# An Intelligent Simulator for Telerobotics Training

Khaled Belghith, Roger Nkambou, Froduald Kabanza, and Leo Hartman

**Abstract**—Roman Tutor is a tutoring system that uses sophisticated domain knowledge to monitor the progress of students and advise them while they are learning how to operate a space telerobotic system. It is intended to help train operators of the Space Station Remote Manipulator System (SSRMS) including astronauts, operators involved in ground-based control of SSRMS and technical support staff. Currently, there is only a single training facility for SSRMS operations and it is heavily scheduled. The training staff time is in heavy demand for teaching students, planning training tasks, developing teaching material, and new teaching tools. For example, all SSRMS simulation exercises are developed by hand and this process requires a lot of staff time. Once in an orbit ISS astronauts currently have only simple web-based material for skill development and maintenance. For long duration space flights, astronauts will require sophisticated simulation tools to maintain skills. Roman Tutor addresses these challenges by providing a portable training tool that can be installed anywhere and anytime to provide "just in time" training. It incorporates a model of the system operations curriculum, a kinematic simulation of the robotics equipment, and the ISS, a high performance path planner and an automatic task demonstration generator. For each element of the curriculum that the student is supposed to master, Roman Tutor generates example tasks for the student to accomplish within the simulation environment and then monitors its progression to provide relevant feedback when needed. Although motivated by the SSRMS application, Roman Tutor remains applicable to any telerobotics system application.

**Index Terms**—Telerobotics training, intelligent tutoring, robot manipulation, path planning, demonstration generation.

## 1 Introduction

Roman TUTOR (RObot MANipulation Tutor) is a simulation-based tutoring system to support astronauts in learning how to operate the Space Station Remote Manipulator (SSRMS), an articulated robot arm mounted on the international space station (ISS). Once in orbit, ISS astronauts currently have only simple web-based material for skill development and maintenance. For long duration space flights, astronauts will require sophisticated simulation tools to maintain skills. Roman Tutor addresses these challenges by providing a portable training tool that can be installed anywhere and anytime to provide "just in time" training. Fig. 1 includes a image of the SSRMS on the ISS. Astronauts operate the SSRMS from a robotic workstation located inside one of the ISS compartments. Fig. 1 also shows the workstation which has an interface with three monitors, each of which can be connected to any of the 14 cameras placed at strategic locations on the exterior of the ISS. Roman Tutor's user interface in Fig. 2 includes the most important features of the robotic workstation.

The SSRMS is a key component of the ISS and is used in the assembly, maintenance, and repair of the station, and also for moving payloads from visiting shuttles. Operators manipulating the SSRMS on orbit receive support from ground operations. Part of this support consists of visualizing and validating maneuvers before they are actually carried out on the ISS. Operators have in principle rehearsed the maneuvers many times on the ground prior to the mission, but unexpected changes are frequent during the mission. In such cases, ground operators may have to generate 3D animations for the new maneuvers and upload them to the operator on the station. So far, the generation of these 3D animations are done manually by computer graphic programmers and thus are very time consuming.

SSRMS can be involved in various tasks on the ISS, including moving a load from one place of the station to another, inspecting the ISS structure (using a camera on the arm's end effector) and making repairs. These tasks must be carried out very carefully to avoid collisions with the ISS structure and to maintain safety-operating constraints on the SSRMS (such as avoiding self-collisions and singularities). At different phases of a given manipulation, the astronaut must choose a setting of cameras that provides him with the best visibility while maintaining awareness of his progress on the task. Thus, astronauts are trained not only to operate the arm itself, but also to recognize visual cues on the station that are crucial in mentally reconstructing the actual working environment from the partial and restricted views provided by the three monitors, and to select cameras depending on the task and other parameters.

One challenge in developing a good training simulator is, of course, to build it so that one can reason about it. This is even more important when the simulator is built for training purposes [1]. Until now, simulation-based tutoring is

- *K. Belghith and F. Kabanza are with the Department of Computer Science, University of Sherbrooke, Sherbrooke, QC J1K 2R1, Canada. E-mail: {khaled.belghith, kabanza}@usherbrooke.ca.*
- *R. Nkambou is with the Department of Computer Science, University of Quebec at Montreal, 201, av. PrŽsident-Kennedy, Montreal, QC H2X 3Y7, Canada. E-mail: nkambou.roger@uqam.ca.*
- *L. Hartman is with the Canadian Space Agency, 6767 Airport Rd., St-Hubert, QC J3Y 8Y9, Canada. E-mail: leo.hartman@asc-csa.gc.ca.*

*Manuscript received 30 May 2010; revised 16 Nov. 2010; accepted 4 Feb. 2011; published online 30 Mar. 2011.*
*For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2010-05-0083.*
*Digital Object Identifier no. 10.1109/TLT.2011.19.*

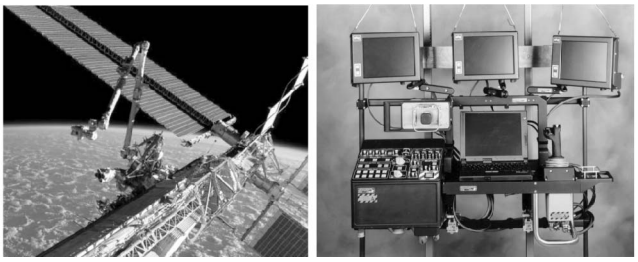


Fig. 1. SSRMS on the ISS (left) and the Robotic Workstation (right).

possible only if there is an explicit model or representation of the problem space associated with training tasks. The explicit representation is required in order to track student actions, to identify if these actions are still on a path to a solution and to generate relevant tutoring feedback [2], [3]. Knowledge and model tracing are only possible in these conditions [4]. It is not always possible to develop an explicit comprehensive task structure in complex domains, especially in domains where spatial knowledge is used, as there are many possible ways to solve a given problem. The robot manipulation that Roman Tutor focuses on is an example of such a domain. For each robot manipulation task, there is a combinatorial explosion of possible solutions for moving SSRMS from one place to another in the ISS environment. Such domains has been identified as "ill-structured" [5], [6].

Conventional tutoring approaches such as model-tracing [7] or constraint-based modeling [8] are very limited when applied on "ill-structured" domains. A model-tracing approach consists of comparing a predefined task model with a student's solution. In the context of robot manipulations, because of the infinity of solutions we have associated with each task, designing a task model by hand becomes practically infeasible. Applying a constraint-based modeling approach in the context of robot manipulations will also face the same kind of limitations. Here, identifying the constraints associated with robot manipulation tasks can be difficult and very time consuming. Since a huge number of constraints is required to achieve an adequate level of tutoring assistance [6], the approach becomes impractical.

To overcome these limitations, we propose a solution to this issue by integrating a sophisticated path planner FADPRM [9] as a domain expert system to support spatial reasoning within the simulator and make model tracing tutoring possible without any explicit task structure.

Flexible Anytime Dynamic PRM path planner (FADPRM) is an extension to the PRM planning framework [10] to handle regions to which we assign preferences within complex workspaces. By being flexible in this way, FADPRM not only computes collision free paths but also capable of taking into account the placement of cameras on the ISS, the lighting conditions and other safety constraints on operating the SSRMS. This allows the generation of collision-free trajectories in which the robot stays within regions visible through cameras and in which the manipulation is, therefore, safer and easier. FADPRM also implements a dynamic strategy to adapt efficiently to dynamic changes in the environment and replan on the fly by exploiting results from previous planning phases.

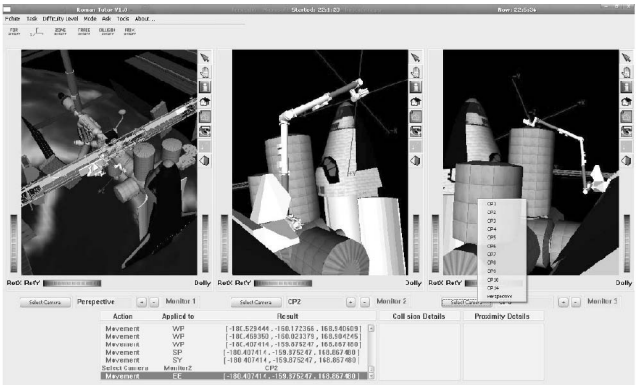


Fig. 2. Roman tutor interface.

FADPRM also implements an anytime strategy to provide a correct but likely suboptimal solution very quickly and then incrementally improve the quality of this solution if more planning time is allowed.

Roman Tutor uses the different capabilities implemented within the FADPRM path planner to provide useful feedback to a student operating the SSRMS simulation. To illustrate, when a student is learning to move a payload with the robot, 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. By using FADPRM as a robot manipulation domain expert, we follow an "expert system approach" to support the tutoring process within Roman Tutor. This approach has proven successful and has been used within different well-known intelligent tutoring systems such as SOPHIE I [11] and GUIDON [12]. But in our case, we are applying it in the context of robot manipulations, an "ill-structured" domain.

We also developed within Roman Tutor an automatic task demonstration generator (ATDG) [13], which generates 3D animations that demonstrate how to perform a given task with the SSRMS. The ATDG is integrated with the FADPRM path planner and can contribute to ground support of SSRMS operations by generating useful task demonstrations on the fly that help the student carry out his tasks. ATDG includes a component based on TLPlan [14] for camera planning and uses Linear Temporal Logic (LTL) as the language for specifying cinematographic principles and filming preferences. A robot trajectory is first generated by FADPRM and TLPlan is then called to find the best sequence of camera shots following the robot on its path.

In the next section, we start by presenting FADPRM and ATDG in detail. We then describe Roman Tutor's internal architecture and outline its basic functionalities. After enumerating the tasks on which a student is trained within Roman Tutor, we describe the approaches followed to provide the tutoring assistance. In a following section, we show how the use of FADPRM as a domain expert within the simulator helped in providing very relevant tutoring feedback to the student. We finally conclude with a discussion on related work.

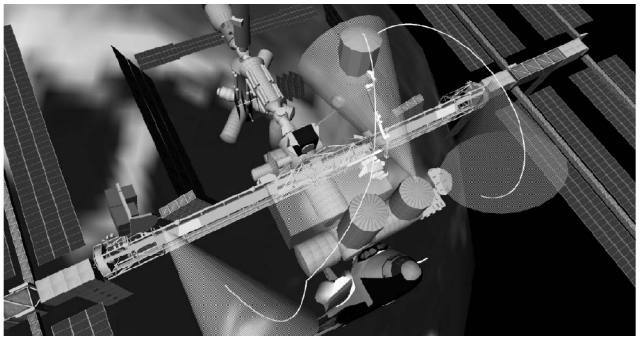


Fig. 3. SSRMS going through three different cameras fields of view (cones) and avoiding a nondesired zone (rectangular box).

## 2 FADPRM Path Planner

In its traditional form, the path planning problem is to plan a path for a moving body (typically a robot) from a given start configuration to a given goal configuration in a workspace containing a set of obstacles. The basic constraint on solution paths is to avoid collision with obstacles, which we call hereby a hard constraint. There exist numerous approaches for path planning under this constraint [10], [15], [16], [17], [18]. In order to take into account the visibility constraints we have in the SSRMS environment, we developed a new class of flexible path planners FADPRM [9] able to express and take into account preferences in the navigation of the robot within very complex environments. In addition to the obstacles the robot must avoid, our approach takes account of desired and undesired (or dangerous) zones. This will make it possible to take into account the placement of cameras on the station. Thus, our planner will try to keep the robot in zones offering the best possible visibility of progress on the task while trying to avoid zones with reduced visibility.

The robot free workspace is segmented into zones with each zone having an associated degree of desirability (dd), 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. A zone covering the field of view of a camera will be assigned a high dd and will have a cone shape; whereas, a zone with very limited lighting conditions will be considered as an nondesired zone with a dd near 0 and will take an arbitrary polygonal shape. Fig. 3 illustrates a trajectory of the SSRMS going through three cameras fields of view (three cones) and avoiding a nondesired zone (rectangular box).

For efficient path planning, we preprocess 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) [19], 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 the Anytime Dynamic A* (AD*) approach [20], to get new paths when the conditions defining safe zones have dynamically changed, we can quickly replan 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 (i.e., anytime after the first such path has been found) and the more time it is given, the more the path is optimized to move through desirable zones.

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 initial and goal configurations, we search backward from the goal toward the initial (current) robot configuration. Backward search instead of forward search is done because the robot moves and, hence, its current configuration is not in general the initial configuration; we want to recompute 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 nonfrontier 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 $\beta$. With $\beta$ 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 guarantee the

diffusion of nodes and to accelerate the search for a path [19]. As $\beta$ 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 the speed of finding a solution.

- 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 of 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 $\beta$.
- Changes in the environment (moving obstacles or changes in dd for zones) cause updates of the roadmap and replanning.

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) [21]. 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 [19].

## 3 The Automatic Task Demonstration Generator

The automatic task demonstration generator [13] takes as input a start and a goal configuration of the SSRMS. ATDG will generate a movie[1] demonstration of the required manipulations that bring the SSRMS from the start configuration to the goal configuration. The top figure in Fig. 4 shows the internal architecture of the ATDG. The bottom one shows the different steps the data go through in order to transform the two given configurations into a complete movie demonstration.

First, ATDG calls the FADPRM path planner to generate a collision free path between the two given configurations. The path is then passed to the trajectory parser which separates it into categorized segments. This will turn the continuous trajectory into a succession of scenes, where each scene can be filmed by a specific group of idioms. An idiom is a succession of shots that represents a stereotypical way to film a scene category. The parser looks for uniformity in the movements of the SSRMS to detect and recognize the category of scenes. Once the path is parsed, a

1. This paper has three supplemental movie files, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TLT.2011.19.

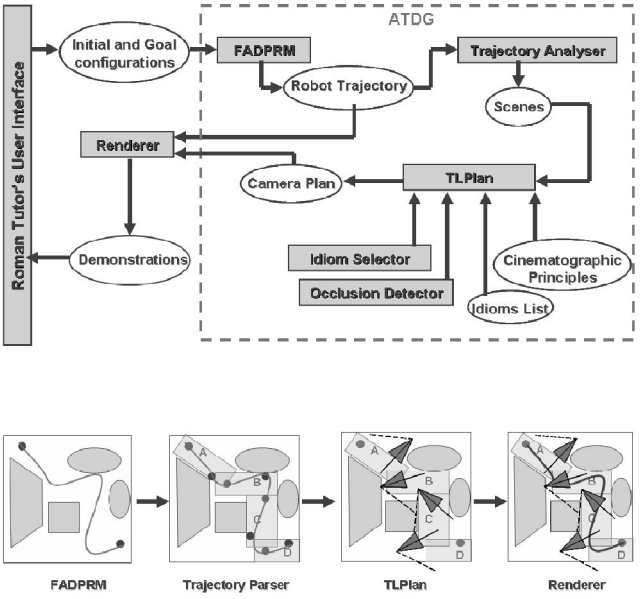



Fig. 4. ATDG architecture.

call is made to the camera planner TLPlan to find the best shots that best convey each scene, while making sure the whole is pleasing and comprehensive.

The use of TLPlan as a camera planner within ATDG provides two advantages. First, Linear Temporal logic, the language used by TLPlan is more expressive, yet with a simpler defined semantics, than previous camera planning languages such as DCCL [22]. For instance, we can express arbitrary temporal conditions about the order in which objects should be filmed, which objects should remain in the background until some condition become true, and more complex constraints that the LTL language can express. Second, TLPlan is more powerful than other camera planners presented in the literature such as [22], [23], [24], [25] because with TLPlan, LTL shot composition rules provide a search pruning capability. In ATDG, each shot in the idiom is distinguished by three key attributes: shot type, camera placement mode, camera zooming mode.

- Shot Types: five shot types are currently defined in the ATDG System: Static, GoBy, Pan, Track, and Pov. A Static shot is done from a static camera when the robot is in a constant position or moving slowly. A GoBy shot has the camera in a static position showing the robot in movement. For a Pan shot, the camera is in a static position but doing incremental rotations following the movement of the robot. A Track shot has the camera following the robot and keeping a constant position relative to it. Finally, the POV shot has the camera placed directly on the SSRMS, moving with the robot.
- Camera Placements: for each shot type, the camera can be placed in five different ways according to some given line of interest: External, Parallel, Internal, Apex, and External II. Currently, we take the trajectory of the robot's center of gravity as the line of interest which allows filming of a number of many typical maneuvers. For larger coverage

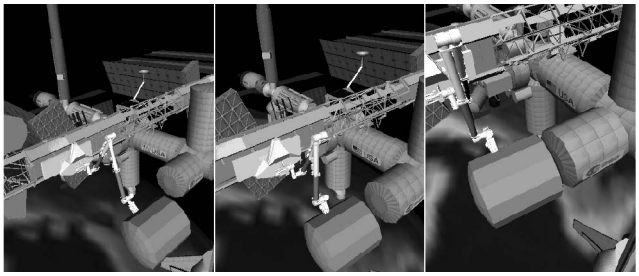


Fig. 5. Idiom to film the SSRMS anchoring a new component on the ISS.

of maneuvers, additional lines of interest will be added later.

- Zoom modes: for each shot type and camera placement, the zoom of the camera can be in five different modes: Extreme Close up, Close up, Medium View, Full View, and Long View.

Fig. 5 shows an idiom illustrating the anchoring of a new component on the ISS. It starts with a Track shot following the robot while moving on the truss. Then, another Track shot follows that shows the rotation of one joint on the robot to align with the ISS structure. And finally there is a Static shot focusing on the anchoring operation. In TLPlan, idioms are specified in the Planning Definition Language (PDDL 3.0). Intuitively, a PDDL operator specifies preferences about shot types in time and in space depending on the robot maneuver. Parsing the trajectory of the robot with the successive scenes performed, TLPlan will try to find a succession of shots that captures the best possible idioms. TLPlan also takes into account cinematic principles to ensure consistency of the resulting movie. Idioms and cinematic principles are in fact encoded in the form of temporal logic formulas within the planner. TLPlan uses also an occlusion detector to make sure the SSRMS is visible all the time. Once TLPlan is done, we are left with a list of shots that is passed to the rendering system to create the animation. The renderer uses both the shots given by TLPlan and the SSRMS trajectory in order to position the cameras in relation with the SSRMS, generating the final task demonstration.

For each SSRMS movement type (or scene), we have several idioms (from 6 to 10 in the current implementation) and each idiom is defined by taking into account the complexity of the movement, the geometry of the ISS, the visual cues on the ISS, and the preferences of the viewer. For example, if the SSRMS and the mobile base are moving along the main truss of the ISS, it is important that the camera shows not only the entire arm but also some visual cues on the ISS so the operator can get a sense of situational awareness for the relocation of the base of the arm. Consequently, the idioms for this manipulation will involve shots with a Full or Long View zoom. In contrast, movements involving the end effector require a high precision, so an Extreme Close Up zoom will be involved. Some of these parameters can also be set directly by the user's preferences. The user can choose, for example, to always prefer a precise set of cameras to use for the filming. The user can also choose some parts of the SSRMS the film should focus on the more possible.

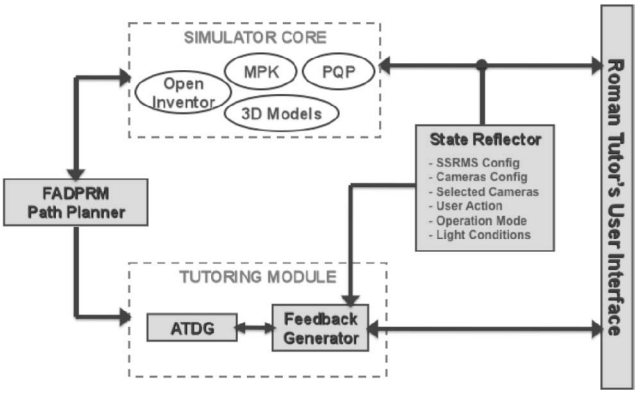



Fig. 6. Roman Tutor architecture.

## 4 Roman Tutor Architecture and Basic Functionalities

### 4.1 Architecture and Main Components

Roman Tutor works with any robot manipulator provided a 3D model of the robot and its workspace are specified. Roman Tutor's architecture includes the following components (Fig. 6): the graphic user interface, the State Reflector, the FADPRM path planner, the automatic task demonstration generator, the Tutoring Module and the Simulator Core with several third-party libraries: Proximity Query Package [21], Open Inventor from Silicon Graphics, and Motion Planning Kit [19].

As shown in Fig. 2, Roman Tutor's user interface has 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. In Fig. 2, different cameras are selected, displaying the same robot configuration from different viewpoints. 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 even though there is no such camera on the ISS. Normal training uses small physical models of the ISS for the same purpose.

In Roman Tutor students could carry out 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.

### 4.2 Training Tasks

Training tasks can be classified as open, recognition, localization, or robot manipulation. Open tasks are those in which the student 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 cues or ISS elements and to relate them

spatially to other elements. 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 manipulation 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. Robot manipulation tasks are made more or less unpredictable by dynamically changing the lighting conditions, thus requiring the revalidation of safe corridors.

### 4.3 Tutoring Approaches in Roman Tutor

The Feedback Generator inside the Tutoring Module (Fig. 6) 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. For the case of open, recognition and localization tasks, these rules are "pure domain-dependent pedagogical rules" related to task models designed by hand. For robot manipulation tasks with a goal, they are generic pedagogical rules. Feedback rules take into account how long the student has been trying on a subtask and how good or bad he is progressing on it.

In the context of open, recognition, and localization tasks, providing tutoring assistance seems straight forward. The domain knowledge is well defined: what element or cue of the ISS to recognize or to localize? what camera to choose and when?, etc. Here, we follow a model-tracing approach and define for each category of tasks a well structured task model to support the tutoring process. Task models are designed by hand starting from recommendations provided by human experts and are structured in the form of a graph encoding if-then rules. The Feedback Generator uses the predefined task graphs to validate student actions, identify gaps and provide feedback accordingly.

As we stated previously in an early section, the domain of robot manipulations is an "ill-structured" domain where classical tutoring approaches start to loose efficiency and show limitations. To overcome these limitations, we choose to follow an "expert system approach" and use the FADPRM path planner as a domain expert in our system to support the tutoring process. In the context of robot manipulation tasks, the Feedback Generator evaluates student actions by comparing it to the optimal solutions found by FADPRM and provides useful feedback accordingly. The tutoring process that uses FADPRM as an expert of the domain knowledge is described in more details in the next section.

One of the very important early results in intelligent tutoring research is the importance of the cognitive fidelity of the domain knowledge module. That is, it is important for the tutor to reason about the problem in the same way that humans do [26]. Approaches for modeling a domain expert within intelligent tutoring systems can be grouped into three main categories: black box models, glass box models, and cognitive models [27]. The main difference between these models lies in the cognitive fidelity with which each model represents the expert domain knowledge.

A black box model describes problem states differently than the student. The classic example of such a system is SOPHIE I [11]. SOPHIE I is a tutor for electronic troubleshooting that used its expert system to evaluate the measurements students were making in troubleshooting a circuit. The expert system made its decisions only by solving sets of equations. A glass box model is an intermediate model that reasons in terms of the same domain constructs as the human expert. However, the model reasons with a different control structure than the human expert. A classic example of such a system is GUIDON [12], a tutoring system for medical diagnosis. This system was built around MYCIN, an expert system for the treatment of bacterial infections. A cognitive approach, on the other hand, aims to develop a cognitive model of the domain knowledge that captures the way knowledge is represented in the human mind in order to make the tutor respond to problem-solving situations in a way very similar to humans. This approach, in contrast to the other approaches, has as an objective to support cognitively plausible reasoning [27]. A good example for such a tutoring system is SHERLOCK [28], another practice environment for electronics troubleshooting. SHERLOCK used a procedural domain knowledge representation based on a cognitive analysis of human skill acquisition.

At different phases of a given manipulation such as moving a payload using the SSRMS (Fig. 5), the astronaut must choose the best setting of cameras that provides him with the best visibility while keeping a good appreciation of his evolution in the task. Thus, astronauts are trained not only to manipulate the arm per se, but also to recognize the best cameras suitable to film a given configuration of the SSRMS within the ISS environment. Here, astronauts need to mentally reconstruct the actual working environment from just three monitors each giving a partial and restricted view.

The FADPRM planner tries to keep the SSRMS in zones offering the best possible visibility of the progress on the task while trying to avoid zones with reduced visibility. By taking into account the placement of cameras on the ISS, FADPRM reasons about actions in a way very similar to students: for each portion of the path, FADPRM tries the enter the field of view of the best suitable camera available. Thus, the use of FADPRM as a domain expert in Roman Tutor results in a tutoring approach that lies in between a glass box approach and a cognitive approach. Even if we are applying it in the context of an "ill-structured" domain, we believe that this will guarantee good quality of the tutoring provided to the student, at least at the same level as the one provided by a glass box model like GUIDON. In the

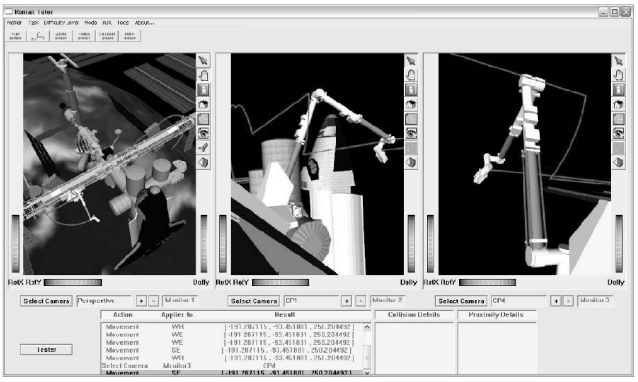


Fig. 7. Roman Tutor showing a robot trajectory to the student.

next section, we describe and evaluate the tutoring provided using FADPRM as an expert of the domain to a student working on robot manipulation tasks.

## 5 FADPRM AS A DOMAIN EXPERT IN ROMAN TUTOR

Roman Tutor initiates a robot manipulation task and monitors the student's progress toward accomplishing it. Students begin the task and can ask Roman Tutor for help or for a recommendation about what to do next. Students can ask Roman Tutor about how to avoid a collision with a nearby obstacle, how to go to a desired location in the workspace or how to go through a desired zone. In this situation, the Feedback Generator calls the ATDG (which calls the FADPRM planner) to compute and show a movie illustrating how to complete the manipulation task. If the objective is to give the operator a sense of the task as he will be seeing it from the command and control workstation, then virtual camera positions will be selected from the 14 cameras on the exterior of the ISS. But if the objective is to convey some cognitive awareness of the task, then virtual cameras are selected around the robot while moving on its trajectory to best help the operator gain a maximal cognitive awareness. The objective is set manually by the learner through Roman Tutor's interface to one of the following values "Use Cameras on ISS" or "Use Virtual Cameras."

Using the real time dynamic capability of the FADPRM path planner, the Feedback Generator monitors the student's activity in the State Reflector to validate incrementally student's action or sequence of actions, give information about the next relevant action or sequence of actions. The Feedback Generator regularly evaluates whether the task can be completed from the current configuration of the manipulator and whether it can be completed efficiently. At the point at which it discovers that the student would have to backtrack from the current position or that achieving the task takes more than the time planned for it, the Feedback Generator will intervene and begin to show the student a more efficient trajectory. Once a better initial trajectory has been demonstrated, the student can take control and resume the task. This error-prompted turn taking repeats until the task is completed (Fig. 7). We see here the importance of having FADPRM as a planner in our system to guide the operations by the student. By taking into account the placement of the cameras on the station, we are assured that the plan shown to the student passes through zones that are visible from cameras placed in the ISS environment and can then be followed by the student.

To evaluate the tutoring mechanics we implemented to support a student working on robot manipulation tasks, we compare the types of feedback we provide in our application to those provided by a classic intelligent tutoring system SHERLOCK [28] that is known to be efficient. SHERLOCK is a practice environment for electronics troubleshooting and provides advice on problem solving steps upon student request. Four types of feedback are available [26]:

1. advice on what test action to carry out and how,
2. advice on how to read the outcome of the test,
3. advice on what conclusion can be drawn from the test, and
4. advice on what option to pursue next.

As described earlier, our "FADPRM as a domain expert" tutoring approach provides feedback not only upon request but also intervenes automatically when it detects errors or difficulties experienced by the student. Different types of feedback are also available:

1. advice on what current action (or manipulation) to execute and how by showing a valid path to the current goal or by showing a movie computed with ATDG;
2. advice on how to avoid errors while progressing on a task by showing paths that avoid a nearby obstacle or by showing movies recorded from the most useful cameras.
3. advice on what conclusion can be drawn from the errors made by detecting incorrect choices made by the student and by proposing the right path to follow, and
4. advice on what future action or sequence of actions to pursue next in order to reach the goal.

For the feedback of types 2 and 3, the current trace of actions (robot manipulations and camera selections) made by the learner in order to reach the current configuration is saved. A call is then made to FADPRM and ATDG to evaluate the current trace, to diagnosis and identify errors and propose improvements: better manipulations to do, better cameras to select, etc. The list of these improvements is then displayed to the learner with a video illustrating them. The call to this diagnosis loop is made if requested by the learner after accessing the "Ask Menu" within the simulator interface or every time the system detects a nearing collision or a dangerous manipulation with the SSRMS too close to an obstacle on the ISS. In the current implementation of Roman Tutor, immediate feedback is provided to the learner every time he attempts to execute a dangerous path. Also, the Feedback provided always consists in showing the correct solution to the learner based on the diagnosis made. Hence, the tutoring behavior inside Roman Tutor remains limited. This issue will be addressed in future versions of Roman Tutor in order to investigate the use of different levels of intervention. Depending on the user's skills, his preferences and on the task being executed, an appropriate level of intervention should be applied.

In summary, the types of feedback provided by Roman Tutor are quite similar to those provided by SHERLOCK. The main difference is in the level of detail of the feedback provided. Since we are working in an ill-defined domain, the feedback provided by FADPRM remains less expressive and not as precise as the feedback provided by SHERLOCK. This issue can be addressed if the problem space generated by FADPRM can be manually edited to add, where needed, more information that can be used to enhance the quality and the precision of the Tutoring. Conversely, one of the main advantages of Roman Tutor is that, it operates in a domain where a cognitive approach like the one used within SHERLOCK cannot work due to the ill defiess of the domain. In this perspective, by using FADPRM as expert of the domain, we succeeded in achieving a good level of quality for the tutoring.

## 6 Conclusion

In this paper, we presented a real-time flexible approach for robot path planning called FADPRM and showed how it can be used efficiently to provide very helpful feedback to a student on a robot manipulation training simulator. FADPRM supports spatial reasoning and makes model tracing tutoring possible without any explicit task structure. By using FADPRM as a domain expert within the simulator, we showed how to achieve a high-quality level for the tutoring assistance without planning in advance what feedback to give to the student and without creating a complex task graph to support the tutoring process.

We also detailed the architecture of the intelligent training simulator Roman Tutor in which FADPRM is integrated. Among other components, Roman Tutor contains an automatic task demonstration generator used for the on the fly generation of useful task demonstrations that help the student carry on his manipulation tasks on the simulator.

Roman Tutor's 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. Crew members would be able to use it onboard 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. In particular Roman Tutor is able to generate as many training examples as the student wants. This capacity provides important learning challenges and opportunities that are not possible with the current system based on a fixed set of manually generated examples. Although motivated by the SSRMS application, Roman Tutor with its innovative components (FADPRM and ATDG) remains applicable to any other telerobotics system application.

Although Roman Tutor brings some interesting solutions for training in highly complex environments, a number of enhancements and extensions are still possible. First of all, its pedagogical value has to be evaluated. We are negotiating an evaluation of the system in collaboration with the Canadian Space Agency. Second, the diagnosis process can be improved by explicitly connected declarative knowledge of the domain to the paths provided by FADPRM. This will allow a deep knowledge tracing, thus a fine grained cognitive diagnosis.

## Acknowledgments

The work presented herein was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

## References

[1] K. Forbus, "Articulate Software for Science and Engineering Education," *Smart Machines in Education: The Coming Revolution in Educational Technology,* vol. 1, no. 1, pp. 235-267, 2001.

[2] R. Angros, W. Johnson, J. Rickel, and A. Scholer, "Learning Domain Knowledge for Teaching Procedural Skills," *Proc. First Int'l Conf. Autonomous Agents and Multiagent Systems (AAMAS),* pp. 1372-1378, 2002.

[3] K. VanLehn, "The Advantages of Explicity Representing Problem Spaces," *Proc. Ninth Int'l Conf. User Modeling (UM),* p. 3, 2003.

[4] R. Crowley, E. Legowski, O. Medvedeva, E. Tseytlin, E. Roh, and D. Jukic, "An Its for Medical Classification Problem-Solving: Effects of Tutoring and Representations," *Proc. 12th Int'l Conf. Artificial Intelligence in Education,* pp. 192-199, 2005.

[5] H. Simon, "The Structure of Ill Structured Problems," *Artificial Intelligence,* vol. 4, no. 3, pp. 181-201, 1973.

[6] P. Fournier-Viger, R. Nkambou, and E.M. Nguifo, "Supporting Tutoring Services in Ill-Defined Domains," *Advances in Intelligent Tutoring Systems,* Nkambou et al., eds., Springer, 2010.

[7] K. Koedinger, J. Anderson, W. Hadley, and M. Mark, "Intelligent Tutoring Goes to School in the Big City," *Int'l J. Artificial Intelligence in Education,* vol. 8, no. 9, pp. 30-43, 1997.

[8] A. Mitrovic, M. Mayo, P. Suraweera, and B. Martin, "Contraint-Based Tutors: A Success Story," *Proc. 14th Int'l Conf. Industrial, Eng. and Other Applications of Applied Intelligent Systems (IEA/AIE),* pp. 931-940, 2001.

[9] K. Belghith, F. Kabanza, L. Hartman, and R. Nkambou, "Anytime Dynamic Path-Planning with Flexible Probabilistic Roadmaps," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA),* pp. 2372-2377, 2006.

[10] L. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars, "Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces," *IEEE Trans. Robotics and Automation,* vol. 12, no. 4, pp. 566-580, Aug. 1996.

[11] J. Brown, R. Burton, and F. Zdybel, "A Model-Driven Question-Answering System for Mixed Initiative Computer-Assisted Instruction," *IEEE Trans. Systems, Man and Cybernetics,* vol. 3, no. 3, pp. 248-257, May 1973.

[12] W. Clancey, *Tutoring Rules for Guiding a Case Method Dialogue,* D. Sleeman and J. Brown, eds. Academic, 1982.

[13] F. Kabanza, K. Belghith, P. Bellefeuille, B. Auder, and L. Hartman, "Planning 3D Task Demonstrations of a Teleoperated Space Robot Arm," *Proc. 18th Int'l Conf. Automated Planning and Scheduling (ICAPS),* pp. 164-173, 2008.

[14] F. Bacchus and F. Kabanza, "Using Temporal Logics to Express Search Control Knowledge for Planning," *Artificial Intelligence,* vol. 116, nos. 1/2, pp. 123-191, 2000.

[15] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT , 2005.

[16] M. Likhachev, D. Ferguson, G.J. Gordon, A. Stentz, and S. Thrun, "Anytime Search in Dynamic Graphs," *Artificial Intelligence,* vol. 172, no. 14, pp. 1613-1643, 2008.

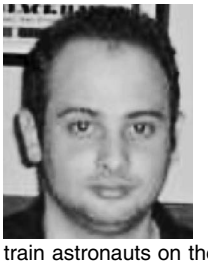
[17] M. Saha, J. Latombe, Y. Chang, and F. Prinz, "Finding Narrow Passages with Probabilistic Roadmaps: The Small-Step Retraction Method," *J. Autonomous Robots,* vol. 19, no. 3, pp. 301-319, 2005.

[18] S.M. Lavalle, *Planning Algorithms.* Cambridge Univ., 2006.

[19] G. Sanchez and J. Latombe, "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking," *Proc. 10th Int'l Symp. Robotics Research (ISRR),* pp. 403-417, 2001.

[20] M. Likhachev, D.I. Ferguson, G.J. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," *Proc. 15th Int'l Conf. Automated Planning and Scheduling (ICAPS),* pp. 262-271, 2005.

[21] E. Larsen, S. Gottshalk, M. Lin, and D. Manocha, "Fast Proximity Queries with Swept Sphere Volumes," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA),* pp. 3719-3726, 2000.
[22] D. Christianson, S. Anderson, L. He, D. Salesin, D. Weld, and C.M.F., "Declarative Camera Control for Automatic Cinematography," *Proc. 13th Nat'l Conf. Artificial Intelligence (AAAI/IAAI),* pp. 148-155, 1996.
[23] W. Bares, L. Zettlemoyer, D. Rodriguez, and J. Lester, "Task-Sensitive Cinematography Interfaces for Interactive 3D Learning Environments," *Proc. Third Int'l Conf. Intelligent User Interfaces (IUI),* pp. 81-88, 1998.
[24] D. Friedman and Y. Feldman, "Automated Cinematic Reasoning about Camera Behavior," *J. Expert Systems with Applications,* vol. 30, no. 4, pp. 694-704, 2006.
[25] A. Jhala and R. Young, "A Discourse Planning Approach to Cinematic Camera Control for Narratives in Virtual Environments," *Proc. 20th Nat'l Conf. Artificial Intelligence (AAAI/IAAI),* pp. 307-312, 2005.
[26] A. Corbett, K. Koedinger, and J. Anderson, "Intelligent Tutoring Systems," *Handbook of Human-Computer Interaction,* M. Helander, T.K. Landauer, P. Prabhu, eds., Elsevier Science, 1997.
[27] R. Nkambou, "Modeling the Domain: An Introduction to the Expert Module," *Advances in Intelligent Tutoring Systems,* Nkambou et al., eds., Springer, 2010.
[28] A. Lesgold, G. Eggan, S. Katz, and G. Rao, "Possibilities for Assessment Using Computer-Based Apprenticeship Environments," *Cognitive Approaches to Automated Instruction,* J. Regian and V. Shute, eds., Lawrence Eribaum Assoc., 1992.



**Khaled Belghith** received the electrical engineering diploma (Dîplome Ingénieur) from l'École Nationale d'Ingénieurs de Monastir, Tunisia, in 2002, the MSc degree in computer science from the University of Quebec at Montreal in 2004, the Executive MBA degree from the University of Quebec at Montreal in 2007, and the PhD degree in computer science from the University of Sherbrooke in June 2010. His thesis is about a simulator prototype for telerobotics training to train astronauts on the manipulation of the Canadian Robot Arm within the International Space Station. He has worked since 2007 as an AI software engineer at Ubisoft. He took part in the development of several games produced at Ubisoft Montreal's Studio.

**Roger Nkambou** received the PhD degree in 1996 in computer science from the University of Montreal. He is currently a full professor of computer science at the University of Quebec at Montreal and the director of the GDAC Laboratory (http://gdac.dinfo.uqam.ca). He is also the chair of graduate studies (PhD program in cognitive computing). His research interests include knowledge representation, intelligent tutoring systems, intelligent software agents, ontology engineering, student modeling, and affective computing. He also serves as a member of the program committees of the most important international conferences in artificial intelligence in education.

**Froduald Kabanza** is a professor of computer science at the Université de Sherbrooke. His research concerns artificial intelligence (AI), mainly in the areas of automated planning, AI architectures, intelligent decision support systems, and intelligent tutoring systems. He has led various successful research projects on these topics. He is currently developing AI techniques with applications to training medical students on clinical diagnosis, training astronauts on robot manipulation, supporting the optimal prescription of antibiotics in hospitals, and controlling mobile robots. He founded MENYA Solutions, a company that offers software developments and services in artificial intelligence (www.menyasolutions.com).

**Leo Hartman** received the PhD degree in computer science from the University of Rochester in 1990. His dissertation investigates the use of decision theory for allocating computational resources in deductive planning. After a postdoc at the University of Waterloo, he joined the Canadian Space Agency in 1993 as a research scientist. His work there focuses on computing, networking, and automation for space missions.