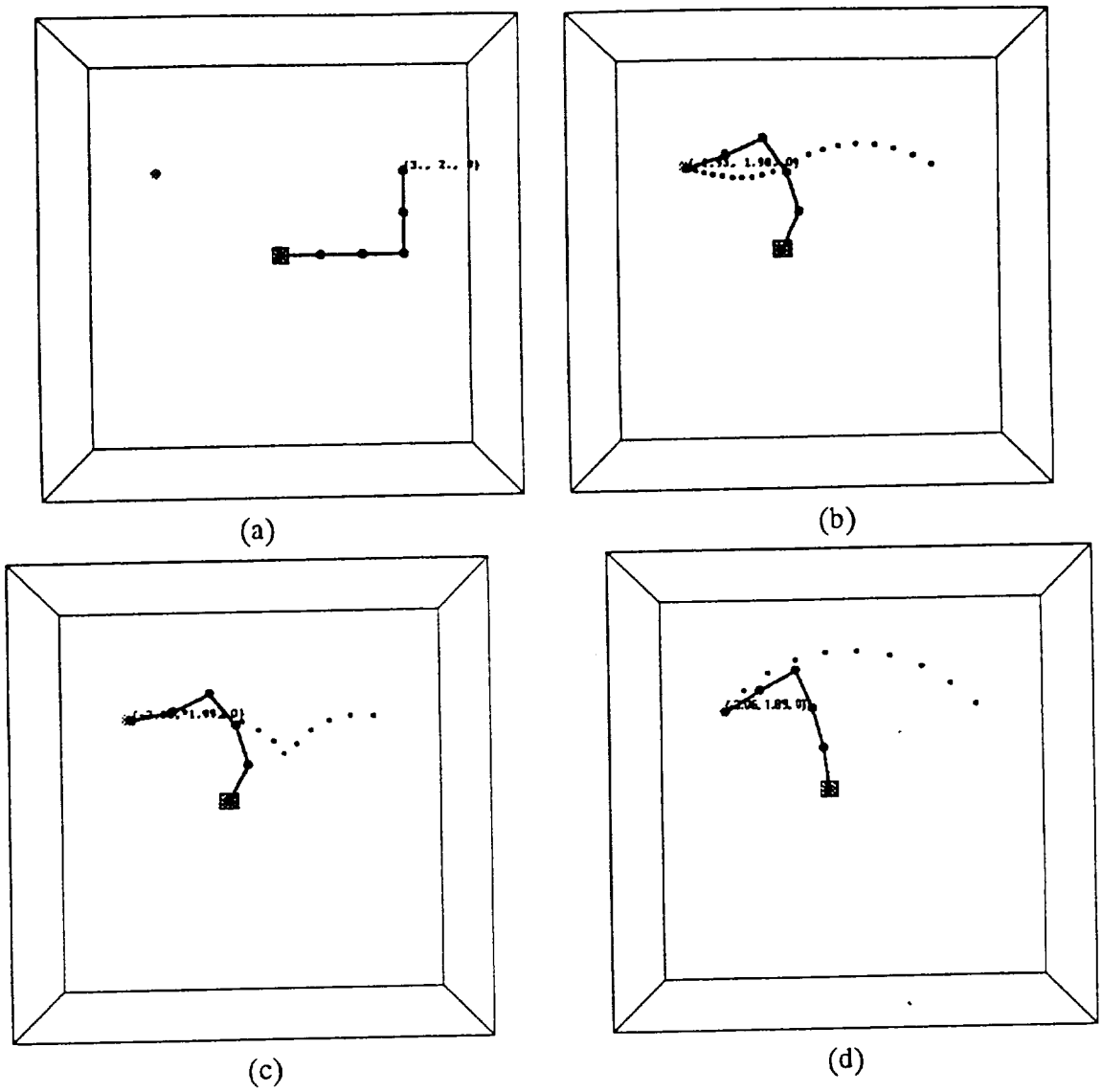

Figure 2.
End Effector Trajectory Determined By Lyapunov Function
(a) Initial Configuration
(b) Final Configuration with Equal Weights on Joint Rates
(c) Final Configuration with Last Two Joint Rates Emphasized
(d) Final Configuration with First Two Joint Rates Emphasized

V. OBSTACLE AVOIDANCE

As noted before, most other proposed methods of obstacle avoidance presume a prescribed end effector path that is obstacle free. This can be an important constraint because it may require detailed knowledge of the work space or excessively complicated path planing. By allowing the effector to seek its own path, the overall manipulator configuration becomes much more robust in its ability to avoid obstacles.

As suggested by Khatib[10], each obstacle is assigned a cost function. Figure 3 shows a representative manipulator arm with obstacle avoidance points $\underline{p}_i$, i = 1,... n_p, where $\underline{p}_i = \lfloor x_i \quad y_i \quad z_i \rfloor$, identified along it. A likely location for such points would be the manipulator joints and the link mid points but they may be dictated by sensor location or other criteria. In this paper, obstacles are assumed to be rectangular parallelepipeds with their center points o_j, and with dimensions $2a_j$, $2b_j$, $2c_j$, j= 1,...n_O.

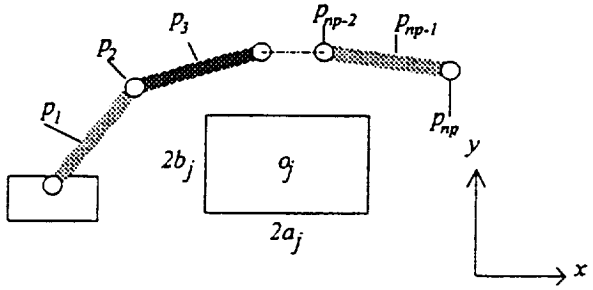

Figure 3.
Typical Designated Obstacle Avoidance Points

The cost function for the j_{th} obstacle with respect to the i_{th} obstacle avoidance point is given by the super-ellipsoid

$$C_j(p_i) = \left(\frac{x_i - x_j}{a_j}\right)^8 + \left(\frac{y_i - y_j}{b_j}\right)^8 + \left(\frac{z_i - z_j}{c_j}\right)^8 \tag{15}$$

Contact with the surface of the obstacle by the i_{th} point is approximated by $C_j(p_i) = 1$. A potential function is defined by:

$$P = \sum_j^{no} \sum_i^{np} \left[C_j(p_i) - 1\right]^{-1} \tag{16}$$

which guarantees that the cost becomes infinite before actual contract is made with the obstacle. The gradient of the potential function is

$$\underline{\mu} = \frac{\partial P}{\partial \underline{q}} \tag{17}$$

The time rate of change of P can thus be expressed

$$\frac{dP}{dt} = \underline{\mu}^T \dot{\underline{q}} \tag{18}$$

As one might expect, joint rates generated by Eq. (14) may also adversely change the proximity to obstacles. However, if the only requirement on the joint rates is that the error vector be decreased over every sub-interval, then an infinite number of trajectories may be found which accomplish this. Assuming that at least one unobstructed trajectory exists, one possible solution is to find the component of $\dot{\underline{q}}_R$ from Eq. (14) which is orthogonal to $\underline{\mu}$. This results in $\dot{P} = 0$. The Gram-Schimdt procedure described by Luenberger [11], subtracts from $\dot{\underline{q}}_R$ its projection in the $\underline{\mu}$ direction. This results in the commanded joint rates:

$$\dot{\underline{q}} = \dot{\underline{q}}_R - \left[\left(\dot{\underline{q}}_R\right)^T \hat{\underline{\mu}}\right]\hat{\underline{\mu}} \tag{19}$$

where $\hat{\underline{\mu}}$ is a unit vector parallel to $\underline{\mu}$. A three dimensional analogy is shown in Figure 4. Moving toward a destination, a traveler's path intersects a portion of a hill. The traveler's location is analogous to the robot's current joint configuration; the hill is an obstacle. Going uphill in the direction of the gradient, increases the cost. By moving along a contour line of constant cost, orthogonal to the slope, the traveler may simultaneously move closer to his destination without going uphill. Eventually a point is reached where the target is downhill.

In the three dimensional analogous state, it is easy to see that if the destination lies exactly opposite of the summit from the present position, forward motion eventually becomes impossible (the cost becomes infinite). Motion stops for the robot in the event that no motion whatever will move the end effector closer to the target without colliding with an obstacle. In the n dimensional joint space however, this possibility recedes as n becomes large. That is, although this algorithm does not explicitly require redundancy, redundancy increases its robustness.

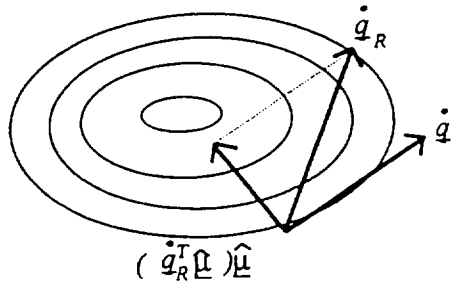

Figure 4.
Three Dimensional Analogy of Obstacle Avoidance

The control embodied in Eqs. (14) and (19) is demonstrated in the simulation shown in Figure 5. Because the interest in serpentine manipulators is largely due to the potential for reaching targets in constricted areas, in this example, the target is located "down the hall and around the corner" with the walls modeled by three obstacles. A collision free trajectory is generated by the method described above. Once again the end effector trajectory is non-intuitive.

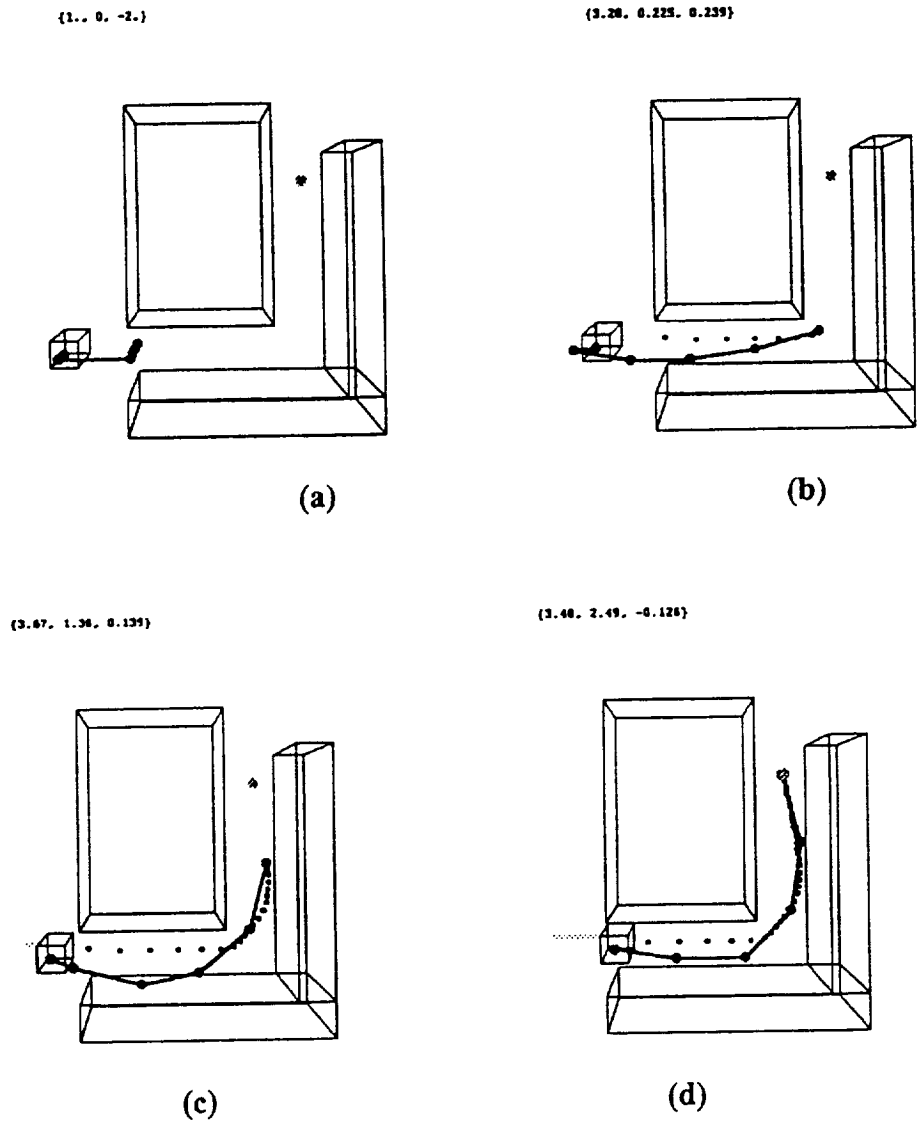

Figure 5
Obstacle Avoidance Trajectory
(a) Initial Configuration
(b),(c) Intermediate Configurations
(d) Final Configuration

VI. LIMITATIONS ON JOINT RATES AND DEFLECTIONS

In addition to avoiding obstacles, manipulator arms are frequently limited in the magnitude of the joint deflections which can be achieved. In addition, the joint rates are usually limited by the manipulator architecture. While not explicitly examined in the author's research, some possible solutions are suggested.

It has been demonstrated that the joint rates can be influenced by the weighting of the elements of the matrix M. The most straightforward approach is to weight each joint rate independently by making M diagonal. The weight on each joint rate may be made a function of its current deflection and commanded joint rate.

It is useful to think of the diagonal elements of M as the stiffness coefficients of n non-linear spring. The deflection of the i_{th} joint is bounded by $q_{i_{min}} \le q_i \le q_{i_{max}}$. Defining:

$$\begin{aligned} \Delta_i &= q_{i\max} - q_{i\min} \\ \Gamma_i &= q_{i\max} + q_{i\min} \\ f_i &= \frac{2q_i - \Gamma_i}{\Delta_i} \\ \eta_i &= \operatorname{sign}\left[\left(J^T \underline{e}\right)_i\right] \end{aligned} \tag{20}$$

The diagonal elements of M are defined:

$$m_{ii} = k_i(1 - \eta_i f_i), \quad i = 1,\ldots,n \tag{21}$$

where $k_i \le \dot{q}_{i_{max}} / 2$.

This function guarantees that the maximum allowable joint rate is never exceeded and motion away from the maximum deflections is encouraged while motion toward maximum deflection is discourage. This approach has not been implemented in any of the examples in this paper and requires further verification.

VII. CONCLUSIONS AND RECOMMENDATIONS

A method for moving a serpentine manipulator's end effector to a target in a constricted area while avoiding collision's of the manipulator's arm with the surrounding workspace has been demonstrated. This method has the virtues of being computationally straightforward. It is robust in the vicinity of singularities and multiple obstacles. While it must be emphasized that this paper discusses only very preliminary results, the algorithm appears to have great potential for successful implementation for achieving numerous robot tasks.

Although the algorithm appears to be fairly versatile, the ability of the manipulator to reach a target can be sensitive to its initial configuration relative to the target. While path planning is not explicitly required, it is necessary to orient the robot with respect to the target so that a likely path is unambiguous. In addition, it has been observed that there are a number of cases in which the robot will not be able to reach the target. It is possible for the end effector to arrive a point where no further forward motion is possible. This is the case where a wide, flat obstacle is approached and only moving away from the target will eventually result in a configuration in which forward motion. Additional heuristics need to be developed to address this possibility.

The scaling matrix requires further research. While the suggested configuration works adequately, there is room for improvement. Further research is especially necessary in properly scaling the control vector in order to avoid joint rate and deflection limitations. These limitations, although addressed in this paper, should be further investigated in the context of a realistic robot architecture.

Finally, although *Mathematica* is a versatile programming tool, it is far too slow for practical numerical integration. For implementation on an actual robot, programming in *C* language is recommended. *Mathematica* may be linked to a *C* code to evaluate symbolically complicated expressions such as the Jacobian matrix or the obstacle gradient vector.

VIII. APPENDIX: *MATHEMATICA* PROGRAM FOR ROBOT SIMULATION

Mathematica was used to produce the simulations in this paper. The parameters in the following program are those for the simulation depicted in Fig. 5. *Mathematica* commands are shown in Courier font. Explanatory comments have been added in Times Roman font.

```
ClearAll[x,theta,al,p,r0,J,T0];
Clear[n,d,obsnum];
JacobianMatrix[funs_List, vars_List]:=Outer[D,funs,vars];
Norm[vars_List]:=Sqrt[vars.vars];
UnitVector[vars_List]:=vars/Sqrt[vars.vars];
al:=Array[alpha,n];  (*Vector of frame rotation angles*)
x:=Append[Array[theta,n],d];  (*Generalized Vector of joint displacements.
The translational displacement of the base is given by d.*)
p:=Array[l,n];  (*Vector of link lengths*)
```

(*R[n] defines the rotation matrix relating the *nth* link frame to the (*n*-1)*th* frame using standard Denavitt-Hartenberg convention *)

```
R[n_]:={{Cos[x[[n]]],-Cos[al[[n]]]  Sin[x[[n]]],Sin[al[[n]]]
Sin[x[[n]]]},
     {Sin[x[[n]]],Cos[al[[n]]]     Cos[x[[n]]],-Sin[al[[n]]]
Cos[x[[n]]]},
     {0,Sin[al[[n]]],Cos[al[[n]]]}};
```

(*T0[n] gives the orientation of the *nth* link with respect to the base frame*)

```
T0[n_]:=T0[n]=T0[n-1].R[n];
T0[0]=IdentityMatrix[3];
R[0]=IdentityMatrix[3];
r0[n_]:=r0[n]=r0[n-1]+T0[n].{p[[n]],0,0};  (*Endpoint of nth link*)

mp[n_]:=mp[n]=r0[n-1]+T0[n].{p[[n]]/2,0,0};(*Midpoint of nth link*)
r0[0]={d,0,0};
mp[0]={d,0,0};
(*Manipulator architecture is defined by n, al, and p *)
n=10;
al={Pi/2,-Pi/2,Pi/2,-Pi/2,Pi/2,-Pi/2,Pi/2,-Pi/2,Pi/2,-Pi/2};
p={0,1,0,1,0,1,0,1,0,1};
joints=Table[Point[r0[i]],{i,0,n}];  (*Table of  joint coordinates; for
plotting purposes*)
arm=Line[Table[r0[i],{i,0,n}]];(Line from joint to joint; for plotting
purposes*)
J=JacobianMatrix[r0[n],x];
```

(*Obstacles are depicted as rectangular solids with six coordinates: first three values are coodinates of mass center. Second three values are *x,y,z* dimensions*)

```
obsnum=3;
ob[1]={2,2,-.5,2,3,5};
ob[2]={2.75,-.75,-.5,3.5,.5,5};
ob[3]={4.25,1.25,-.5,.5,3.5,5};
```

(*obstacleshape is a function which draws a rectangular solid in the plot to represent each obstacle*)

```
obstacleshape[k_]:=
Cuboid[{(ob[k][[1]]-ob[k][[4]]/2),
(ob[k][[2]]-ob[k][[5]]/2),(ob[k][[3]]-ob[k][[6]]/2)},
{(ob[k][[1]]+ob[k][[4]]/2),
(ob[k][[2]]+ob[k][[5]]/2),(ob[k][[3]]+ob[k][[6]]/2)}];
```

(*cost is the potential function*)

```
cost= (Sum[Sum[
1/(Sum[((mp[i][[j]]-ob[k][[j]])/(ob[k][[j+3]]/2))^8,
{j,1,3}]-1.1),{i,0,n}]+
1/(Sum[((r0[n][[j]]-
ob[k][[j]])/(ob[k][[j+3]]/2))^8,{j,1,3}]-1.1),
{k,1,obsnum}]);
```

(* mu is the obstacle gradient vector*)

```
mu=Table[D[cost,x[[i]]],{i,1,n+1}];

step=.05;(*step size*)
imax=100;(*maximum number of steps*)
target={3.5,2.5,0};
d=0;
theta[1]=0;
theta[2]=N[Pi/2];
theta[3]=0;
theta[4]=-N[Pi/2];
theta[5]=0;
theta[6]=-N[Pi/2];
theta[7]=0;
theta[8]=0;
theta[9]=0;
theta[10]=0;
pts={Point[r0[n]]};
base={Point[{d,0,-3}]};
Metric=DiagonalMatrix[1,1,1,1,1,1,1,1,1,1,1];
```

(* Robotsim is a function which performs the actual simulation. It is largely devoted to drawing the graphic images of the robot motion. It is started by compiling the program and then typing "Robotsim"*)

```
Robotsim:=
For[i=0,i<=imax,i++,
If[EvenQ[i], path=Append[pts,Point[r0[n]]];
track=Append[base,Point[{d,0,-3}]];
Show[
Graphics3D[{{AbsoluteThickness[2],arm,
          Cuboid[{d-.2,-.2,-3},{d+.2,.2,0}]},
     {RGBColor[1,0,0],PointSize[.02],joints},
{RGBColor[0,0,1],AbsoluteThickness[1],Table[path]},
{RGBColor[1,1,0],AbsoluteThickness[1],Table[track]},
{RGBColor[0,1,0],PointSize[.02],Point[target]},
{Table[obstacleshape[i],{i,1,obsnum}]},
{Text[NumberForm[r0[n],3],{0,5,0},{-1, 1}]}}],
Boxed->False,  ViewPoint->{0.000,0.000,3.384},PlotRange->{{-
1,5},{-1,5},{-3,5}}];
pts=path;
base=track;
Print[i]];
If[Sqrt[(target-r0[n]).(target-r0[n])]<.001,i=imax];
```

(* The last six lines of code perform a fairly crude numerical integration with a first order Euler's method*)

```
vr=UnitVector[(target-r0[n]).J.Metric];
nu=UnitVector[mu];
v=UnitVector[vr- vr.nu nu];
For[j=1,j<=n,j++,
new[j]=theta[j]+step v[[j]];theta[j]=new[j]];
new[n+1]=d+step v[[n+1]];d=new[n+1]];
```

IX. REFERENCES

[1] Pasch, K., "Self-Contained Deployable Serpentine Truss for Pre launch Access of Space Shuttle Orbiter Payloads", NAS -2659-FM-9106-387, Final Report, Contract No. NAS10-11659. NASA, Kennedy Space Center, FL.

[2] Maciejewski, A., and Klein, C., "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments", *The International Journal for Robotics Research,* Vol. 4, No. 3, Fall 1985, pp. 109-117.

[3] Nakamura, Y., *Advanced Robotics, Redundancy and Optimization,* Addison-Wesley, Publishing Co., Inc., Redwood City, CA, 1991.

[4] Wegerif, D, Rosinski, D., and Parton, W., "Results of Proximity Sensing Research for Real-Time Collision Avoidance of Articulated Robots Working Near the Space Shuttle", *Proceedings of the 6th Annual Conference on Recent Advances in Robotics,* University of Florida, Gainsville, FL, 19-20 April 1993.

[5] Sciavicco, L., and Siciliano, B., "A Solution Algorithm to the Inverse Kinematic Problem for Redundant Manipulator", *IEEE Journal of Robotics and Automation,* Vol. 4, No. 4, Aug. 1988, pp. 403-410.

[6] Asano, K, et al, "Multijoint Inspection Robot", *IEEE Transactions on Industrial Electronics,* Vol. IE-30, No. 3, August 1983, pp. 277-281.

[7] Craig, J., *Introduction to Robotics*, 2nd Ed., Addison-Wesley, Publishing Co., Inc., Redwood City, CA, 1989.

[8] Wolfram, S., *Mathematica,* 2nd Ed., Addison-Wesley Publishing Co., Inc.,Redwood City, CA, 1991.

[9] Doty, K., Melchiorri, C., & Bonivento, C., " A Theory of Generalized Inverses Applied to Robotics", *The International Journal of Robotics Research,* Vol. 12., No. 1, Feb. 1993, pp. 1-19.

[10] Khatib, O. and Le Maitre, J.-F., "Dynamic Control of Manipulators Operating in a Complex Environment", *Proc. 3rd Int. CISM-IFToMM Symp.*, pp. 267-282.

[11] Luenberger, D., *Optimization by Vector Space Methods,* John Wiley & Sons, Inc., New York, 1969.

[12] Nakamura, Y., and Hanafusa, H., "Optimal Redundancy Control of Robot Manipulators", *International Journal of Robotics Research,* Vol. 6, No. 1., Spring 1987, pp. 32-42.