EXTENSIBILITY IN LOCAL SENSOR BASED PLANNING FOR HYPER-REDUNDANT MANIPULATORS (ROBOT SNAKES)

Howie Choset Joel Burdick
Dept. of Mechanical Engineering
Mail Code 104-44, CALTECH, Pasadena, CA 91125

ABSTRACT: This paper extends a *local* sensor based planning method for hyper-redundant robot mechanisms. In a previous paper, sensor feedback control methods are considered. A highly localized sensor feedback method for hyper-redundant manipulators is termed, *partial shape modification* (PSM). A PSM utilizes a mechanism's hyper-redundancy to enable both local obstacle avoidance and end-effector placement in real time. This paper considers the situation in which the limits of a PSM is violated. In other words, what does the robot do when it can not only locally adapt to the environment. Local sensor based planning has been implemented on a thirty degree of freedom hyper-redundant manipulator which has eleven ultrasonic distance measurement sensors and twenty infrared proximity sensors. The robot is controlled by a real time control computer which communicates with sensors through an innovative sensor bus architecture. Experimental results obtained using this test bed show the efficacy of the proposed method.

1. Introduction

This paper extends experimental results from [TCB] in the area of *local sensor based planning* for hyper-redundant robot manipulators. Recall from [ChB90b] that a "hyper-redundant" manipulator is a kinematically redundant manipulator in which the degree of redundancy is very large or infinite. Such robots are analogous in morphology to tentacles, an elephant's trunk, a monkey's tail or a snake. "Sensor based planning" incorporates sensory information into some stage of a robotic motion planning, whether it be navigation, locomotion, grasping, etc. "Local Sensor Based Planning" fine tunes a robot's plan, based on sensor information. Local sensor based planning is useful when: (1) the robot only has a coarse knowledge of the world because of limited memory; (2) the robot's world model contains inaccuracies; and (3) the world is subject to unexpected occurrences or rapidly changing situations. These situations can be overcome with local sensor based planning strategies.

Due to their many degrees of freedom, hyper-redundant robots are potentially superior for operations in highly constrained and unusual environments encountered in applications such as inspection of nuclear reactor cores, chemical sampling of buried toxic waste, and medical endoscopy. Hyper-redundant robots can also be used as tentacle-like grasping devices for capturing and manipulating floating satellites [ChB90c] or to enable complex "whole arm manipulation." Mobile hyper-redundant robots also offer novel means for locomotion [ChB91a, ChB93a, ChB93b, ChB93c] in complex environments.

The above mentioned applications are characterized by environments which are difficult to precisely model and which are time varying. Thus, local sensor-based motion planning schemes are vital to the realistic deployment of hyper-redundant robots in these applications. While hyper-redundant robots have many advantages for the above described applications, they have one disadvantage. Since hyper-redundant manipulators have a large number of joints or actuators, small joint displacement errors can accumulate to reasonably large errors in the position of the tip relative to the base. Thus, the effective accuracy of hyper-redundant robots could be improved by distributing sensors along their length and employing sensor based planning schemes.

Thus, local sensor based planning can be used to: (1) account for spatial uncertainty or inaccuracies in the world model used by a "global" planner to construct a robot plan; (2) increase the effective accuracy of a hyper-redundant robot mechanism; and (3) locally adapt to rapid environmental variations, such as moving obstacles, that can not be easily or rapidly handled by a global planner.

The local sensor based planning algorithm of hyper-redundant manipulators is based on the analysis found in [ChB90b, ChB91a, ChB92a, ChB92c, Ch]. This work has been demonstrated on an actual extensible 30 degree-of-freedom hyper-redundant robot system. A hyper-redundant manipulator which can vary its length, within the limitations of its actuators, is termed "extensible." A more detailed account of this mechanism and its capabilities can be found in [ChB92b].

Robotic motion planning has been an important area of research. Since the introduction of configuration space methods [LoWes], several other theories have been published, some of which are summarized in [Sh,Lat]. However, these methods plan from a perfect model of the world, which is normally unavailable to a real robot. More recently, methods have been developed in which the robot explores the environment to gather information for the planning process [CaLi]. These approaches assume that the sensors provide perfect information about the environment. There has been little work devoted explicitly to motion planning for robot snakes. One approach is based on the construction of *tunnels* through the obstacle field, through which the manipulator "slithers" [ChB90a, Ch]. In another work, sensor based planning for highly redundant robots is based on a *tactrix* [ReLu]. However, this work assumes that there are perfect sensors on the robot; nor has it been implemented on a real robot. Hirose [HiU] implemented an "active cord" mechanism, which used tactile sensors to guide its motion. A previous paper [TCB] presents preliminary strategies for local sensor based planning, which are implementable in real time, can employ a variety of sensors, and exploit the benefits of hyper-redundancy.

In this paper, a local sensor based planning strategy for hyper-redundant manipulators is extended so it can be more accommodating. The local sensor based planner in [TCB] did not consider its own limitations. There, the hyper-redundant manipulator can only locally adapt to a changing environment over a fixed length of the robot. For a fixed length of robot, a hyper-redundant manipulator uses its extensibility so it can locally avoid obstacles. However, the robot can only deform until its joint limits are met, in which case the hyper-redundant manipulator can no longer adapt. In other words, the robot used up all of its extensibility over the fixed length. The longer this fixed length, the more the robot can deform. However, the longer the length, the less local the response, which is undesireable. In the new algorithm, the length of the deforming part of the robot is variable, therefore enhancing its ability to locally adapt. In effect, the manipulator uses more of its extensibility from other parts of the robot to locally avoid objects. Experiments demonstrate that local sensor based planning is not only useful, but also implementable in real time with very reasonable computing power and simple sensors.

The structure of this paper is as follows. Section 2 reviews the basic framework of hyper-redundant manipulator kinematics which is based on "backbone curves." The backbone curve and its deformation is the basis for the algorithms of Section 3. We primarily consider algorithms for planar mechanisms, as the experimental verification of these ideas was performed on a planar robot. Many of these algorithms can be extended to the spatial case. Section 4 describes the experimental setup, while Section 5 described the actual results of these experiments.

2. Background

This section reviews a hyper-redundant robot kinematic analysis framework that forms the basis of this work. Recall from [ChB90b] that we assume that (regardless of mechanical implementation) the important macroscopic features of a hyper-redundant robot can be captured by a *backbone curve.* A backbone curve parametrization and an associated set of reference frames which evolve along the curve are collectively called the *backbone reference set.* In this paradigm, inverse kinematics and task planning reduces to the determination of the proper time varying behavior of the backbone reference set [ChB90b]. Similarly, local sensor based planning is equivalent in this approach to modification of the backbone curve shape in order to accommodate impinging obstacles.

In [ChB92c], many techniques are introduced for parametrizing the backbone curve. In this paper, we will assume that the Cartesian position of points on a backbone curve can be parametrized in the form:

$$\vec{x}(s,t) = \int_0^s l(\sigma,t)\ \vec{u}(\sigma,t)d\sigma \qquad (2.1)$$

where $s \in [0,1]$ is a parameter measuring distance along the backbone curve at time t. The *backbone curve base* is the point $s = 0$. $\vec{x}(s,t)$ is a vector from the backbone curve base to point s. By convention, $\vec{x}(0,t) = 0$. $\vec{u}(s,t)$ is the unit tangent vector to the curve at s. $l(s,t)$ is the length of the curve tangent and assumes the general form:

$$l(s,t) = 1 + \epsilon(s,t) > 0. \qquad (2.2)$$

$\epsilon(s,t)$ is the *local extensibility* of the manipulator, which expresses how the backbone curve locally expands or contracts relative to a fixed reference state. We show later on that the robot needs this extensibility in order to locally avoid obstacles.

The parametrization of Eq. (2.1) has the following interpretation. The backbone curve is "grown" from the base by propagating the curve forward along the tangent vector, which is varying its direction according to $\vec{u}(s,t)$ and varying its magnitude (or 'growth-rate') according to $l(s,t)$.

Our experiments have been performed on a device with planar geometry. In the planar case, the backbone curve is the locus of points:

$$\vec{x}(s,t) = [x_1(s,t),\ x_2(s,t)]^T$$

where

$$x_1(s,t) = \int_0^s l(\sigma,t)\sin\theta(\sigma,t)d\sigma \qquad (2.3)$$

$$x_2(s,t) = \int_0^s l(\sigma,t)\cos\theta(\sigma,t)d\sigma. \qquad (2.4)$$

$\theta(s,t)$ is the angle, measured clockwise, which the tangent to the curve at s makes with the x_2-axis at time t. By convention (2.4), $\theta(0) = 0$, and $x_1(0) = x_2(0) = 0$. By comparing equations (2.1) with equations (2.3) and (2.4), it easy to see that $\vec{u}(s,t) = [\sin\theta(s,t),\ \cos\theta(s,t)]^T$ in the planar case. $l(s)$ and $\theta(s)$ are termed "shape functions," as they control they shape of the backbone curve through the forward kinematic relations (2.3) and (2.4).

Within the context of this modeling technique, the inverse kinematic problem, or "hyper-redundancy resolution" problem, reduces to the determination of the time varying behavior of backbone curve shape functions that satisfies task requirements. Different hyper-redundancy resolution techniques can be found in [ChB90a, ChB91a, ChB92a, ChB92c, Ch]. In one approach, which is relevant to the algorithm of Section 3, the backbone curve shape functions are restricted to a "modal form"

$$\theta(s,t) = \sum_{i=1}^{N_\theta} a_i(t)\ \Phi_i(s) \qquad l(s,t) = \sum_{i=N_\theta+1}^{N_l} a_i(t)\ \Phi_i(s) \tag{2.5}$$

where $\Phi_i(s)$ is a "mode function," and $a_i(t)$ is the associated "modal participation factor." $N = N_\theta + N_l$ is the total number of modes, which must equal or exceeds the number of task constraints. Hyper-redundancy is resolved in the modal approach by constraining the backbone curve to N effective DOF. The $\{\Phi_i\}$ are predetermined functions chosen by the programmer, and can often be selected to incorporate physical characteristics of the task [Ch]. Thus, the backbone curve geometry becomes solely a function of the $\{a_i\}$. The inverse kinematics problem reduces to finding the $\{a_i\}$ which satisfy task constraints. In [ChB91a, ChB92c], closed form solutions are given for several choices of mode functions.

A continuous backbone curve inverse kinematic solution is used to determine the actuator displacements of a continuous morphology robot such as one constructed from pneumatic actuator bundles. For discretely segmented morphologies, such as the prototype described in this paper in Section 4, the continuous curve solution can be used, via a "fitting" process, to compute the actuator displacements which cause the manipulator to exactly assume or closely approximate the continuous backbone curve model. The fitting techniques which are used in subsequent examples are reviewed in [ChB91a, ChB92c].

3. Local Sensor Based Planning Algorithm

Local Sensor-Based Planning (LSBP) assumes that a backbone curve is somehow determined by a high level global planning process. The backbone curve shape is then modified in response to sensory information. LSBP does not use a model of the environment, and is intended for rapid response to environment changes. In order to describe the local sensor based planning strategy, a sensor model must be described.

3.1. Sensor Models

The algorithm described below uses a very simple sensor model. We assume that the sensors are rigidly attached to the backbone curve at a fixed point. That is, they move with the backbone curve, and their orientation is a function of the backbone curve tangent at the point of attachment. The sensors are assumed to measure, along a fixed direction termed the *sensor measurement axis*, the distance to a nearby obstacle. The sensor measurement axis is a function of the sensor and the backbone curve geometry (See Fig. 1). Our sensors *do not* measure the distance to the point on the obstacle which is nearest to the backbone curve. Rather, they measure the distance which would actually be computed by realistic sensors. This simple model is representative of the infrared and ultrasonic sensors discussed in Section 4. In addition, there is often some directional ambiguity due to the finite width of a typical sensor's beam pattern. We assume that the sensor measurement axis is the centerline of the beam pattern. The distance measurement returned by the sensor is the nearest point of the obstacle lying within beam pattern cone. Since it is impossible to resolve the angular ambiguity, we assume that nearest point of the obstacle lies along the beam pattern centerline.

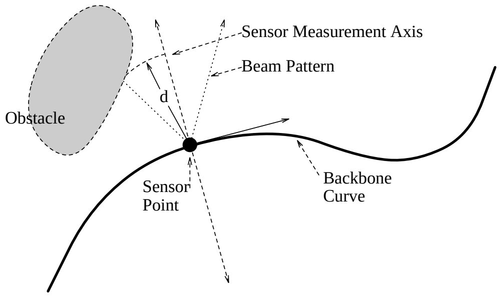

Figure 1: Simplified Distance Measurement Sensor Model

3.2. Partial Shape Modification Control

This section describes a PSM planning strategy in which the backbone curve is approximated by a large number, n_d, of discrete endpoints. The sensors are assumed to be rigidly attached at points along the discretized backbone curve. There are typically many approximating segments between adjacent sensor attachments. When a sensor detects the presence of an obstacle, the backbone curve shape locally deforms in a region around the sensor. In our simulations and experiments, $n_d \sim 100$, and there were about 10 discrete points between sensor points.

The actual response, a displacement of the approximating points, is determined by a *local sensor response function* (LSRF), which is assumed to be a discrete unimodal function. A unimodal function is one which has one local maximum, the global maximum, over its domain. The response function is "added" to the current backbone curve, locally drawing it away from an obstacle. In Figure 2, a triangle LSRF is added to a straight backbone curve, deforming the backbone curve away from an sufficiently close obstacle.

Since the robot only detects the obstacle at a sensor point, the reaction to the obstacle should be greatest at the sensor point, and should monotomically decrease at points away from the sensor point. Therefore, the sensor point is the center of this unimodal function, and is assumed to be farthest away from the obstacle.

The LSRF should also look like the beam pattern of the sensor associated with the sensor point. Typically, beam patterns have a central lobe along the sensor axis, in which the obstacle likely lies. In the experimental setup, the spatial resolution of the robot's actuators is much lower than the azimuth resolution of the sensors on the robot. Therefore, a simple triangle (or cone) is a sufficient approximation to the main lobe of the beam pattern, and thus, a reasonable choice for a LRSF. Later on, it is shown that a triangle response function leads to a trivial and efficient solution to LSBP for planar hyper-redundant manipulators. So, the example displayed in Figure 2 is a good example of a LSRF.

The half width of the LSRF is slightly larger than the distance between two adjacent sensors on the backbone curve. This way, if two adjacent sensors detect the same obstacle, their cumulative response function is still unimodal.

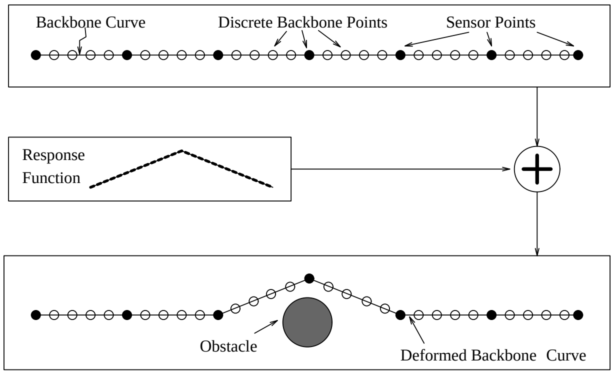

Figure 2: Backbone, Response Function, and Deformed Backbone

In this approximation method, the position of the discrete segment endpoints can be approximated by the discretization of the continuous forward kinematics integral (Eq. (2.1)):

$$\vec{p}(s_i, t) = \sum_{k=0}^{k=i} l(s_k, t)\vec{u}(s_k, t) \tag{3.2.1}$$

$l(s)$ and $\vec{u}(s)$ are continuous shape functions which are specified by a global planner. They need not assume a modal form. Also, an endpoint may or may not coincide with a sensor point.

A small differential change in $\vec{p}(s_i)$ is:

$$\delta\vec{p}(s_i, t) = \sum_{k=0}^{k=i} \delta l(s_k, t)\vec{u}(s_k, t) + l(s_k, t)\delta\vec{u}(s_k, t) \tag{3.2.2}$$

where $s_k = \frac{k}{n_d}$, where n_d is the number of discrete points along the back bone curve. $\delta\vec{u}$ is a local change in the backbone curve tangent direction, while $\delta\vec{l}$ represents a local stretch.

The goal of this PSM method is to compute the local perturbations, $\delta\vec{u}$ and $\delta\vec{l}$, which deform the backbone curve away from obstacles. The changes in backbone curve tangent and stretch are determined from $\delta\vec{p}$, which in turn is determined by the LSRFs. The modified backbone curve shape is then used by the fitting algorithms to determine the appropriate actuator displacements.

Assume that at some initial time, a global planner specifies a backbone curve shape. Thus, $\delta\vec{p} = 0$ initially. For each sensor point along the backbone curve that detects an obstacle within its response envelope, a discrete unimodal LSRF is added to (or subtracted from, depending upon from which direction an obstacle appears to be) δp, the vector which contains the prescribed changes to the backbone curve. Setting $\delta p(s_n, t) = 0$, guarantees that, within the limits of the discretization approximation, the end effector position will not change. Setting $\delta p(s_n, t) = 0$, $\delta p(s_{n-1}, t) = 0$, and $\delta\vec{u}(s_n, t) = \vec{0}$ guarantees that the end effector position and orientation will not change. The new backbone curve can be computed after $\delta\vec{u}$ and $\delta\vec{l}$ are determined from (3.2.1).

In the case of a planar backbone curve, (3.2.2) can be written in matrix form:

$$\begin{pmatrix} \delta p_x^1 \\ \delta p_y^1 \\ \delta p_x^2 \\ \delta p_y^2 \\ \vdots \\ \delta p_x^{n-1} \\ \delta p_y^{n-1} \\ \delta p_x^n \\ \delta p_y^n \end{pmatrix} =$$

$$\begin{pmatrix} l_1 & 0 & 0 & 0 & \dots & 0 & 0 & u_x^1 & 0 & 0 & \dots & 0 \\ 0 & l_1 & 0 & 0 & \dots & 0 & 0 & u_y^1 & 0 & 0 & \dots & 0 \\ l_1 & 0 & l_2 & 0 & \dots & 0 & 0 & u_x^1 & u_x^2 & 0 & \dots & 0 \\ 0 & l_1 & 0 & l_2 & \dots & 0 & 0 & u_x^1 & u_y^2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ l_1 & 0 & l_2 & 0 & \dots & 0 & 0 & u_x^1 & u_x^2 & u_x^3 & \dots & 0 \\ 0 & l_1 & 0 & l_2 & \dots & 0 & 0 & u_y^1 & u_y^2 & u_y^3 & \dots & 0 \\ l_1 & 0 & l_2 & 0 & \dots & l_n & 0 & u_x^1 & u_x^2 & u_x^3 & \dots & u_x^n \\ 0 & l_1 & 0 & l_2 & \dots & 0 & l_n & u_y^1 & u_y^2 & u_y^3 & \dots & u_y^n \end{pmatrix} \begin{pmatrix} \delta u_x^1 \\ \delta u_y^1 \\ \delta u_x^2 \\ \delta u_y^2 \\ \vdots \\ \delta u_x^n \\ \delta u_y^n \\ \delta l_1 \\ \delta l_2 \\ \delta l_3 \\ \vdots \\ \delta l_n \end{pmatrix} \tag{3.2.3}$$

where $\delta\vec{p}^i = \delta\vec{p}(s_i, t)$, $\delta\vec{u}^i = \delta\vec{u}(s_i, t)$, $\delta\vec{l}_i = \delta\vec{l}(s_i, t)$, $\vec{l}_i = \vec{l}(s_i)$, and $\vec{u}^i = \vec{u}(s_i, t)$. The x and y components of $\delta\vec{p}(s_i, t)$ are respectively denoted δp_x^i and δp_y^i. Similary, the x,y components of $\delta\vec{u}(s_i, t)$ are δu_x^i and δu_y^i. $\delta\vec{p}$ and $\delta\vec{u}$ each have $2n$ elements, and $\delta\vec{l}$ has n elements.

For given $\delta\vec{p}$, there is not a unique solution to Eq. (3.2.3). A simplified (and unique) solution for

(3.2.3) is obtained by setting $l(s_i,t) = 1\ \forall_{1\le i\le n}$ (i.e $\delta l(s_i,t) = 0$). Although all of the $l(s_i,t) = 1$, the snake can still use its extensibility to avoid obstacles because $\|\vec{u}^i + \delta\vec{u}^i\| \neq 1$. In other words, the length of the tangent vectors are no longer constrained to have unit length in this approximation unless additional restrictions are employed. After setting $\delta l(s_i,t) = 0$ and enforcing the end effector constraints,(3.2.3) becomes:

$$\begin{pmatrix} \delta p_x^1 \\ \delta p_y^1 \\ \delta p_x^2 \\ \delta p_y^2 \\ \vdots \\ \delta p_x^{n-2} \\ \delta p_y^{n-2} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & 0 & \dots & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \delta u_x^1 \\ \delta u_y^1 \\ \delta u_x^2 \\ \delta u_y^2 \\ \vdots \\ \delta u_x^{n-2} \\ \delta u_y^{n-2} \\ \delta u_x^{n-1} \\ \delta u_y^{n-1} \end{pmatrix} \tag{3.2.4}$$

which has a simple, obvious and easily computed solution to (3.2.4).

$$u_x^i = p_x^i - p_x^{i-1} \tag{3.2.5}$$

$$u_y^i = p_y^i - p_y^{i-1} \tag{3.2.6}$$

The discretized backbone curve is then used, via a fitting procedure, to compute the local actuator displacements which implement the desired deformation. Figure 3 displays a PSM deformation of a 30 DOF variable geometry truss manipulator (kinematically identical to the real system described in Section 4) in response to an impinging obstacle. The backbone curve is approximated by 100 segments. The manipulator is originally in a straight configuration, which locally deforms to avoid a simulated obstacle in Figure 4. In this simulation, the response function is shaped like a triangle.

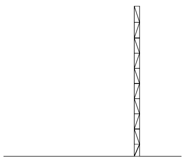

Figure 3: Original

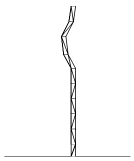

Figure 4: Resulting

3.3. Extended PSM

Due to mechanical limitations of the robot, such as joint limits, a real hyper-redundant manipulator has a limited amount of local extensibility. That is, $\|\epsilon\| < \Upsilon$ where Υ is the limit of local extensibility of the manipulator. This means that at any point, there is a fixed range over which the backbone can expand or contract relative to a fixed reference state.

The original PSM assumes that there is infinite local extensibility, which is unrealistic for actual robots. In the example in figure 5, an object becomes unacceptably close to the robot, which locally moves away from the object using an LSRF. However, the object continues to move towards the already deformed robot, which wants to move further away locally from the object, using the same LSRF. Although a backbone curve is determined in this situation, it is not likely that a real mechanism can fit this curve because this backbone requires a lot of local extensibility from the manipulator. In this case, the $\|\epsilon\| < \Upsilon$ constraint was violated because $\|\delta\vec{u}\,(s_i,t)\| > \Upsilon$.

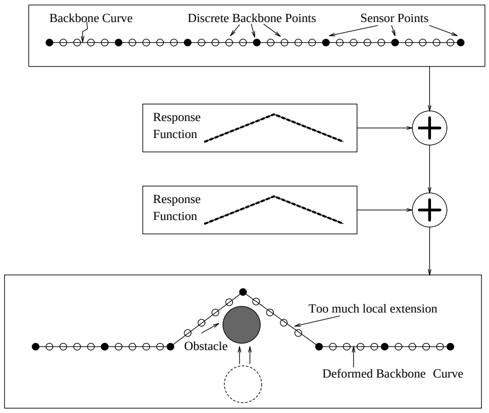

Figure 5: Same Response

So, a new LSRF has to be used which utilizes the extensibility of neighboring regions on the backbone curve, while not using any local extensibility from the already deformed section (i.e. maintaining the constraint $\|\epsilon\| < \Upsilon$). This is called "sucking" extensibility from other parts of the robot snake. In figure 6, the local extensibility limit is not exceeded, and the robot is still able to accommodate for the object's displacement.

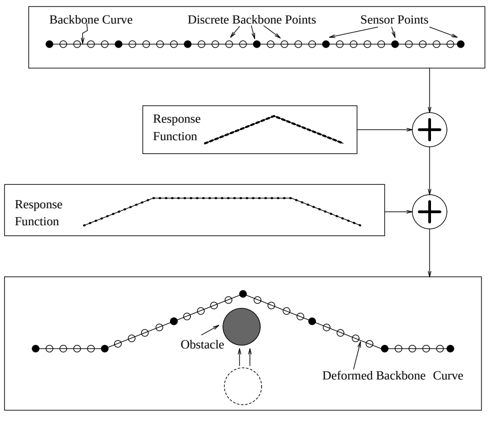

Figure 6: Modified Response

4. Experimental Setup

To prove the feasibility of the proposed algorithms, a distributed sensor system was developed for the 30 degree-of-freedom hyper-redundant robot system described in [ChB92b]. Figure 7 shows the structure of this testbed. This section describes the test bed structure in detail.

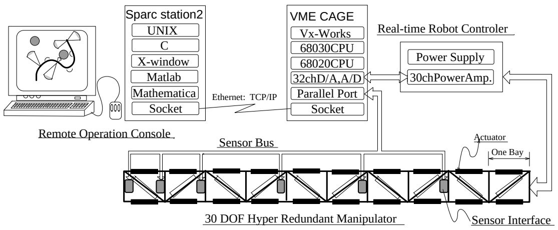

Figure 7: Hyper Redundant Robot Test Bed

4.1. Hyper redundant manipulator and control system

The hyper-redundant manipulator is a modular Variable Geometry Truss design [Ch]. The 30 degree-of-freedom (DOF) planar robot consists of ten modules (also called bays) of 3 DOF each (Fig. 7). Each DOF consists of a D.C. servo motor which drives a lead screw. Each lead screw is instrumented with a linear potentiometer. The real time system controller is based on a VME-bus multiprocessing computer, currently consisting of two Heurikon (68030 and 68020) single board processors, and the VxWorks real time operating system. One processor is dedicated to the closed loop feedback control of the actuator positions. The other processor is dedicated to processing of sensor data and real time computation of the PSM algorithms.

To enable flexible, modular, and easily expandable experimentation with sensor based planning, a novel 34 wire "Sensor Bus" architecture was developed for the sensor system. One end of the sensor bus is connected to the PSM processor via a parallel port. The sensor bus consists of an eight bit outgoing data path, a four bit status line, a two bit strobe and one interrupt request line. The data path and the two strobe lines enable the CPU to access up to 256 sensors and to send eight bits of information to the sensor peripherals for possible sensor control purposes. The interrupt request line is connected to the hardware counter on the CPU board so that accurate timing measurements can be made in real time.

Sensors can be added to the system via "Sensor Interface Modules." This module decodes the sensor bus address and generates signals to control sensors. Up to two ultrasonic sensor modules and six sets of sensors which produce data with 4 bit (or less) quantization can be controlled. Currently, only four infrared sensors per board are present, though up to eight infrared sensors and eight mechanical switchs can be directly connected. The sensor interface module circuitry is mounted on a printed circuit board which is 15cm by 12cm in size. Fig. 8 shows a photograph of the sensor interface module. The sensor bus is physically connected in the bottom of the module in a daisy-chain fashion. In the figure, two ultrasonic transducers are shown above the module. Also the infrared proximity sensor is shown on the side of the module.

Figure 8: Sensor Interface Module

4.2. Sensors

Currently, the robot has two types of sensors: infrared (IR) and ultrasonic (US). There are five sets of two US sensors. Each set is rigidly attached to alternate bays so that each sensor points outward from the backbone curve (or the centerline of the mechanism). An additional US sensor will be mounted at the front of the mechanism, pointing forward. Twenty IR sensors, again two at each the ten bays facing outward, are currently mounted on the snake. In the near future, twenty-four additional IR sensors will be added, two more at each bay, and four in front. Fig. 9 schematically shows how these sensors are distributed throughout the mechanism.

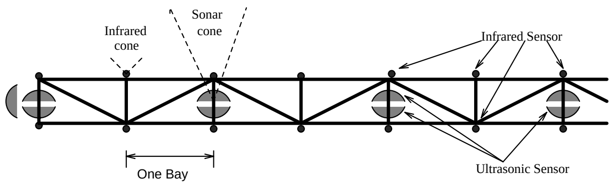

Figure 9: Sensor Arrangement

Electrostatic type ultrasonic sound transducers determine distance by measuring the time of flight of the ultrasound pulse leaving the transducer, bouncing off an object and returning to the sensor. A 50kHz sonar wave burst is transmitted when the sonar-ranging module is triggered [Ci]. The ranging module output is connected to the sensor bus interrupt request line. The echo return time is computed by the CPU hardware counter. Since the sixteen bit resolution distance measurement result is read from the hardware counter, no result is ever sent through the sensor bus. This design greatly simplifies the hardware required.

The US sensors are activated sequentially (at 16 millisecond intervals) to prevent interference between sensors. These sensors are calibrated to measure distances ranging from 10cm to 2.5m, with a 2% accuracy. There is about twenty degrees of conical ambiguity for direction, because of the transmitting beam pattern of the transducer [Ci]. In this work, it is assumed that the obstacle lies along the cone's centerline, which is locally normal to the backbone curve at the point of sensor attachment.

The IR sensors yield binary proximity information– i.e., the presence or absence of the obstacle in some pre-set range. An infrared LED emits modulated infrared light, and if an obstacle is near the robot, the IR sensor will detect the reflected light. The range of the IR system can be adjusted by setting potentiometers on the sensor boards. Currently, the IR system is set up to detect the presence of obstacles up to four inches away from the robot. Like the US, the location of an obstacle is not precisely known, but lies somewhere in a cone emanating from the IR sensor.

Each sensor has its own advantage. The IR sensors have a very fast response and can be sampled at extremely high rates. They are thus suitable for the PSM system. The US sensors provide proportional obstacle distance, rather than binary proximity information. They are thus more useful for accurate planning. However, since the US sensors are sequentially polled to prevent interference, the minimum sampling period is 176 milliseconds. The IR sensors are sequentially polled in a similar fashion, but at a significantly higher rate. To maximize the use of both types of sensors, the sensor interface module is designed to operate both US and IR sensors simultaneously in different intervals.

4.3. Remote Operation Console

The real time computers are connected to Sun workstations via the ethernet. Via software sockets, information can be transferred through the ethernet between the real time computer running VxWorks and the Sun workstations running Unix. C programs and many software packages, such as Matlab, are able to directly communicate with the real time computers via the sockets. Therefore, these programs can control the snake. The FSM and higher levels of control are implemented on the SUN workstation.

Experimental robot control programs are developed in a combination of C and Matlab. Via an X-Window interface, these programs graphically display and continually update the robot's configuration and sensor measurements. Fig.11 shows the X-Window operation console window. In addition, many motion planning and sensing commands can be executed using a graphical menu interface. End-effector via points of a hyper-redundant trajectory can be specified by a mouse, and the trajectory is then executed by the real-time system.

Figure 10: Infrared Sensor

In addition to graphically depicting the current configuration of the manipulator, this system displays US and IR sensor measurements. The solid cones emanating from the manipulator represent US sensor data. In this representation scheme, the closet point to an obstacle in the sensor beam pattern lies somewhere on the distal arc of the cone. The dashed arcs much closer to the mechanism indicate that the IR sensors have detected nearby obstacles at these locations.

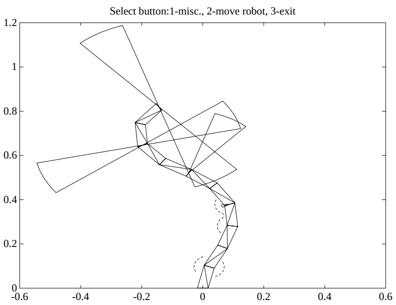

Figure 11: Operation Console

5. Results

The PSM algorithms described in Section 3.2 and 3.3 have been implemented on our hyper-redundant robot test-bed. Photographs of two experiments are shown below. In the first experiment, the backbone curve, dictated by some high level planner, was a straight line. Two obstacles were moved into an unacceptably close proximity (about 10cm or 4in) to the mechanism, and the manipulator locally deformed away from each obstacle while maintaining constant end-effector position. Truthfully, the end-effector was displaced slightly from its original position (less than a 1 inch displacement over a distance of $\sim$16 feet). The current implementation of the discrete approximation algorithm employs only IR sensors, because there are many more IR sensor distributed along the snake. See figure 12.

In the next experiment, again, the backbone curves starts off as a straight line. See figure 13. One obstacle was moved unacceptably close to the robot which resulted in the mechanism moving, as it did in the first experiment. See figure 14. Then, the object was moved sufficiently close to the deformed robot, passing through the original backbone curve, and the manipulator still deformed away. See figure 15. Such a large local deformation would not have been possible with the original PSM.

Th second experiment showed the local shape modification capability of the new PSM algorithm proposed in this paper. In real time, the old PSM reliably works, when the actuators in the section of the robot that is deforming are not near their joint limits. In such a case, the new PSM is the same as the old PSM. As actuators' limits are approached, local deformation may become infeasible, and this is where the new PSM becomes useful. The new planner uses extensibility from neighboring actuator displacements along the manipulator so, the robot can still locally deform. However, once all the actuators reach there limit, i.e. all the extensibility is used up, the robot has to report to a higher level planner in order to accommodate for all the constraints in the environment. This ability is currently being implemented in our system.

Figure 12: First Experiment

Figure 13: Second Experiment

Figure 14: Intermediate

Figure 15: Final

6. Conclusion

In this paper, a local sensor based planning method for hyper-redundant robots is extended. This method is based on a backbone curve kinematic framework. In the previous work, the limit of local deformation was limited to the extensibility over a fixed portion of the robot. In this paper, in order to better compensate for objects penetrating the backbone curve, extensibility was used over a variable portion of the robot.

This method was implemented on an actual 30 DOF hyper-redundant manipulator test bed. An innovative sensor bus architecture and a graphical programming and display interface were reviewed. Experiments using this system showed the applicability and effectiveness of the proposed method to real hyper-redundant manipulators. A reasonable amount of computer power was required for real-time implementation of these algorithms.

As suggested in the previous section, we are currently working to improve the communication between low level planners and a high level planner so that the robot can better interpret and react to exceptional conditions indicated by the PSM level. In addition, we intend to develop better sensor function methods which properly combine ultrasonic and infrared sensor readings from adjacent sensors. The highly distributed nature of sensors on a hyper-redundant mechanism also point to the need for new theories on deploying and using massively redundant sensor arrays. Finally, future work will focus on using sensor data for higher level, i.e. global, hyper-redundant robotic planning.

Acknowledgements: This work has been supported by National Science Foundation Presidential Young Investigator Grant MSS-9157843, and by the office of Naval Research Young Investigator Award N00014-92-J-1920 and N00014-93-1-0782. The authors would also like to thank Mr. Nobuaki Takanashi of the NEC Corporation.

7. References

[CaLi] Canny J. F., Lin, M. C. "An Opportunistic Global Path Planner," *Proc. IEEE Int. Conf. on Robotics and Automation*, Cincinnati, Ohio, May 14-17, 1990, pp. 1554-1559.

[Ch] Chirikjian, G. "Theory and Applications of Hyper-Redundant Robotic Manipulators," PhD Thesis, California Institute of Technology, Pasadena, Ca, 1992.

[ChB90a] Chirikjian, G.S., Burdick, J.W., "An Obstacle Avoidance Algorithm for Hyper-Redundant Manipulators," *Proc. IEEE Int. Conf. on Robotics and Automation*, Cincinnati, Ohio, May 14-17, 1990, pp. 625-631.

[ChB90b] Chirikjian, G.S., Burdick, J.W., "Kinematics of Hyper-Redundant Manipulators," *Proc. ASME Mechanisms Conference*, Chicago, IL, DE-Vol. 25, pp. 391-396, Sept. 16-19, 1990.

[ChB90c] G.S. Chirikjian and J.W. Burdick, "Applications of Hyper-Redundant Manipulators for Space Robotics and Automation," *Proc. Int. Symposium on Artificial Intelligence* *and Robotics Applications to Space*, Kobe, Japan, November, 1990.

[ChB91a] Chirikjian, G.S., Burdick, J.W., "Parallel Formulation of the Inverse Kinematics of Modular Hyper-Redundant Manipulators," *Proc. IEEE Int. Conf. on Robotics and Automation*, Sacramento, California, April, 1991, pp. 708-713.

[ChB91b] Chirikjian, G.S., Burdick, J.W., "Kinematics of Hyper-Redundant Locomotion with Applications to Grasping," *Proc. IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, April, 1991.

[ChB92a] Chirikjian, G.S., Burdick, J.W., "On the Determination of Kinematically Optimal Hyper-Redundant Manipulator Configurations," *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, France, May 10-15, 1992.

[ChB92b] Chirikjian, G.S., Burdick, J.W., "Design, Implementation, and Experiments with a 30 Degree-of-Freedon Hyper-Redundant Robot," *Proc. Int. Symp. Robotics and Manufacturing*, Albuquerque, NM, Nov., 1992.

[ChB92c] Chirikjian, G.S., Burdick, J.W., "A Modal Approach to Hyper-Redundant Manipulator Kinematics," submitted to *IEEE Trans. on Robotics and Automation.*

[ChB93a] Chirikjian, G.S., Burdick, J.W., "The Kinematics of Planar Hyper-Redundant Robotic Locomotion," submitted to *IEEE Trans. on Automation and Robotics.*

[ChB93b] Chirikjian, G.S., Burdick, J.W., "Curvilinear Hyper-Redundant Robotic Locomotion Over Uneven Terrain, with Applications to Grasping," submitted to *IEEE Trans. on Automation and Robotics.*

[ChB93c] Burdick, J.W., Radford, J., Chirikjian, G.S., "A 'Sidewinding' Locomotion Gait for Hyper-Redundant Robots," submitted to *IEEE Int. Conf. on Robotics and Automation*, Atlanta, GA, May, 1993.

[Ci] Ciarcia, S. "An Ultrasonic Ranging System." *Byte Magazine*, October, 1984, pp.113-123.

[HiU] Hirose, S., Umetani, Y., "Kinematic Control of Active Cord Mechanism With Tactile Sensors," *Proceedings of Second International CISM-IFT Symposium on Theory and Practice of Robots and Manipulators*, pp. 241-252, 1976.

[Gr] Greville, T.N.E. "The Pseudoinverse of a Rectangular or Singular Matrix and its Applications to the Solutions of Systems of Linear Equations," SIAM Review 1(1),1959,pp.38-43.

[Lat] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Boston, 1991.

[Li] Liegeois, A. "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms," IEEE Trans. Systems, Man and Cybernetics, 1977, Vol. SMC-7, No.12, pp.868-871.

[LoWes] Lozano-Perez, T. and Wesley, M.A. "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications ACM*, Oct. 1979, Vol ACM 22, pp.560-570.

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

[Nak] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task Priority Based Redundancy Control of Robot Manipulators," *International Journal of Robotics Research*, vol. 6, 1987, pp. 3-15.

[Pe] Penrose, R. "On Best Approximate Solutions of Linear Matrix Equations," Proc. Cambridge Phill.,1956, Soc.52,pp.17-19.

[ReLu] Reznik, D. Lumelsky, V. "Motion Planning with Uncertainty for Highly Redundant Kinematic Structures I. 'Free Snake' Motion," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Raleigh, N.C., 1992.

[Sh] Sharir, "Algorithmic Motion Planning for Robots."

[TCB] Takanashi, N., Choset, H., and Burdick, J.W. "Local Sensor Based Planning for Hyper-redundant Manipulator," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, 1993.

[Yo] Yoshikawa, T. "Analysis and control of robot manipulators with redundancy," Robotics research: The First International Symposium, ed. Brady, M. and Paul, R. Cambridge:MIT Press,1984, pp.735-747.