Roman Tutor: A Robot Manipulation Tutoring Simulator

Khaled Beghith and **Froduald Kabanza**
Université de Sherbrooke
{khaled.belghith, froduald.kabanza}@usherbrooke.ca

Roger Nkambou and **Mahie Khan**
Université du Québec a Montréal
{nkambou.roger, khan.mahie}@uqam.ca

Leo Hartman
Canadian Space Agency
leo.hartman@space.gc.ca

Abstract

This demonstration will be about *Roman Tutor*, a system that we are developing to teach astronauts how to operate a robot manipulator deployed on the International Space Station (ISS). Operators do not have a direct view of the scene of operation on the ISS and must rely on cameras mounted on the manipulator and at strategic places of the environment where it operates. *Roman Tutor* uses a robot path-planner, not to control the manipulator, but to automatically check errors of a student learning to operate the manipulator, and to automatically produce illustrations of good and bad motions in training.

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. However, as we still do not understand well how to model human cognition, efforts in the design of intelligent software-based education systems remain experimental. In this line of inquiry, we have been developing an intelligent tutoring software, *Roman Tutor*, to support astronauts in learning how to operate the Space Station Remote Manipulator System (SSRMS), an articulated robot arm mounted on the international space station (ISS). Figure 1 illustrates a snapshot of the ISS with the SSRMS.

The SSRMS is operated from a robotic 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 SSRMS 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 manipulator.

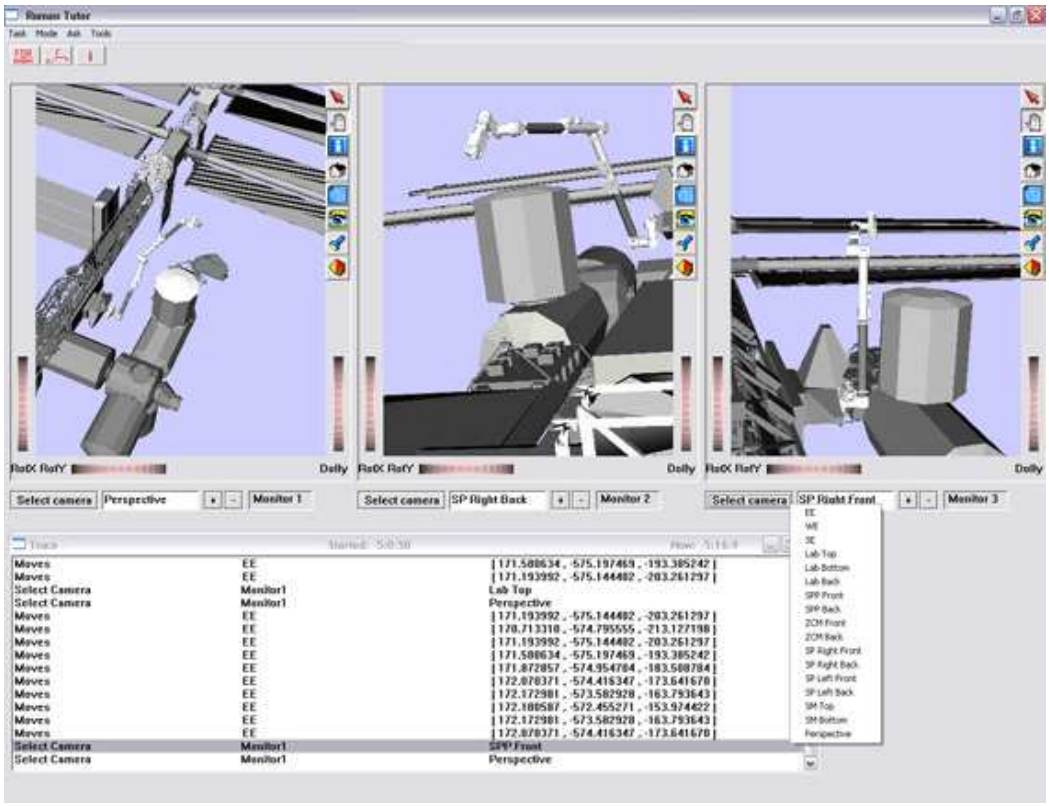
Even though they are supported by on-ground operators and have precise safety protocols to follow during their operations, training provides crewmembers with (1) manipulator teleoperation skills, (2) the ability to carry out operations independently (i.e., without ground support) and (3) detailed knowledge of SSRMS design that is required for the tasks scheduled for their mission and SSRMS operation. Consequently the associated training is challenging due to the limited direct view of the ISS exterior, unpredictable lighting conditions, the complexity of the SSRMS (seven degrees of freedom), high masses, costly payloads, and vulnerable robot mechanics requiring very slow speeds to avoid overruns and to reduce risks of oscillations, collisions or singularities.

Figure 1 ISS with the SSRMS

Roman Tutor is still under development; at ICAPS, we would like to demonstrate its current features that include a path-planner called FADPRM and used to provide tutoring feedback to the student. To illustrate, when an astronaut is learning to move a payload, *Roman Tutor* invokes the FADPRM path-planner periodically to check whether there is a path from the current configuration to the target, and provides feedback accordingly.

FADPRM path-planner not only computes collision free paths among obstacles , as is normal for a classic path-planner, but is also capable of taking into account the lim-

Figure 2 Roman Tutor Student Interface



ited direct view of the ISS, the lighting conditions and other safety constraints about operating the SSRMS. We developed such a robot-path planner by extending the probabilistic roadmap method (Sanchez & Latombe 2001) with constraints on safe and unsafe corridors of operation. In addition this planner proves useful in the automatic generation of movies that illustrate good and bad operations.

In the next section we describe *Roman Tutor*'s architecture, outline its basic functionalities and discuss details about its main components, mainly the FADPRM path-planner. We, then, conclude with a discussion on related work.

Architecture and Basic Functionalities

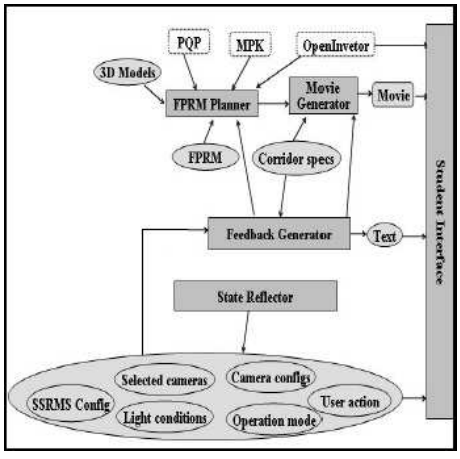
Main Components

One challenge in developing a good training simulator is of course to have the simulator in the first place. Then we need to integrate useful training strategies, in particular a model for tracing the student to assess his progress on a task and a process to provide really useful, intelligent feedback.

Roman Tutor 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 3): a graphic user interface, a feedback generator, a path planner, a movie generator, and third-party libraries: Proximity Query Package (PQP) (Larsen *et al.* 2000), Open Inventor from Silicon Graphics, and Motion Planning Kit (MPK) (Sanchez & Latombe 2001).

A snapshot of the user interface is shown on Figure 2. It emulates the ISS robotic workstation using three screens (for the three monitors). The keyboard is used to operate the robot (the SSRMS in our case). 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 3 Roman Tutor Architecture



In Figure 2, different cameras are selected, displaying the same robot configuration with different views. The perspective camera (on the left) can inspect the entire ISS 3D model. It is used in training tasks aimed at helping a student to develop a mental 3D model of the ISS, but it's not physically available on the ISS. Normal training uses replica models of the ISS for the same purpose.

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.

Students could carry out on *Roman Tutor* several kinds of training tasks that we describe more formally in the next section. The *state reflector* periodically updates the student's actions (i.e, keyboard inputs) and their effects on the ISS environment (robot configuration, cameras mapped to the monitors, their view angles, and the operation mode). It also monitors lighting conditions.

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.

Training Tasks

Training tasks can be classified as open, recognition, localization, or robot manipulation. Open tasks are those in which the learner interacts with the simulator, without any formally set goal, with minimal assistance configured in the system's preferences (e.g., collision warning and avoidance).

Recognition tasks train to recognize the different elements in the workspace. An example is to show a picture of an element of the ISS and ask the student to name it and describe its function.

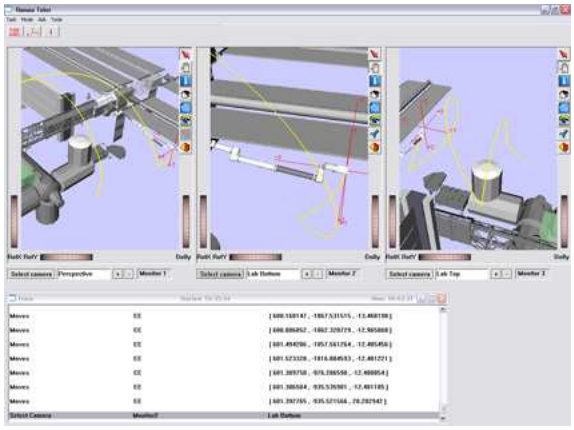
Localization tasks train to locate visual cue or ISS elements and to relate them spatially to each others. An example is to show a picture of a visual cue and ask the student to make it visible on the screen using an appropriate selection of cameras; or we can ask to name elements that are above another element shown on the screen.

Robot operation tasks deal with 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. An illustration is to move the arm from one position to another, with or without a payload. Another example is to inspect an indicated component of the ISS using a camera on the end-effector. These tasks require the student to be able to define 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.

Tasks are made more or less unpredictable by dynamically changing the lighting conditions, thus requiring the revalidation of safe corridors. Feedback rules can take into account how long the student has been trying on a subtask and how good or bad he is progressing on it.

As most complex tasks deal in one way or another with moving the SSRMS, for the simulator to be able to understand students' operations in order to provide feedback, it must be aware of the space constraints and be able to move the arm by itself. A path-planner inside *Roman Tutor* that calculates arm's moves without collision and consistent with best available cameras views is then the key training resource on which other resources and abstract tutoring processes hinge. Figure 4 illustrates a solution path computed by the FADPRM path-planner given to the student on the *Roman Tutor* user interface to help him carry on his task.

Figure 4 FADPRM Path-Planner



The FADPRM 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.

The calculation of a configuration *dd* and a path *dd* is a straightforward extension of collision checking for configurations and path segments. For this, we customized the Proximity Query Package (PQP) (Larsen *et al.* 2000). The 3D models for the ISS, the SSRMS and zones are implemented using a customization of Silicon Graphics' Open Inventor. The robot is modeled using Motion Planning Kit (MPK), that is, an implementation of Sanchez and Latombe's PRM planner (Sanchez & Latombe 2001).

Following probabilistic roadmap methods (PRM) (Sanchez & Latombe 2001), we build the roadmap by picking robot configurations probabilistically, with a prob-

ability 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 the Anytime Dynamic A* (AD*) approach (Likhachev *et al.* 2005), to get new paths when the conditions defining safe zones have dynamically changed, we can quickly re-plan by exploiting the previous roadmap. On the other hand, paths are computed through incremental improvements so the planner can be stopped at anytime to 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 & 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 backward 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 necessarily the initial configuration; we want to re-compute a path to the same goal when the environment changes before the goal is reached.
- A probabilistic queue *OPEN* contains nodes of the frontier of the current roadmap (i.e., nodes are expanded because they are new or because they have previously been expanded but are no longer up to date w.r.t. to the desired path) and a list *CLOSED* contains non frontier nodes (i.e., nodes already expanded).
- Search consists of repeatedly picking a node from *OPEN*,generating its predecessors and putting the new ones or out of date 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. A node n in *OPEN* is selected for expansion with probability proportional to :

  $$(1-\beta)/density(n)+\beta*goal-distance-estimate(n)$$

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

  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 & Latombe 2001). As β approaches 1, the choice of a node to be expanded from *OPEN* will rather depend on its estimated distance to the goal. In this case, we are seeking optimality rather than speed.
- To increase the resolution of the roadmap, a new predecessor is randomly generated within a small neighborhood radius (that is, 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 there is a collision, we backtrack. Collisions that have already been detected are stored in the roadmap to avoid doing them again.
- The robot may start executing the first path found.
- Concurrently, the path continues being improved by replanning with an increased value of β.
- Changes in the environment (moving obstacles or changes in *dd* for zones) cause updates of the roadmap and replanning.

Conclusion

The current prototype of *Roman Tutor* includes a new path-planner called FADPRM to handle workspaces with desirable and undesirable zones, and basic tutoring feedbacks. When completed, potential benefits to future organizational 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 will be able to use it onboard of the ISS to comprehend a workaround for a repair. For very long missions, they will use it to train regularly in order to maintain their skills. We also believe that ideas behind *Roman Tutor* could inspire training systems for other applications of robot manipulators.

Acknowledgment

We are grateful to the Canadian Space Agency, The Infotel Group and the Natural Sciences and Engineering Research Council of Canada (NSERC) for their logistic and financial support. We also thank the Robotics Laboratory at the University of Stanford for making MPK available to us, and Mitul Saha for help on MPK.

References

Larsen, E.; Gottschalk, S.; Lin, M.; and Manocha, D. 2000. Fast distance queries using rectangular swept sphere volumes. In *IEEE International Conference on Robotics and Automation*, 4:24–28.

Likhachev, M.; Ferguson, D.; Gordon, G.; Stentz, A.; and Thrun, S. 2005. Anytime dynamic a*: An anytime, replanning algorithm. In *International Conference on Automated Planning and Scheduling*.

Sanchez, G., and Latombe, J.-C. 2001. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Ninth International Symposium on Robotics Research*.