Sensor-Based Roadmaps for Motion lpqPlanning for Articulated Robots in Unknown Environments: Some Experiments with an Eye-in-hand System

Yong Yu yongyu@cs.sfu.ca
Kamal Gupta kamal@cs.sfu.ca
School of Engineering Science
Simon Fraser University
Burnaby, BC, CANADA, V5A 1S6

Abstract

We present a real implemented "eye-in-hand" test-bed system for sensor-based collision-free motion planning for articulated robot arms. The system consists of a PUMA 560 with a triangulation based area-scan laser range finder (the eye) mounted on its wrist. The framework for our planning approach was presented in [24]. It is inspired by recent motion planning research and incrementally builds a roadmap that represents the connectivity of the free configuration space, as it senses the physical environment. We present some experimental results with our sensor-based planner running on this real test-bed. The robot is started in completely unknown and cluttered environments. Typically, the planner is able to reach (planning as it senses) the goal configuration in about 7 - 25 scans (depending on the scene complexity), while avoiding collisions with the obstacles throughout.

1 Introduction

Our research is concerned with sensor-based motion planning (MP) for articulated robot arms with many degrees of freedom. The arm (called robot from here on), equipped with an "eye" sensor (See Figure 1) that gives distance of the objects from the sensor is required to plan and execute collision-free motions in an environment initially unknown to the robot. Note that the planning space (the configuration space) and the sensor space (physical environment) are very different. This is the fundamental distinction from sensor-based planning for a mobile robot (treated as a single point as in [13, 7]), where sensing and planning basically take place in the same (physical) space. Therefore, these sensor-based approaches for point robots may not be directly extended to the case of articulated arms via the standard "convert to C-space" argument, since, the sensor senses in physical space and will not, in general, correspond to sensing in C-space (See [25] for more on this)[1].

In [24], we presented an approach (and some initial steps toward this goal) for such sensor-based motion planning for articulated robots. Our approach, based on a recent paradigm in solving classical MP, is to capture the connectivity of robot's free configuration space ($\mathcal{C}_{free}$) in a finite graph structure (called a roadmap) [2], [1], [15], with each node representing a free robot configuration, and an edge between two nodes represents that a simple local planner can find a collision-free path between the corresponding robot configurations. Since this roadmap is constructed without explicitly representing the C-obstacles, it is particularly useful as a representation for high dimensional configuration spaces. The crux of our methodology is to *incrementally* construct this roadmap as the sensor senses new free space in the physical environment. The (evolving) roadmap reflects the connectivity of known $\mathcal{C}_{free}$ within which the robot carries out its motion to further sense the physical environment. The $\mathcal{C}_{free}$ (and the roadmap) expand as the sensor senses new free space in the physical world. This process is repeated until either the final goal is reachable from one of the nodes in the graph, or the goal is declared unreachable. Our current implementation adapts the PRM approach of [15]. We call it SB-PRM (sensor-based probabilistic roadmap)[2]. In this paper, we present a real implemented "eye-in-hand" test-bed system, and experimental results with the SB-PRM planner running on this test-bed. The details of the test-bed including hardware description and the system architecture are presented in Section 3. We then present an outline of the SB-PRM algorithm (mostly a summary of what was reported in [24]. Section 5 provides representation details buried in the succinct description of the algorithm. Experimental results are detailed in Section 6, followed by a summary in Section 7.

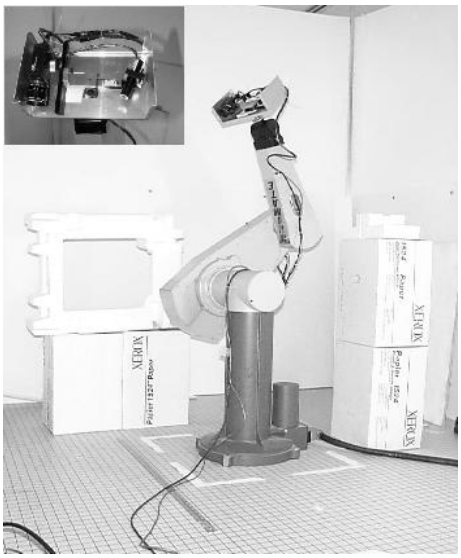


Figure 1: The experimental testbed for sensor-based MP. The sensor, a triangulation based area scan laser range finder, is mounted on the PUMA wrist. Inset shows an enlarged view of the sensor with the camera on the left and the laser striper on the right. See text for a complete explanation of the system.

[1] Our argument applies to an "eye" type sensor. A "skin" sensor as in [11] that senses along the entire manipulator geometry, or a range sensor distributed along the entire robot geometry, as assumed in [4], indeed allows the point robot abstraction for sensor-based case as well.

[2] We are also investigating the adaptation of the ACA aaproach of [2] for the sensor-based case (SB-ACA). A preliminary version is reported in [18].

2 Related Work

Renton et al presented a Plan-N-Scan system comprising a PUMA with a wrist mounted range sensor (very similar hardware to ours) [19]. Their main purpose, however, was to explore the workspace (a part or whole). In order to reach a desired scanning configuration, motion planning was accomplished with a simple A^* algorithm, and therefore somewhat limited. They successfully demonstrated the ability to scan workspace volumes. We believe that our planning would perform significantly better in tight and cluttered environments. For instance, the initial assumed free region around the robot is much bigger in their case than in ours. Furthermore, the roadmap has the advantage that once built, it can be used to quickly answer further path planning queries [2, 15].

Kruse et al describe a sensor-based system for map building. Their simulated system also has a range sensor on the end-effector of a robot arm and they also use a roadmap for the planning component [9]; however, there are key differences. Theirs was a simulation only; their main concern was map-building and they employed rating functions to choose configurations that are for promising for further sensing; path planning is used mainly as a tool to get to these promising configurations. Their simulated examples appear to be mostly for quite sparse environments. They simply repeatedly call the roadmap based planner developed in ([15]). The extension of this roadmap, as the robot senses more of its environment is ad hoc in their approach. In fact they mention that "... after several extensions, the graph tends to become large and degenerate ... a complete recalculation of the graph is performed". Our main emphasis, on the other hand, is on motion planning and an integrated approach to incrementally build the roadmap.

A great deal of work has been done on sensor-based planning for mobile robots [7, 14] with "eye" type sensors (camera or range), however, as mentioned above, with this type of sensor, this body of work is not directly applicable to manipulator arms. Even otherwise, most of these algorithms are applicable to 2-dof (at most 3-dof) systems. Lumelsky has applied these approaches to 2-dof manipulators and to the very specific case of a 3-dof arm with all prismatic joints [12]. Note that his approach requires a whole-arm sensitive sensor (probably more complex than the commercially available range-scanners). Other sensor-based approaches, on the other hand assume abstract sensors that provide distances in C-space [20].

3 Sensor-Based Planning Testbed

A main purpose behind the development of this test-bed is to be able to test different algorithms for sensor-based motion planning for robot manipulators. We describe the physical system in Section 3.1 and the software architecture in Section 3.2.

3.1 The Physical System

The physical system test-bed, shown in Figure 1 consists of a six-dof PUMA 560 equipped with a wrist-mounted area-scan triangulation based laser range scanner. The decision to mount the scanner on the robot end-effector, thus providing a six-dof scanning ability, was motivated by the additional maneuverability for sensing. The scanner can scan from any point in the reachable workspace of the robot and in almost any direction (subject to joint limits).

The physical sensor is a triangulation-based, area-scan laser range finder mounted on the end effector of the robot (See Soucy et al [22] for details of a similar sensor). The scanner hardware is composed of a solid state laser striper, a CCD camera, and a framer grabber residing in a PC. The camera and the laser striper are mounted on the PUMA end-effector with their (the camera and the striper) relative positions fixed. The last joint (joint 6) of the Puma robot is used to scan the laser light stripe across the scene. As this joint rotates, the image frame grabber continuously takes images.

The specification of the sensor are as follows. The sensor is configured to sense in the range of 30cm - 105cm from the camera. Within this measurable range, the sensor is accurate to about 0.5cm (in all three coordinates). This range and resolution is just adequate for our Puma robot whose positioning accuracy is roughly 1cm - 3cm and the workspace radius is about 1m. The field of view of the sensor in the direction perpendicular to the laser strip (joint 6 rotating direction) is not limited. However, in the direction of the laser stripe, it is limited to 35 degrees. The scanning time depends on the scanning resolution and the field of view. In our typical application, it takes about 35 seconds to get a 256 by 256 resolution range image.

3.2 System Architecture

The main component blocks of the testbed are: (i) the robot server (ii) the sensor server, and (iii) the planner server. The three processes, the planner, the sensor and the robot communicate with each other through sockets using TCP/IP. Such an implementation keeps the system modular and hence easier to maintain and add to. Furthermore, to achieve fast response and facilitate intensive data transmission between the processes, we use a switched hub for the system to screen the outside network traffic. The robot server exchanges data with the robot controller through a parallel port. See Figure 2 for the basic system architecture and data flow. We now describe the system details for each of the processes.

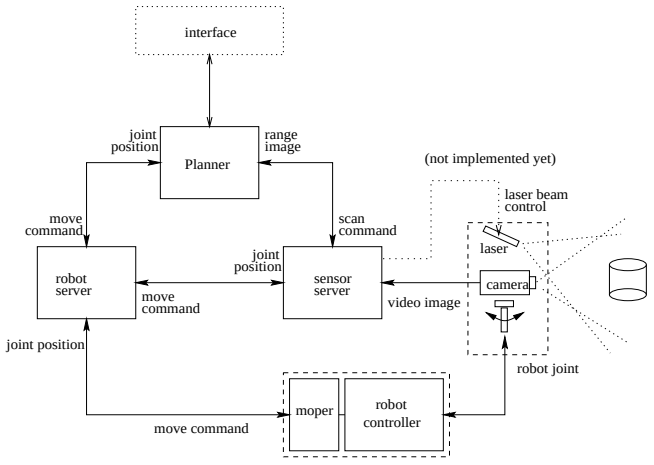


Figure 2: The sensor-based testbed system configuration and data flow.

The Robot Server

The robot server is essentially a robot control program that runs on a host computer, an SGI INDIGO workstation. Its application control interface is RCCL (Robot Control C Library) [8]. It communicates with a program (MOPER) that

runs on the Unimation controller (an LSI 11). MOPER dispatches the joint setpoint commands to individual (original Unimation) joint servo controllers.

In order to communicate with other processes in our system, for instance, the sensor process and planner process, the robot server runs on the same machine as the RCCL program. The robot server accepts most of the RCCL commands and the API is kept the same with only minimal necessary alternation.

The Sensor Server
A range image daemon process running on the PC analyzes (using triangulation) the laser stripes in the images, and using the joint 6 value and each scan, it outputs a range image, an array of points in the physical 3-dimensional workspace. The daemon communicates with other processes on the net with TCP/IP protocols through two sockets. The first socket accepts commands from client process and transmits the range images to the client. These commands include the specification of field of view, interpolation (between range data points) on/off etc.

The second socket communicates with the robot server (in our case, as mentioned above, the robot server resides on the SGI INDIGO). The scanning mechanism is started and synchronized through this socket.

Planner Server
The key task of the planner is to find a collision free path from a given start configuration to a given goal configuration. Currently, the start and goal are specified by the user in configuration space. However, as mentioned earlier, the sensor-based planner (details in Section 4) generates robot motions to explore (scan) the environment as well as to direct the robot toward the goal position. Note that the physical sensor (the scanner) is mounted on the robot (and hence a part of the robot) and must not collide with obstacles. The planner server runs on the PC, the same machine that the sensor server runs on. A main reason behind this choice is that since the planner needs to receive the range image (each image is more than 2 Mbytes) from the sensor server, the two should reside on the same machine.

Data flow
There are several major data flows in the system: (i) planner sends robot move command to the robot server and receives the current robot configuration (joint position) data from it (ii) planner sends scan commands (along with the scanning configuration of the robot) to the sensor server which, in turn, generates and sends move command to the robot server. The sensor server returns the sensed range image along with joint positions (received from the robot serve) back to the planner.

4 The Algorithm: SB-PRM

In this section we briefly describe our planning algorithm, SB-PRM. Details of the algorithm have been presented in [24], here we present a brief overview. The two key assumptions are: (i) the robot work space is static, with the robot itself being the only moving object, and (ii) a small region of physical space (a cuboid) surrounding the robot in its initial configuration is free. This latter requirement is important because otherwise even the very first move may not be possible after the initial scan.

4.1 Definitions and Notation

Let $\mathcal{C}$ denote n-dimensional C-space of the robot and Let q denote a robot configuration, or a point in the C-space. $\mathcal{P}$ denotes the physical space (the 3-D Euclidean space). Correspondingly, $\mathcal{C}_{free}$ is the free C-space and $\mathcal{P}_{free}$ is the free physical space the sensor has cumulatively seen at a certain stage of the planning process (note that this planning and sensing). An obstacle is defined as the "surface" elements of the range data returned by the scanner. We use $\mathcal{P}_{obstacle}$ to represent the obstacles (in the physical space) at a certain stage in the planning process. Note that at a given stage in the planning process, $\mathcal{P}_{obstacle} \bigcup \mathcal{P}_{free}$ may not equal the whole physical space $\mathcal{P}$. This is because there may be some portion of $\mathcal{P}$ that is unknown. $\mathcal{A}$ represents the robot. $\mathcal{A}(q) \subset \mathcal{P}$ denotes the physical space occupied by the robot at configuration q.

We assume that a routine *scan*() returns the sensed free region. However, since the sensed free region (in the current scan) may overlap part of existing $\mathcal{P}_{free}$, an additional computation is needed to calculate $\Delta\mathcal{P}_{free}$, the additional increment in free physical space, i.e., ($\Delta\mathcal{P}_{free}$ = *scan*()$-\mathcal{P}_{free}$). Corresponding to $\Delta\mathcal{P}_{free}$, $\mathcal{C}_{free}$ is augmented by $\Delta\mathcal{C}_{free}$. Formally, let $\mathcal{C}_{free}$ = $\mathcal{M}$ ($\mathcal{P}_{free}$), where $\mathcal{M}$ is the mapping from $\mathcal{P}_{free}$ to $\mathcal{C}_{free}$, then $\Delta\mathcal{C}_{free}$ =$\mathcal{M}$ ($\mathcal{P}_{free}$ $\bigcup\Delta\mathcal{P}_{free}$)$-\mathcal{M}$ ($\mathcal{P}_{free}$), where $-$ denotes set difference.

A landmark, denoted by l, is a particular robot configuration (along with some additional information as will become clear later). q_l denotes the configuration corresponding to a landmark l. L denotes the set of all the landmarks. We distinguish between three types of landmarks: *white* landmarks, similar to the landmarks in [15, 2], are those that belong to $\mathcal{C}_{free}$; *black* landmarks are those that correspond to the robot in collision with a *sensed* obstacle; and, *gray* landmarks are those that correspond to part of the robot in "unknown" environment, i.e., at the configuration corresponding to the landmark, the robot does not intersect with any sensed obstacles, and does not lie completely in $\mathcal{C}_{free}$ either (see Figure 3).

$\mathcal{C}_{free}$ is characterized by a *roadmap* R. The *roadmap* R is defined as an undirected graph R = (N, E) with its nodes as the white landmarks. Note that N includes all the white landmarks and is a subset of L. Two nodes, n_1 and n_2, in R are connected by an edge in E if *local_planner(n_1,n_2)* returns a simple collision-free path from q_{n1} to q_{n2}. We store graph R related data and the color in the landmarks. The *local_planner()* simply tries to execute a discretized straight line in C-space. It checks for collision detection at each discretized point. If any of the points along the straight line results in collision, *local_planner()* returns failure.

4.2 Incrementally Building Roadmap

The basic idea behind our algorithm is to *incrementally* generate the roadmap R that represents the connectivity of $\mathcal{C}_{free}$. As each $\Delta\mathcal{P}_{free}$ is obtained by the range sensor, $\mathcal{C}_{free}$ is expanded towards the goal. New landmarks are generated randomly to expand the roadmap and update gray and black landmark lists. Recall that gray landmarks represent points in the "unknown" $\mathcal{C}$ and could be conceptually thought of as a "wavefront" of $\mathcal{C}_{free}$. These gray and black landmarks are updated (some gray landmarks may be updated to white ones) as new $\Delta\mathcal{P}_{free}$ is added from each new scan. The update process can be thought of as the wavefront of $\mathcal{C}_{free}$, represented by the gray landmarks, expanding toward the goal.

The motion planning algorithm can now be outlined as follows:

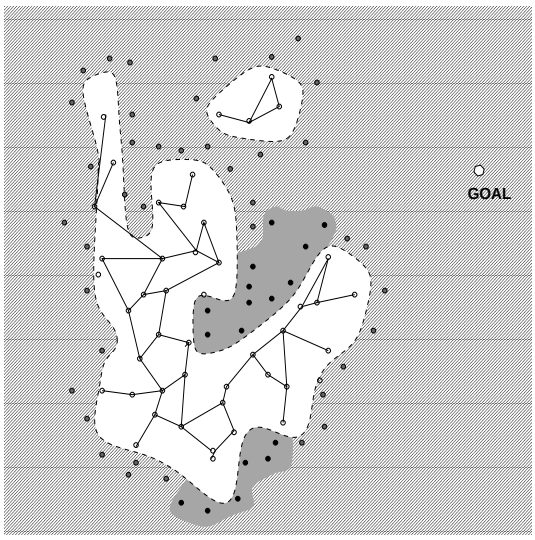


Figure 3: Three types landmarks (all denoted by ∘) are shown above. White region is known $\mathcal{C}_{free}$ and the landmarks in this region are white landmarks. The *Roadmap* is the graph shown in this region. The dark region indicates known C-obstacles and the landmarks in this region are black landmarks (close list). The grey region is the unknown C-space and the landmarks in this region are grey landmarks (open list). The grey and black landmarks form a "crust" enclosing $\mathcal{C}_{free}$.

Algorithm SB-PRM

$SB-PRM()$
  $n_{cur} \leftarrow n_0$
  $\mathcal{P}_{free} \leftarrow$ initial free region
  $\mathcal{P}_{obstacle} \leftarrow 0$
  Initialize R(N, E), $OpenList, CloseList$
  loop
    $newScan = scan()$
    $\Delta\mathcal{P}_{free} = newScan - \mathcal{P}_{free}$
    $updateRoadmap()$
    $SearchShortestPath$(R)
    $moveRobot(n_{scan})$
    $n_{cur} \leftarrow n_{scan}$
  loop end

$updateRoadmap()$
  update $\mathcal{P}_{obstacle}$
  $\Delta L \leftarrow$ randomly generated landmarks within $\Delta\mathcal{C}_{free}$
  $\Delta L \leftarrow \Delta L \bigcup (OpenList \bigcap \Delta\mathcal{C}_{free})$
  /* new landmarks & some in $OpenList$ examined */
  $\forall l \in \Delta L$,
    if $colorOf(l) = WHITE$
      $\forall n \in$ N
      /*connect new white landmarks to roadmap*/
        if n is within certain distance from l
          if *local_planner(l, n)* = true
            E $\leftarrow$ E $\bigcup (l, n)$
    if $colorOf(l) = BLACK$
      $addToList(l, CloseList)$
    if $colorOf(l) = GRAY$
      $addToList(l, OpenList)$
$\forall n \in$ N, and $connected(n_{cur}, n) = TRUE$
  if *local_planner(goal, n)* = true
    report success and exit
$l_{exp} \leftarrow$ closest landmark to *goal* in $OpenList$
/* l_{exp} is the next landmark to be investigated */
$\forall n \in$ N and $connected(n_{cur}, n) = TRUE$
  $n_{scan} \leftarrow bestViewPointfor(l_{exp})$

5 Representations

We now detail some of the underlying representations and computations that are only mentioned in the abstract in the algorithm stated above.

An octree representation is used for the physical space. We maintain two octrees, one for the free physical space, called $\mathcal{P}_{free}$ -octree, and the other for the obstacles, called $\mathcal{P}_{obstacle}$ -octree. In the $\mathcal{P}_{free}$ -octree, white nodes represent free space, and black nodes represent either obstacles or unknown space. Each black node is augmented with an additional field that classifies parts of it as obstacle or unknown. Therefore, a black node is not a terminal node as in the standard octree, but could be further decomposed. During the planning process, $\mathcal{P}_{free}$ increases by $\Delta\mathcal{P}_{free}$. The union of all the $\Delta\mathcal{P}_{free}$ and the initial free space is the current $\mathcal{P}_{free}$. The octrees are constructed and operated efficiently and in an on-line manner from sensed range images by our recently developed algorithms [23]. Furthermore, efficient algorithms are also available to manipulate these representations [21].

The basic collision detection in our computations is based on checking if each face (the robot model is a CAD model composed of polyhedral primitives) of the robot intersects any black node in the octree. Due to hierarchical nature of an octree, such computations are very efficient (on average it takes a few milliseconds to detect if the robot is collision-free). Similar octree based collision detection has been previously reported in [16].

Note that $\Delta\mathcal{C}_{free}$ mentioned in the algorithm pseudocode is not explicitly computed. We simply ensure that at least a part of the robot when placed in the configuration corresponding to each newly generated landmark lies in $\Delta\mathcal{P}_{free}$.

bestViewPointfor() computes the "best" white landmark from which the next scan is to be made. Our current scheme first determines the grey landmark closest to the goal position. It then computes the intersection of the robot (as if it were placed at the grey landmark) with the unknown region. Let us denote the centroid of this intersection as c_{exp}. The algorithm then tries to position the sensor at a white landmark such that the centroid of the sensed region (the cone) is the closest to c_{exp}. Note that one could also use rating function approach of [9] to select the next viewpoint at increased computational cost.

As mentioned earlier, an obstacle is the "surface" of objects seen by the sensor. We calculate $\mathcal{P}_{obstacle}$ from $\mathcal{P}_{free}$. Note that $\mathcal{P}_{obstacle}$ is different from the boundary of $\mathcal{P}_{free}$. The latter differs from the former in following three ways: (i) it may include the boundary of shadows caused by occlusion of obstacles, (ii) it may include the boundary caused by the limit of sensing distance, and (iii) it may include the boundary caused by the robot itself (sensor may scan part of the robot). We determine $\mathcal{P}_{obstacle}$ from the boundary of $\mathcal{P}_{free}$ by excluding the spurious boundary caused by

the above three cases (We omit the details here for lack of space).

6 Experiments

We now present experimental results with SB-PRM on the real test-bed. We assumed a small region of $80cm \times 80cm \times 220cm$ surrounding the robot is free. Note that with the PUMA in its most "compact" configuration, the smallest cube enclosing the projection of PUMA on the x-y plane is 65cm $\times$ 65cm, implying that the assumed initial free region is indeed very tight. A typical task we tested is shown in Figure 7. The goal is in a very tight region, the closest object being only a few centimeters from the robot in its goal configuration.

Figure 7 shows a sequence of snapshots of the robot (a1,a2,a3, and a4 are actual pictures), the physical free space, $\mathcal{P}_{free}$ and the physical obstacles $\mathcal{P}_{obstacle}$, sensed by the robot (b and c are the internal models built by the planner), and the roadmap (landmarks) in the C-space. The robot is started in completely unknown and cluttered environments (Figure 7a1). The corresponding $\mathcal{P}_{free}$ is shown in Figure 7b1. The grey vertical cuboid corresponds to the initially assumed physical free space. Since the robot has not sensed anything yet, $\mathcal{P}_{obstacle}$ is an empty set (Figure 7c1). The initial set of landmarks (their projections in the 3-dimensional space comprising the first three joint angles of the PUMA) are shown in Figure 7d1. Note that the white landmarks are limited to a very thin disk that corresponds to the initially assumed cuboid $\mathcal{P}_{free}$ (Figure 7b1). For each iteration, the system takes a range image, generates new landmarks, updates old landmarks, updates $\mathcal{P}_{free}$ and $\mathcal{P}_{obstacle}$, determines the next scanning position, and moves to it. This process is shown in the next three sets in Figure 7, b,c and d. The goal configuration of the robot lies in between two cuboid boxes and above a flat horizontal surface (Figure 7d). In Figure 7c3 and c4, the largish gray region corresponds to this flat horizontal surface, and the vertical walls of the boxes correspond to other gray patches. In Figure 7d1,2,3and 4, dark gray (and larger) dots are white landmarks, light gray (and smaller) dots are gray landmarks, and black dots are black landmarks. One can see how the roadmap evolves as more scans are taken.

Figure 4 shows the connectivity of the final roadmap (when goal is reached). Note the robot has to pass through a very narrow (and quite complex) channel in order to reach the goal. In this case, the final roadmap consisited of 2411 total landmarks (392 white, 1359 black and 660 grey). The large number of black landmarks is due to the narrow regions in the workspace. The proportion of white, black and gray landmarks varies depending on experiment settings.

The run time for one iteration varies from 1 minute to 2.5 minutes depending on the number of landmarks generated and the complexity of the scene. Table 1 shows the breakdown of the time used in each part of the planner at a stage when there are about 1000 - 6000 landmarks. Typical robot movement in each iteration is composed of 3 - 7 primitive movements (straight line in C-space).

Figure 5 graphs the increase in number of landmarks and $\mathcal{P}_{free}$ as a function of number of scans. As expected, $\mathcal{P}_{free}$ and the total number of landmarks increase as more scans are taken. S1 and S2 are two successful examples. In each, the robot reachs the goal after 15 scans. In F1, the robot was not able to reach the goal within 30 scans. The

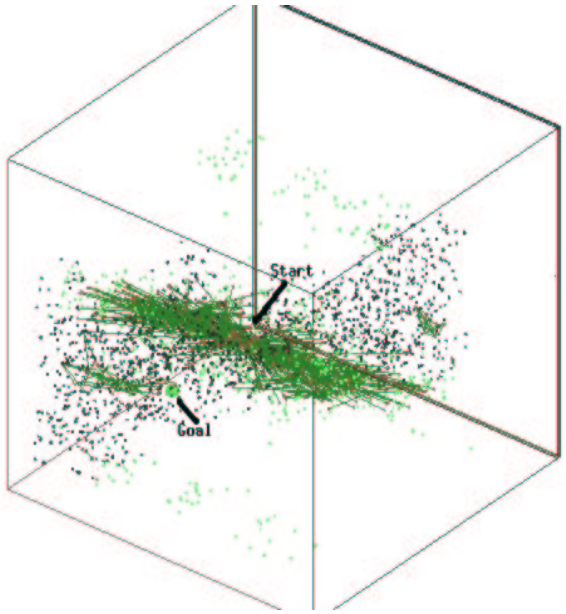


Figure 4: The final roadmap. An edge between two white landmarks is shown by a straight line between them. The robot has to pass through a very narrow (and quite complex) channel in order to reach the goal.

scan()	scanning and constructing octree	44 - 80 sec
updateRoadmap()	(1k - 6k landmarks)	20 - 60sec
moveRobot()	robot movement (3 -7 motions)	5 -30 sec
Total		69 - 160 sec

Table 1: Breakdown of run time for one iteration

failure was most likely due to the fact that the channel to the goal is very narrow. and more landmarks (and scans) may be needed. This problem (of finding narrow channels) is inherent in PRM type approaches, even for the classical model-based case[15]. Additionally, even for the same example, SB-PRM may fail to reach the goal within a given number of scans (30 scans), because of the probabilistic element (a key deficiency in such approaches). We ran the same experiment 24 times. The robot was able to reach the goal within 30 scans in about 80% of the cases (20 times). This success rate could be increased at a cost of increasing the density of landmarks, and correspondingly increased computation times.

An interesting observation is that scan 13 in experiment S1 and scan 12 in experiment S2 (both near the goal), the scanner sensed more free space than the previous ones. This is because robot's final goal was in a narrow region. This region could not be scanned before because of occlusions. Once this region is scanned, the robot is quickly able to reach the final goal.

The algorithm needs to maintain a uniform distribution of the landmarks in C-space. As pointed in [9], the distribution could easily degenerate in the planning process. Our

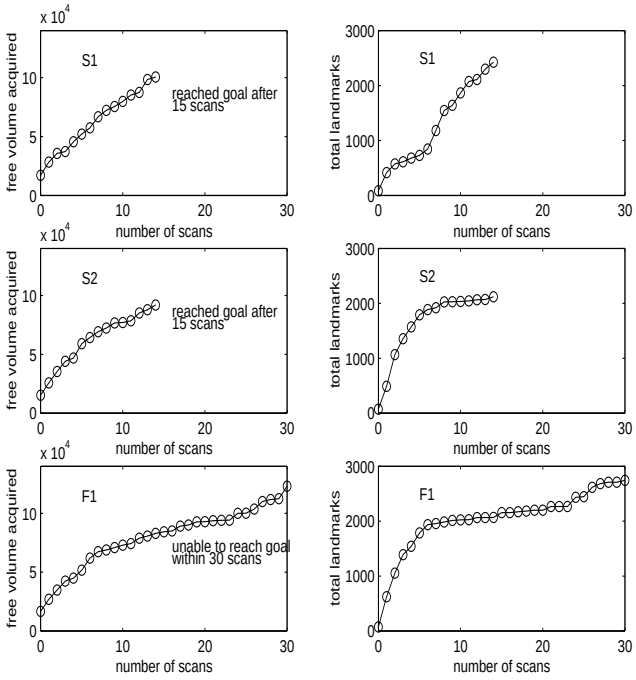


Figure 5: Evolution of $\mathcal{P}_{free}$ and total number of landmarks with successive iterations.

approach strives to generate new landmarks only in the new free region ($\Delta\mathcal{C}_{free}$) and near the (expanding) boundary (wavefront) of $\mathcal{C}_{free}$. This leads to a relatively uniform distribution of landmarks during the entire planning process. To show this, we need a measure of "uniformity" of distribution. Each landmark has a minimum distance associated with it, the distance to the closest landmark. We use the average these minimum distances as a measure of the deviation from uniform distribution [3]. The basic idea is that clustered landmarks will result in lower average minimum distance. The lowest value of this measure (most uniformly distributed) is given by $|\Gamma(k/2+1)|^{1/k}\Gamma(1/k+1)/\rho^{1/k}\pi^{1/2}$, where ρ is the density of landmarks and k is the dimension of the space, and Γ is the standard function.

Figure 6 shows this measure for a typical example, plotted as a function of number of iterations. The horizontal line represents the theoretical value for a random distribution. This suggests our landmarks are uniform distributed in C-space. The fact that our average minimum distance is somewhat greater than the theoretical value is due to the boundary effects. Landmarks on the boundary have greater chance of having a larger minimum distance. We can see the boundary effect fades away as more landmarks are added.

7 Summary

We presented a real implemented "eye-in-hand" system for sensor-based motion planning for articulated robot arms with many degrees of freedom in unknown environments. It consists of a PUMA 560 with a triangulation based area-scan laser range finder mounted on its wrist. The PUMA and the scanner are each controlled by their host machines, an SGI and a PC respectively and communicate with each other through sockets with TCP/IP protocols. The implemented sensor-based planner, SB-PRM incrementally builds

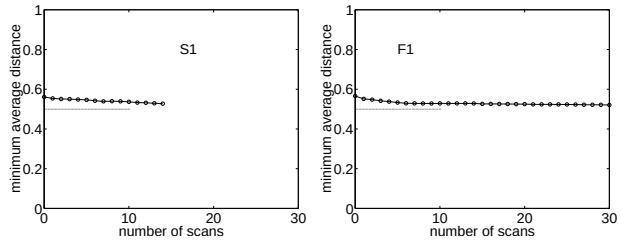


Figure 6: Average minimum distance between the landmarks with successive iterations.

a roadmap that represents the connectivity of the known free configuration space, as it senses the physical environment. SB-PRM adapts the probabilistic roadmap approach of [15] and was presented in detail in [24].

We ran several experiments in which the robot is started in completely unknown and cluttered environments. Typically, the planner is able to reach (planning as it senses) the goal configuration in about 7 - 25 scans (depending on the scene complexity), while avoiding collisions with the obstacles throughout. Each iteration of the planner (a scan, update or plan and move cycle) takes between 1 - 2.5 minutes depending on the scene complexity and the size of the roadmap; and the planner is able to plan a collision-free path in about and 7-25 scans to find the goal configuration.

There is much exciting work to be done. In the short term we hope to conduct further experiments and implement more efficient versions of basic computations (collision detection, for example). We expect this will give run time improvements of as much as order magnitude. We will also be investigating the adaptation of the ACA approach of [2, 1] for the sensor-based case (SB-ACA). A preliminary version is reported in [18]. Finally, we plan to formally address theoretical issues in sensor-based planning for articulated arms – issues such as completeness, explorability of space. As mentioned in the introduction, a key difference from the mobile robot case is that the planning space (C-space) and the sensing space (physical space) are different leading to several interesting complexities.

Acknowledgments

This research is being carried out in co-operation with National Research Council (NRC), Canada and with the International Submarine Engineering (ISE) Ltd. and is jointly funded by the National Science and Engineering Research Council (NSERC) of Canada, NRC, Canada and ISE Ltd. Thanks to Michael Greenspan, Juan-Manuel Ahuactzin and Eric Jackson for stimulating discussions, and to Gilbert Soucy for providing us with a robust and excellent scanner at a reasonable cost.

References

[1] J. Ahuactzin, K. Gupta, E. Mazer, *Manipulation Planning for Redundant Robots: A Practical Approach*, The Inter. J. of Robotics Research, Vol. 17, No. 7, July, pp. 731 - 741

[2] P. Bessiere, J. Ahuactzin, et al. *The "Ariadne's Clew" Algorithm: Global Planning with Local Methods*, Algorithmic Foundation of Robotics, A K Peters, Ltd, pp. 39-47, 1997.

[3] Clark, P. J. and Evans, F. C. ,*Distance to nearest neighbor as a measure of spatial relationship in populations*, Ecolgy 35: 445 - 453, 1954

[4] Choset, H. and Burdick, J.W.*Sensor Based Planning for a Planar Rod Robot*, Proc. IEEE conference of Robotics and Automation

[5] T. Cormen, C. Leiserson, R. Rivest *Introduction to Algorithms*, McGraw Hill, 1989

[6] K. Gupta, A. P. del Pobil, *Practical Notion Planning in Robotics: Current Approaches and Future Directions*, John Wiley and Sons, 1998.

[7] K. Kutulakos, V. Lumelsky et al *Vision-Guided Exploration: A step Toward General Motion Planning in Three Dimensions* , Proc. 1993 IEEE Inter. Conf. on Robotics and Automation.

[8] V. Hayward and R. Paul, *Robot Manipulator Control Under UNIX: RCCL, a Robot Control Library*, International Journal of Robotics Research, pp. 94 - 111 Vol 5, No. 4, 1986

[9] E. Kruse, R. Gutsche el al, *Effective, Iterative, Sensor Based 3-D Map Building Using Rating Functions in Configuration Space*, Proc. 1996 IEEE Inter. Conf. on Robotics and Automation, pp. 1067 - 1072

[10] J. C. Latomb, *Robot Motion Planning*, Kluwer Academic Publications, 1991

[11] Lumelsky, V.J.; Cheung, E, *Real-time Collision Avoidance in Teleoperated Whole-Sensitive Robot Arm Manipulators*, IEEE Transactions on Systems, Man and Cybernetics, Jan.-Feb. 1993, vol.23, (no.1):194-203.

[12] V. Lumelsky, K. Sun, *Three-dimensional Motional Planning in an Unknown Environment for Robot Arm Manipulators with Revolute or Sliding Joints*, International Journal of Robot and Automation, vol. 9, No. 4, pp. 188- 198.

[13] V. Lumelsky*A Comparative Study On the Path Length Performance of Maze-Searching And Robot Motion Planning Algorithms*, IEEE trans. on Robotics and Automation, vol. 7, No. 1, pp. 57- 66, 1991

[14] H. Noborio S. Fukuda and S. Arimoto, *Construction of the octree approximating three dimensional object by using multiple views*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol, 10, No. 6, Nov. 1988., pp. 769 - 781

[15] L. Kavraki, P. Svestka. J. Latomb, M. Overmars, *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*, IEEE transactions on Robotics and Automation. Aug, 1996, vol 12, pp. 556 - 580.

[16] H. Noborio, T. Yoshioka *Sensor-Based Navigation for a Mobile Robot Under Uncertain Conditions*, in Pratical Motion Planning in Robotics, Current Approaches and Future Directions. Ed. by K. Gupta and A. Del Pobil, 1998

[17] M. Overmars, Petr Svestka, *A Probablistic Learning Approach to Motion Planning*, Algorithmic Foundation of Robotics, A K Peters, Ltd, pp. 19-37. 1996

[18] A. Portilla, *Planificacion de trayectorias de un brazo robot en un ambiente desconocido*, Universidad de las Americas-Puebla, Mexico, 1999

[19] P. Renton, M. Greenspan et al, *Plan-N-Scan: A Robotic System for Collision Free Autonomous Exploration and Workspace Mapping*, Journal of Intelligent and Robotic System 24: 207 - 234, 1999

[20] Rimon, E. and Canny, J.F. *Construction of C-space Roadmaps using local Sensory Data – what should the sensors look for?*, Proceedings 1994 IEEE International Conference on Intelligent Robot and System, pp. 117-124.

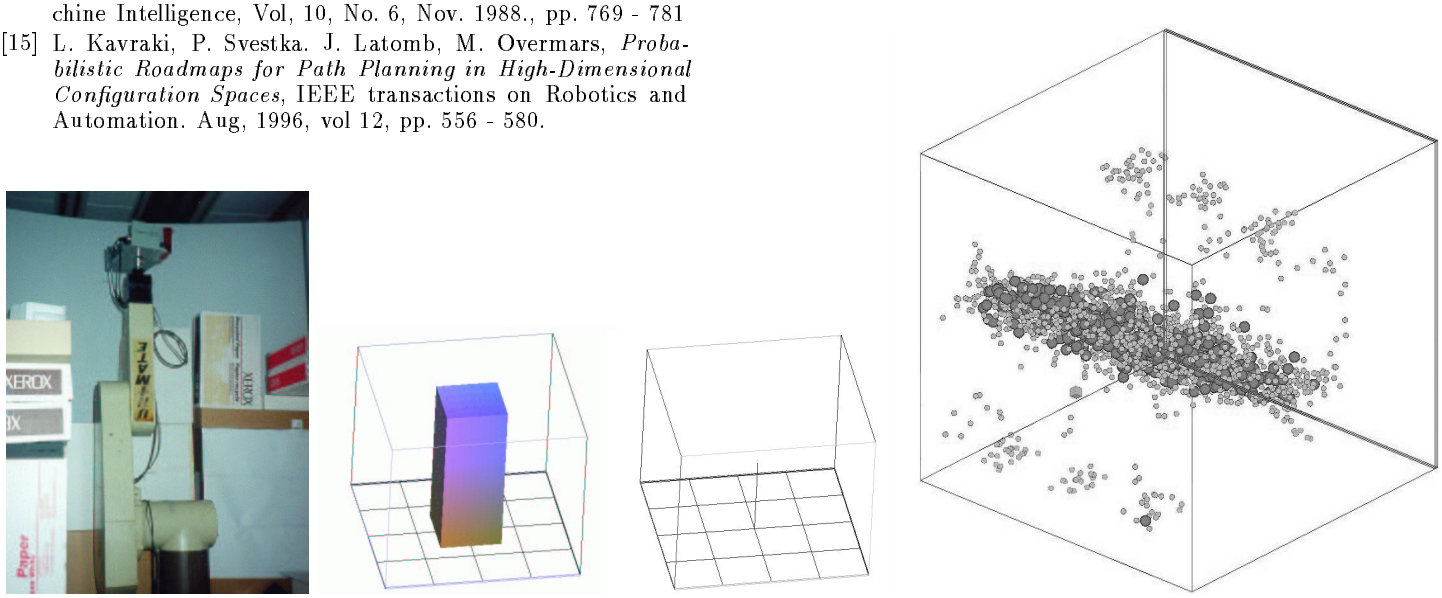
[21] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley Publishing Company, 1989.

[22] G. Soucy, F. Callari and F. Ferrie, *Uniform and Complete Surface Coverage with a Robot-Mounted Laser Range Finder* , Proceedings 1998 IEEE/RSJ International Conference on Intelligent Robot and System, pp. 1682 - 1688.

[23] Y. Yu, K. Gupta, *An efficient On-line Algorithm for Direct Octree Construction from Range Images*, Proc. 1998 IEEE Inter. Conf. on Robotics and Automation, pp. 3079 - 3084.

[24] Y. Yu and K. Gupta, *On Sensor-based Roadmap: A Framework for Motion Planning for a Manipulator Arm in Unknown Environments*, Proc. 1998 IEEE/RSJ Inter. Conf. on Intelligent Robot and System, pp. 1919 - 1924.

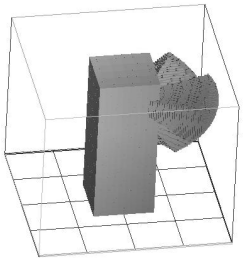
[25] K. Gupta, Y. Yu and J. Ahuactzin*Theoretical Analysis of Sensor Based Articulated Robot Motion Planning*, In preparation.



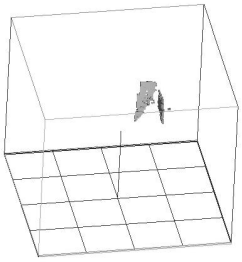
a1 b1 c1 d1

Figure 7: A sequence of snapshots of the robot (column a), $\mathcal{P}_{free}$ (column b) and $\mathcal{P}_{obstacle}$ (column c), sensed by the robot (b and c are the internal models built by the planner), and the roadmap (landmarks) in the C-space (column d). Please see text for further explanation.

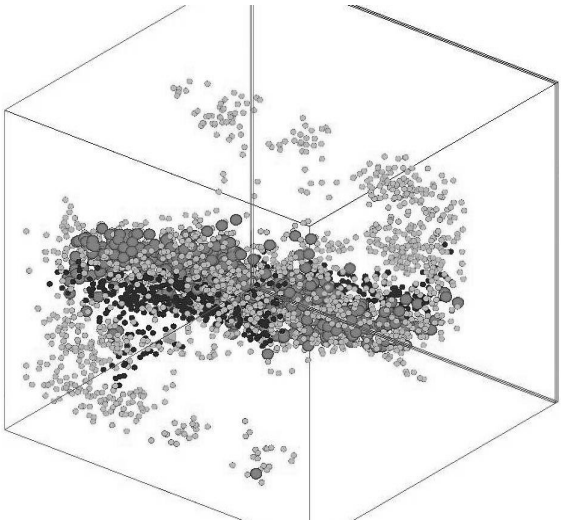
a2



b2

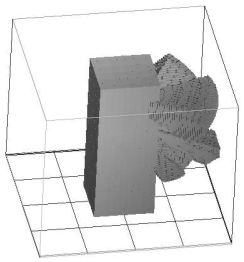


c2

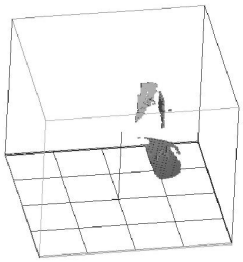


d2

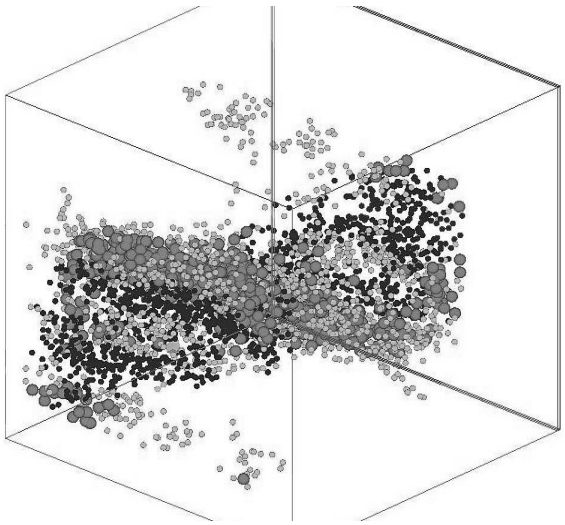
a3



b3

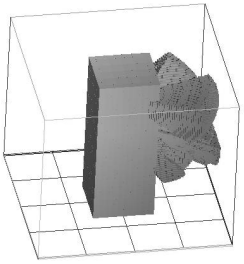


c3

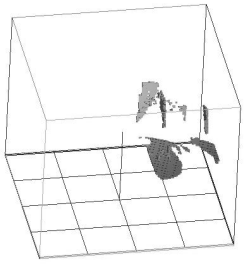


d3

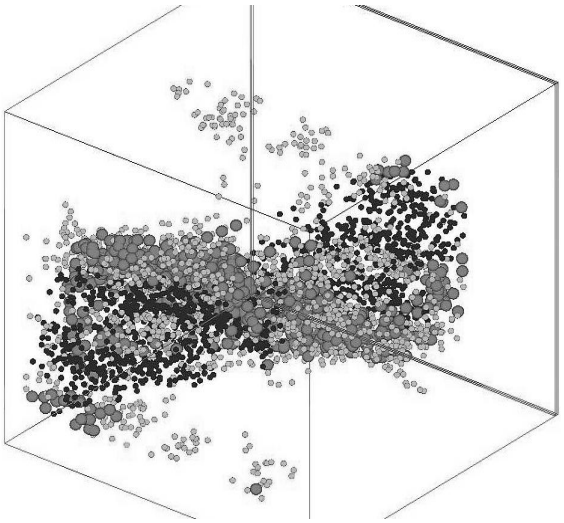
a4



b4



c4



d4

Figure 7: Continued from last page.