Sample copy of poster in Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pp. 1729-1732, 2005

Path-Planning for Autonomous Training on Robot Manipulators in Space

Froduald Kabanza
Université de Sherbrooke
kabanza@usherbrooke.ca

Roger Nkambou
U. du Québec à Montréal
nkambou.roger@uqam.ca

Khaled Belghith
Université de Sherbrooke
khaled.belghith@usherbrooke.ca

Abstract

This paper describes the integration of robot path-planning and spatial task modeling into a software system that teaches the operation of a robot manipulator deployed on International Space Station (ISS). The system addresses the complexity of the manipulator, the limited direct view of the ISS exterior, and the unpredictability of lighting conditions in the workspace. Robot path planning is used not for controlling the manipulator, but for automatically checking errors of a student learning to operate the manipulator and for automatically producing illustrations of good and bad motions in training.

1 Introduction

Designing software that teaches requires, in advanced cases, the implementation of "intelligence" capabilities. After all, best human teachers are those mastering the subject they teach, having communication skills and understanding the student's solving process in order to help him. With the aim of furthering intelligent software-based education systems, we have been developing a software simulator, called *RomanTutor*, that can be used to train astronauts to operate a robotics manipulation system (the Mobile Servicing System, MSS), on the International Space Station (ISS, Figure 1).

Figure 1. ISS with the SSRMS

The MSS consists of a Space Station Remote Manipulator System (SSRMS), a Mobile Base System (MBS), a Mobile Transporter (MT), and a Special Purpose Dexterous Manipulator (SPDM). The SSRMS is a 17-meter long articulated robot manipulator with seven rotational joints and two latching end-effectors which can be moved to various fixtures, giving it the capability to "walk" from one grappling fixture to next on the exterior of the ISS. The SPDM is a dexterous manipulator with two symmetrical six-joint arms and can be operated from the end of the SSRMS. The MT is a platform that serves to move SSRMS along the main truss of ISS.

The MSS is operated from a robot workstation located inside one of the ISS modules, and equipped with three video monitors, each displaying a view from one of the 14 cameras mounted on the ISS exterior and the SSRMS. Crewmembers operating the MSS have no direct view of the ISS exterior other than the three monitors. In fact, choosing the right camera views to display is part of the tasks for operating the SSRMS.

RomanTutor is a system still under development; here we describe the integration of robot path-planning and spatial task modeling into an MSS simulator to provide useful feedback to a student operating the SSRMS. To illustrate, when a student is learning to move a payload, RomanTutor invokes a path-planner periodically to check whether there is a path from the current configuration to the target, and provides feedback accordingly. The path-planner not only computes collision free paths but is also capable of taking into account the limited direct view of the ISS, the lighting conditions and other safety constraints about operating the SSRMS.

2 Architecture and Basic Functionalities

RomanTutor works with any robot manipulator provided a 3D model of the robot and its workspace are specified. The system includes the following components among others (Figure 2): a graphic user interface, a feedback generator, a path planner, a movie generator, and third-party libraries (PQP [Larsen *et al*., 2000], Open Inventor from Silicon Graphics and MPK [Sanchez and Latombe, 2001]).

A snapshot of the *user interface* is shown on Figure 3. It emulates the Robot workstation using three screens (for the three monitors). The keyboard is used to operate the robot. In *command mode*, one controls the joints directly; in *automatic mode*, one moves the end-effector, small increments at a time, relying on inverse kinematics to calculate the joint rotations.

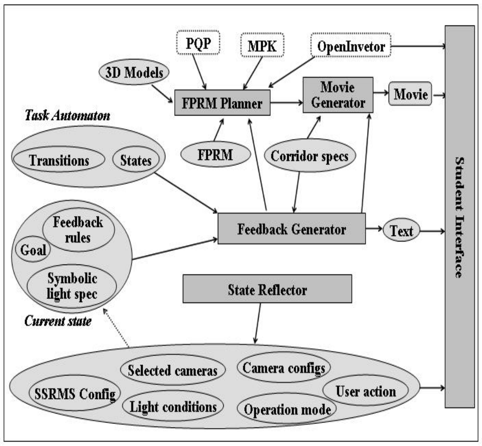

Figure 2. RomanTutor Architecture

The robot free workspace is segmented into zones with each zone having an associated degree of desirability that is, a real number in the interval [0 1], depending on the task, visual cue positions, camera positions, and lighting conditions. The closer the *dd* is to 1, the more the zone is desired. *Safe corridors* are zones with *dd* near to 1, whereas *unsafe corridors* are those with *dd* in the neighborhood of 0. We extend the definition of *dd* to robot configurations and paths in a straightforward way.

Students carry out robot operation tasks that involve moving the manipulator (avoiding collision and singularities, using the appropriate speed, switching cameras as appropriate, and using the right operation mode at different stages), berthing, or mating. These tasks require the student to be able to identify a corridor in a free workspace for a safe operation of the robot and follow it. The student must do this based on the task, the location of cameras and visual cues, and the current lighting conditions. Therefore localization and navigation are important in robot operations.

The *feedback generator* periodically checks the current state to trigger feedback to the student, using rules that are preconditioned on the current state information and the current goal. These are "teaching" expert rules and can be as efficient as the available teaching expertise allows. The feedback generator also changes the lighting conditions based upon specification rules in the current state. Feedback rules, lighting rules and goals are structured into a *task automaton*.

A *task automaton* is a state transition system that abstracts the evolution of the simulated system configuration under the student's actions. In any given state of the task automaton the feedback generator is monitoring the student's actions, focusing on one subgoal, and using a set of feedback rules to help the student. A transition to a new state (e.g., the payload has reached a predefined milestone) switches the subgoal and the feedback rules to those of the new state.

Feedback rules are production rules preconditioned on the state variables and the state goal; the consequents are normally multimedia content to be displayed to the student, but rules can also update user-defined variables (e.g., to keep a record of previous state variables or to update a performance score for the student).

Feedback rules can invoke automated movie generation using templates of the form "illustrate correct move from <current-configuration> to <goal-configuration>". Such a template uses safe corridor specifications and the path-planner to generate a path with a high desirability degree (i.e., in a safe corridor); then, it determines the best sequence of camera views for the different parts of the zone, and uses them as virtual cameras to generate a movie viewing the path.

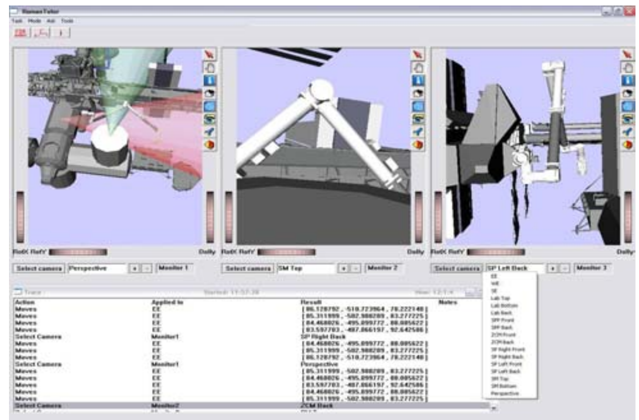

Figure 3. RomanTutor User Interface

3 The Path Planner

For efficient path planning, we pre-process the robot workspace into a roadmap of collision-free robot motions in regions with highest desirability degree. More precisely, the roadmap is a graph such that every node *n* in the graph is labeled with its corresponding robot configuration *n.q* and its degree of desirability *n.dd*, which is the average of *dd* of zones overlapping with *n.q*. An edge *(n,n')* connecting two nodes is also assigned a *dd* equal to the average of *dd* of configurations in the path-segment *(n.q,n'.q)*. The *dd* of a path (i.e., a sequence of nodes) is an average of *dd* of its edges.

Following probabilistic roadmap methods (PRM) [Sanchez and Latombe, 2001], we build the roadmap by picking robot configurations probabilistically, with a probability that is biased by the density of obstacles. A path is then a sequence of collision free edges in the roadmap, connecting the initial and goal configuration.

Following Anytime Dynamic A* (AD*) approach [Likhachev *et al*, 2005], to get new paths when the conditions defining safe zones have dynamically changed, we can re-plan fast by exploiting the previous roadmap. On the other hand, paths are computed through incremental improvements, so that the planner can be stopped at anytime and provide a collision-free path, and the more time it is given, the better the path optimizes moves through desirable zones. Therefore, our planner is a combination of the traditional PRM approach [Sanchez and Latombe, 2001] and AD* [Likhachev *et al*, 2005], and it is flexible in that it can into account zones with degrees of desirability. We call it Flexible Anytime Dynamic PRM (FADPRM).

More precisely, FADPRM works as follows. The input is: an initial configuration, a goal configuration, a 3D model of obstacles in the workspace, a 3D specification of zones with corresponding *dd*, and a 3D model of the robot. Given this input:

- To find a path connecting the input and goal configuration, we search backwardly from the goal towards the initial (current) robot configuration. Backward instead of forward search is done because the robot moves, hence its current configuration is not necessary the initial configuration; we want to re-compute a path to the same goal when the environment changes before it is reached.
- A probabilistic queue OPEN contains node of the frontier of the current roadmap (i.e., nodes be expanded because new or previously expanded but not update anymore w.r.t. to the desired path) and a list CLOSED contains non frontier nodes (i.e., nodes already expanded)
- Search consists in repeatedly picking a node from OPEN then generating its predecessors putting them new ones or not update ones in OPEN.
- The density of a node is the number of nodes in the roadmap with configurations that are a short distance away (proximity being an empirically set parameter, taking into account the obstacles in an application domain). The distance estimate to the goal takes into account the node's *dd* and the Euclidean distance to the goal as explained below.

  With these definitions, a node *n* in OPEN is selected for expansion from OPEN with probability proportional to

  *(1-β)*density(n) + β * goal-distance-etsimate(n),*

  with $0 \leq \beta \leq 1$.

  This equation implements a balance between fast-solution search and best-solution search by choosing different values for β. With β near to 0, the choice of a node to be expanded from OPEN depends only on the density around it. That is, nodes with lower density will be chosen first, which is the heuristic used in traditional PRM approaches to guaranty the diffusion of nodes and to accelerate the search for a path [Sanchez and Latombe, 2001]. As β approaches 1, the choice of a node to be expanded from OPEN will rather depend on its estimate distance to the goal. And here we are seeking optimality rather than speed.
- To increase the resolution of the roadmap, a new predecessor is randomly generated within a short neighborhood radius (the radius is fixed empirically based on the density of obstacles in the workspace) and added to the list of successors in the roadmap generated so far, the entire list predecessors is returned.
- Collision is delayed: detection of collisions on the edges between the current node and its predecessors is delayed until a candidate solution is found; if colliding, we backtrack. Already done collisions are stored in the roadmap to avoid redoing them again.
- The robot may start executing the first path found.
- Concurrently, the path continues being improved by re-planning with an increased value of β.
- Changes in the environment (moving obstacles or changes in *dd* for zones) cause update of the roadmap and re-planning.

4 Conclusion

RomanTutor's potential benefits to future training strategies are (1) the simulation of complex tasks at a low cost (e.g., using inexpensive simulation equipment and with no risk of injuries or equipment damage) and (2) the installation anywhere and anytime to provide “just in time” training. Crewmembers would be able to use it onboard of the ISS, for example, to study complex maintenance or repair operations. For very long missions, they would be able to use it to train regularly in order to maintain their skills.

References

[Sanchez and Latombe, 2001] G. Sanchez and J-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking, In *Proc. of 9th International Symposium on Robotics Research*, pages 403-417, 2001.

[Larsen *et al*., 2000] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast Distance Queries using Rectangular Swept Sphere Volumes. In *Proc. of IEEE International Conference on Robotics and Automation, 4:24-28,* 2000.

[Likhachev *et al*, 2005] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast Distance Queries using Rectangular Swept Sphere Volumes. In *Proc. of IEEE International Conference on Robotics and Automation, 4:24-28,* 2000.