Generating tutoring feedback in an intelligent training system on a Robotic Simulator

Roger Nkambou[1], Khaled Belghith[2], Froduald Kabanza[2]

[1] Université du Québec à Montréal
Montréal, Québec H3C 3P8, Canada
nkambou.roger@uqam.ca
[2] Université de Sherbrooke,
Sherbrooke, Québec J1K 2R1, Canada
khaled.belghith@usherbrooke.ca; kabanza@usherbrooke.ca

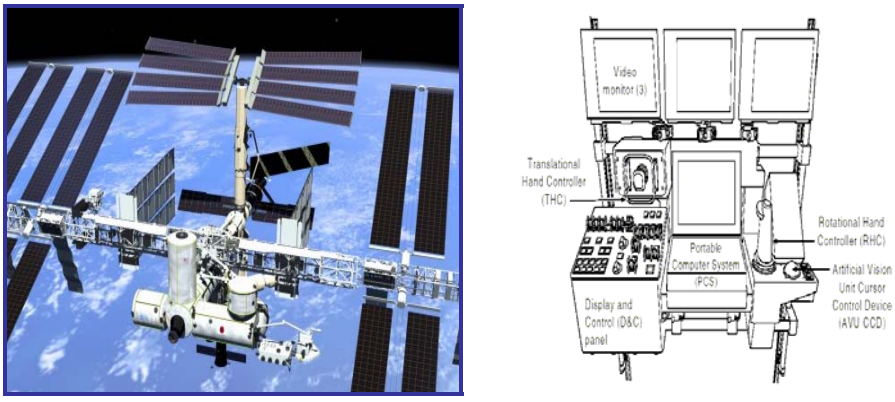
Abstract. Manipulating the Space Station Remote Manipulator (SSRMS), known as "Canadarm II", on the International Space Station (ISS) is a very challenging task. The astronaut does not have a direct view of the scene of operation and must rely on cameras mounted on the manipulator and at strategic places of the environment where it operates. In this paper, we present *Roman Tutor*, an intelligent robotic simulator we are developing to address this kind of problem. We also show how a new approach for robot path planning called FADPRM could be used to provide amazingly useful tutoring feedback for training on such a manipulator and under this big constraint of restricted sight.

1 Introduction

This paper presents *Roman Tutor*, an intelligent 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). Fig. 1-a illustrates a snapshot of the SSRMS on the ISS. Astronauts operate the SSRMS through a workstation located inside one of the ISS compartments. As illustrated on Fig. 1-b, the workstation has an interface with three monitors, each connected to a camera placed at a strategic location of the ISS. There are a total of 14 cameras on the ISS, but only three of them are seen at a time through the workstation. A good choice of the camera on each of the three monitors is essential for a correct and safe operation of the robot.

SSRMS can be involved in various tasks on the ISS, ranging from moving a load from one place of the station to another, to inspect 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 collision with the ISS structure and to maintain safety-operating constraints on SSRMS (such as avoiding collisions with itself and singularities). At different phases of a given manipulation such as moving a payload using the arm (Fig. 2), the astronaut must choose a 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 visual cues on the station that are crucial in mentally reconstructing the actual

working environment from just three monitors each giving a partial and restricted view, and to remember and be able to select cameras depending on the task and other parameters.



(a) **(b)**

Fig. 1. ISS with SSRMS (a) and the workstation (b)

One challenge in developing a good training simulator is of course to build it so that one can reason on it. This is even more important when the simulator is built for training purpose [1]. Up until now, Simulation-Based Tutoring is possible only if there is an explicit problem space associated with tasks that are carried out during training to be able to track student actions and to generate relevant tutoring feedbacks [2,3]. Knowledge and model tracing are only possible in these conditions [4]. However, it is not always possible to explicitly develop a well-informed task structure in some complex domains, especially in domains where spatial knowledge is used, as there are too many possibilities to solve a given problem. This paper proposes a solution to this issue through a system called *RomanTutor* which uses a path planner to support spatial reasoning on a simulator and make it possible model tracing tutoring without an explicit task structure. We developed a simulator of the ISS and SSRMS with quite realistic rendering and kinematics constraints as with the real environment. The Simulator knowledge structure will be described later. Fig. 2 illustrates a snapshot of the simulator with SSRMS moving a payload from one place to another on the ISS. This simulates an operation that actually took place during the construction of the ISS.

As most complex tasks deal in one way or another with moving the SSRMS and for the simulator to be able to understand students' operations in order to provide feedback, it must itself be aware of the space constraints and be able to move the arm by itself. A path-planner that calculates arm's moves without collision and consistent with best available cameras views is the key training resource on which other resources and abstract tutoring processes hinge.

After a brief description of the path planner, we outline the different components of *Roman Tutor* and show how the FADPRM path planner is used to provide amazingly relevant tutoring feedback to the learner.

2 The FADPRM Path-Planner

In the literature, several approaches dealing with the path-planning problem for robots in constrained environments were found [5-7]. Several implementations were carried out on the basis of these various approaches and much of them are relatively effective and precise. The fact is that none of these techniques deals with the problem of restricted sight we are faced with in our case [8].

That's why we designed and implemented FADPRM [9] a new flexible and efficient approach for robot path planning in constrained environments. In more of the obstacles the robot must avoid, our approach holds account of desired and non-desired (or dangerous) zones. This will make it possible to take into account the disposition of the cameras on the station. Thus, our planner will try to bring the robot in zones offering the best possible visibility of the progression while trying to avoid zones with reduced visibility (Fig. 2).

FADPRM [9] allows us to put in the environment different zones with arbitrary geometrical forms. A degree of desirability *dd*, a real in [0 1] is assigned to each zone. The *dd* of a desired zone is then near 1, and the more it approaches 1, the more the zone is desired; the same for a non-desired zone where the *dd* is in [0 0.5]. On the international Space Station, the number, the form and the placement of zones reflect the disposition of cameras on the station. A zone covering the field of vision of a camera will be assigned a high *dd* (near 1) and will take a shape which resembles that of a cone (Fig. 2); whereas a zone that is not visible by any camera from those present on the station will be considered as an non-desired zone with a *dd* near to 0 and will take an arbitrary polygonal shape (a cube in Fig.2).

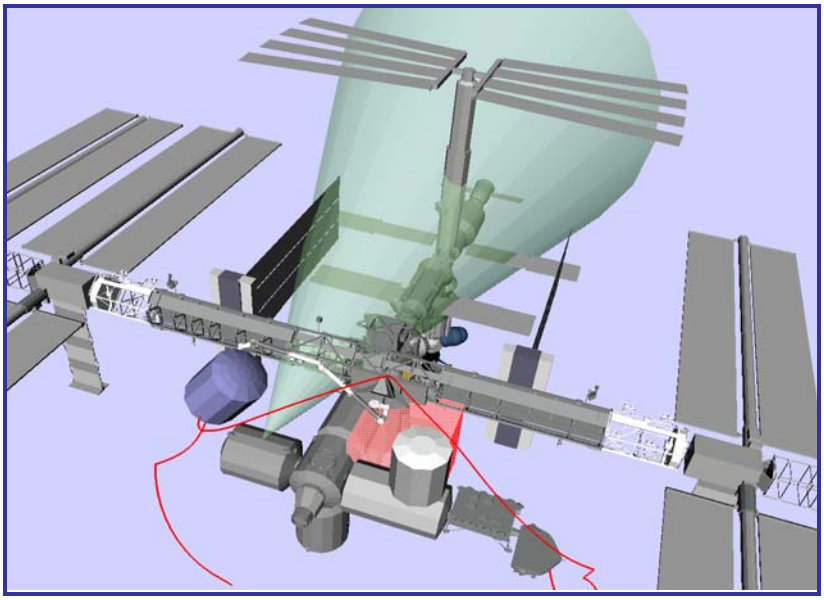


Fig. 2. SSRMS avoiding a non-desired zone (cube) and going through a desired zone (cone)

The robot workspace (the ISS environment here) is then preprocessed into a roadmap of collision-free robot motions in regions with highest desirability degree. More precisely, the roadmap is a graph such every node *n* is labeled with its corresponding robot configuration *n.q* and its degree of desirability *n.dd*, which is the average of *dd*s 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) [10], 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 configurations.

Following the Anytime Dynamic A* (AD*) approach [11], 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 [10] and AD* [11] and it is flexible in that it takes into account zones with degrees of desirability. This explains why we called 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 configurations, we search backward from the goal towards the initial (current) robot configuration. Backward instead of forward search is done because the robot is moving, 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 on the current roadmap's frontier (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

$$(1-\beta) / \text{density}(n) + \beta * \text{goal-distance-estimate}(n) \qquad \text{with } 0 \leq \beta \leq 1.$$

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

[10]. 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 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 redoing them.
- 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, moving arm, or changes in *dd* for zones) cause updates of the roadmap and re-planning.

We implemented FADPRM as an extension to the Motion Planning Kit (MPK)[10] by changing the definition of PRM to include zones with degrees of desirability and changing the algorithm for searching the PRM with FADPRM. 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)[12].

3 Roman Tutor

3.1 Roman Tutor architecture

Roman Tutor's architecture contains seven main components: the simulator, the FADPRM path planner, the movie generator, the task editor, the student model and the tutoring sub-system.

As most knowledge related to the simulation-based robotic manipulation in *Roman Tutor* is mainly consistent with spatial reasoning, an appropriate structure is needed for the simulator. We equipped our system with the FADPRM path planner, which as we said before will provide a framework to support the reasoning process within the simulator. However, this reasoning base won't be sufficient to bring useful tutoring explanations to guide and orient the astronaut during his manipulations. The level of explanation that could be given here remains very limited because the planner is connected at the basic level of the simulator which is made up essentially of physical components in the environment such as the robot, obstacles, cameras, zones, and some related low-level parameters and variables such as the robot's configuration, its degree of desirability and whether it is colliding or not. This level is equivalent to the structure proposed by Forbus [13]. However a proper use of FADPRM could exploits information in that level to generate very important tutoring feedbacks without any hard-coded structure.

3.2 Roman Tutor user interface

The *Roman Tutor* interface (Fig. 3) resembles that of the robotic workstation on which the astronaut operates to manipulate the SSRMS. We have three monitors each of them connected to one of the fourteen cameras present on the station. On each monitor, we have buttons and functionalities to move the corresponding camera: Tilt, Pan and Zoom.

SSRMS can be manipulated in two modes: the For mode or the Joint-by-Joint mode. In the For mode, the learner moves the robot starting from his end-effector's position and by commanding him to go forward or backward, left or right and up or down. In the Joint-by-Joint mode, the learner selects a joint in the robot and moves it according to the corresponding link. In the two modes, the manipulation is done incrementally. While manipulating the robot, the astronaut can choose and change the camera in each monitor to have a better sight of the zone he is working in.

Windows at the bottom of the *Roman tutor* interface contain the trace done so far by the learner. Every operation done by the learner is posted on this window: the selection of a new camera in a monitor, the displacement of a camera and the manipulation of the robot in the For/Joint-by-Joint mode. This trace contains also all information relevant to the current state: if there is a collision or not, the coordinates of the End-Effector, the position and the orientation of the cameras (Fig. 3).

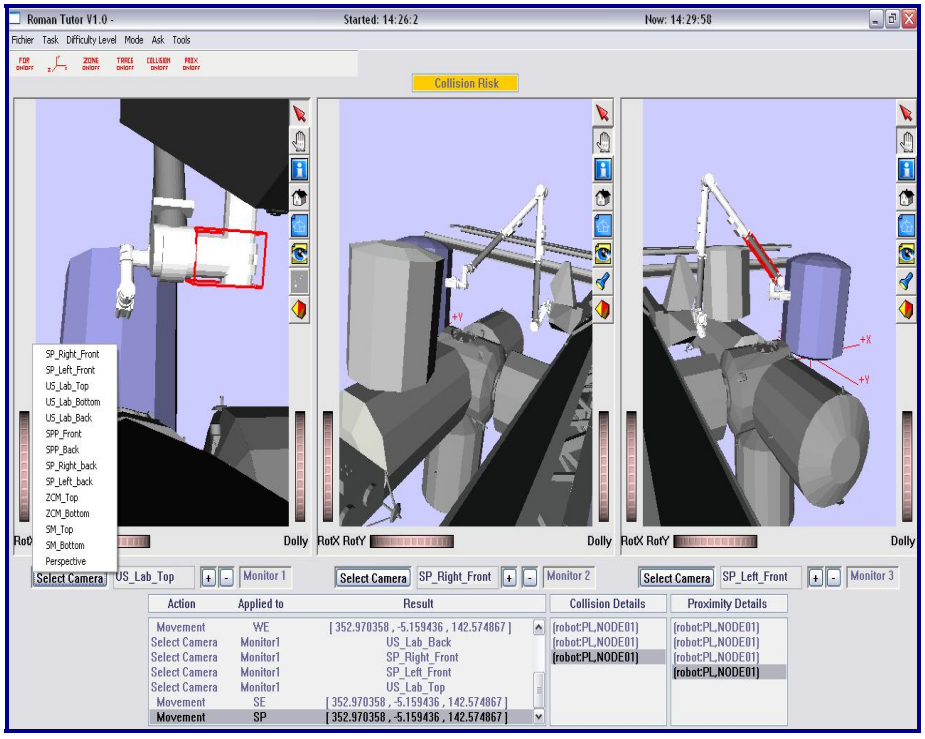


Fig. 3. Roman Tutor User Interface

The Trace window keeps then a continuous track of all operations done so far by the learner. So if he had done a mistake (a collision for example), by examining the Trace window, he will find out to what this was due and he will see what to do to get out of it.

Roman Tutor's interface contains four main menus: Task, Mode, Ask and Tools. From the menu 'Task', the learner chooses one of the several tasks he wants to work on. From the second menu 'Mode', the learner chooses between two different modes: Free and Assisted. In the 'Assisted' mode, the non-cognitive tutor intervenes when needed to support the learning of the student by guiding him or by giving him an illustrated video of the task he has to do. Whereas in the 'Free' mode, the learner relies only on the Trace window to carry on his task. In the two modes, the 'Ask' menu allows the learner to ask different types of questions while executing a task. In the last menu 'Tools', the expert is provided with three different tools that will help him design and validate new tasks to add into the system. These different tools are: the FADPRM Path-Planner, the Movie Generator and the Task Generator.

4 Using FADPRM Path-Planner for the Tutoring Assistance

One of the main goals of an intelligent tutoring system is to actively provide relevant feedback to the student in problem solving situations [3]. This kind of support becomes very difficult when an explicit representation of the training task is not available. This is the case in the ISS environment where the problem space associated to a given task consists of an infinite number of paths. Moreover, there is a need to generate new tasks (or sub-tasks) on the fly without any cognitive structure. *Roman Tutor* brings a solution to these issues by using FADPRM as principal resource for the tutoring feedback.

Roman Tutor includes four different types of tasks on which we may train astronauts: Spatial Awareness, GoTo, Inspect and Repair.

The 'Spatial Awareness' task improves the learner's knowledge about the space station's environment by providing him with some exercises such as naming and locating ISS elements, zones and cameras. This is a very important activity since astronauts don't have a complete view of the station while manipulating the robot and must memorize a spatial model of the ISS in order to execute the different tasks. The tutor can invoke this type of activity when it notices a lack of understanding in the student profile about some environment-related knowledge. In Fig. 4, the learner is asked to name an element of the station and to select the camera from which it is shown.

In the 'GoTo' task, the learner has to move the SSRMS, carrying a load or not, from one position to another different on the ISS. Inspect and Repair tasks are variants of the 'GoTo' task. In 'Inspect', the astronaut is trained on how to go towards an element or a zone in the station and how to inspect it at several different points. In the 'Repair' task, the astronaut is trained on how to go towards an element of the station and how to execute some repairs on it at several points using the manipulator. These "GoTo"-like tasks are well described in the next paragraph.

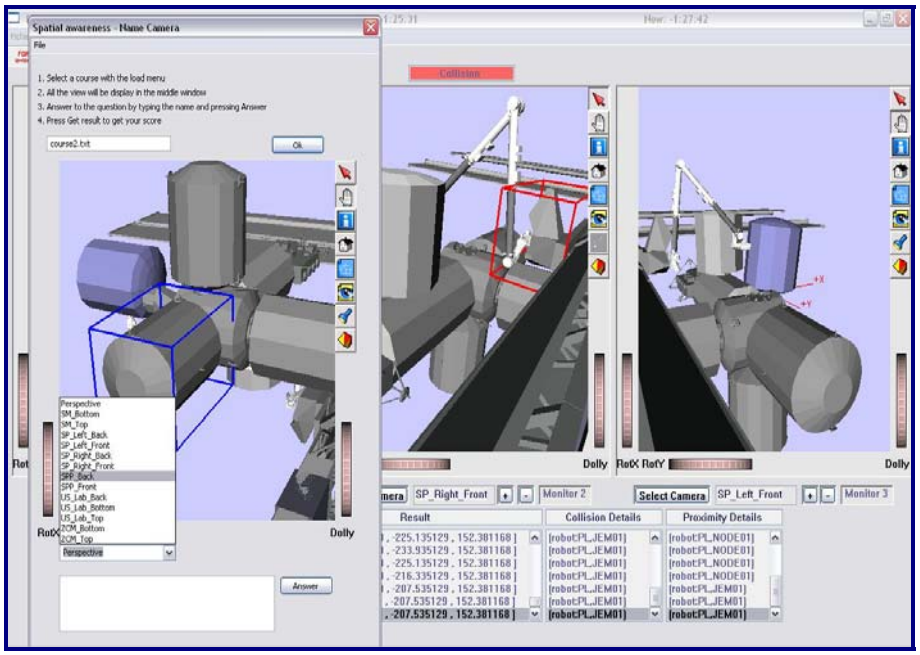


Fig. 4. 'Spatial Awareness' Task in Roman Tutor

During a 'GoTo' task, the non-cognitive *Tutor* uses the anytime capability of the FADPRM path planner to validate incrementally student's action or sequence of actions, give information about the next relevant action or sequence of actions, and generate relevant task demonstration resources using the movie generator. In Fig. 5, the learner is shown an illustration (demonstration) of the task he's working on. The movie is created so that the end-effector will follow the generated path (in red).

RomanTutor gives access to a menu that allows the learner to ask some relevant questions during manipulations. These questions may be of three different forms: How To, What if and Why Not. Several extensions are associated to these different questions. For example, with 'How to', one could have: How to Go To, How to avoid obstacle, How to go through zone. *Roman Tutor* answers How-To questions by generating a path consistent with the best cameras views using FADPRM and by calling the movie generator to build an interactive animation that follows that path. The incremental planning capability of FADPRM is used by *Roman Tutor* to bring answers to the What-If and Why-Not questions. In both cases, *Roman Tutor* provides the learner with relevant illustrations explaining whether his action or sequence of actions is appropriate or out of scope (thus may bring him to a dead end) given the generated plan that corresponds to the correct path.

We see here the importance of having FADPRM as a planner in our system to guide the evolution of the astronaut. By taking into account the disposition of the cameras on the station, we are assured that the proposed plan the learner is following

passes through zones that are visible from at least one of the cameras placed on the environment.

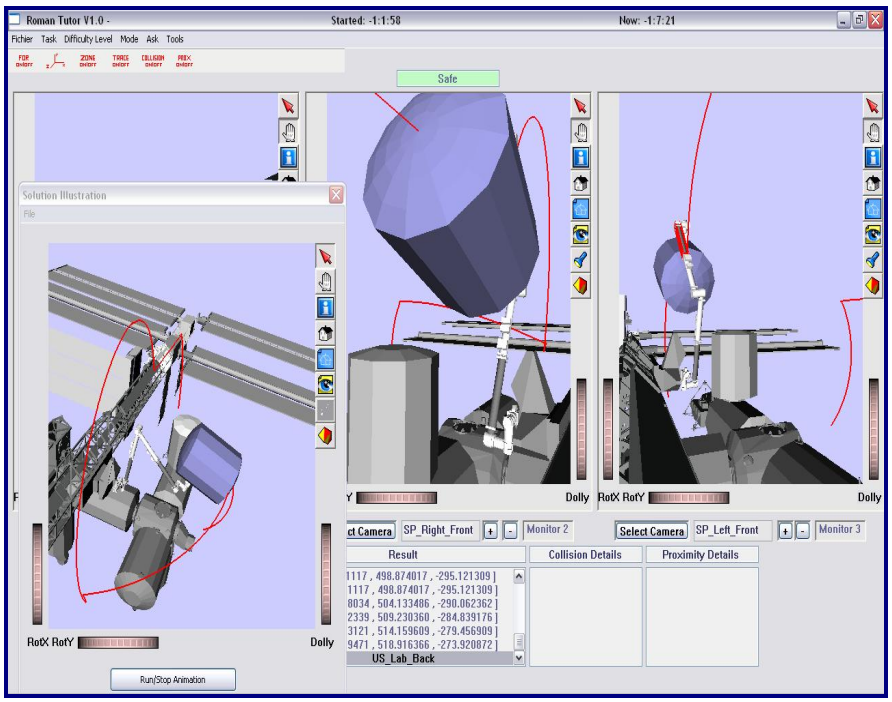


Fig. 5. 'GoTo' Task in Roman Tutor

5 Conclusion

In this paper, we described how a new approach for robot path planning called FADPRM could play an important role in providing tutoring feedback for training on a robot simulator. The capability of our path-planner, built within the simulator's architecture to provide predictions and to determine what manipulations might achieve a desired effect, makes it a useful component in simulation-based training systems. We also, detailed the architecture of the intelligent tutoring system *Roman Tutor* in which FADPRM is integrated. All the components of *Roman Tutor* but the cognitive tutor, are working properly.

This constitutes a very important contribution especially in the field of intelligent tutoring systems: we showed that it is not necessary to plan in advance what feedback to give to the learner or to explicitly create a complex task graph to support the tutoring process.

The next step is to improve the expressiveness of domain knowledge structures under SSRMS to lead to a better diagnosis process of problems experienced by the

learner during manipulations. Also, an evaluation of the system will be done in collaboration with the Canadian Spatial Agency.

Acknowledgment. We would like to thank the Canadian Space Agency, The Infotel Group and the Natural Sciences and Engineering Research Council of Canada (NSERC) for their logistic and financial support. Special thanks to Mahie Khan, Daniel Dubois, Patrick Hohmeyer and Kaufmann Meudja, who contributed in the design of this system. Special thanks also to all members of J.C. Latombe's Laboratory at the University of Stanford for providing as with MPK.

References

1. Forbus, K.: Articulate software for science and engineering education. In Forbus, K., Feltovich, P., and Canas, A. (Eds.) Smart machines in education: The coming revolution in educational technology. AAAI Press (2001)
2. Richard Angros, W. Lewis Johnson, Jeff Rickel, Andrew Scholer: Learning domain knowledge for teaching procedural skills. AAMAS, 1372-1378 (2002)
3. VanLehn, K.: The advantages of Explicity Representing Problem Spaces. User Modeling, Springer Verlag LNAI 2702:3. (2003)
4. Crowley R.S., Medvedeva, O., Jukic, D.: SlideTutor - A model-tracing Intelligent Tutoring System for teaching microscopic diagnosis. Proceedings of the 11th International Conference on Artificial Intelligence in Education (2003)
5. Latombe, J.C.: Robot motion planning. Kluwer Academic Publishers, Boston, MA, (1991)
6. Overmars, M.H.: Recent developments in motion planning. P. Sloot, C. Kenneth Tan, J. Dongarra, A. Hoekstra (Eds.): Computational Science - ICCS 2002, Part III, Springer-Verlag, LNCS 2331, (2002) 3-13
7. LaValle, S. M., Branicky, M. S., Lindemann, S.R.: On the relationship between classical grid search and probabilistic roadmaps. International Journal of Robotics Research 23: 673–692, (2004)
8. Roy, J., Nkambou, R., Kabanza, F.: Supporting spatial awareness in training on a telemanipulator in space. Intelligent Tutoring Systems, LNCS 3220, 860-863, Springer-Verlag Berlin Heidelberg (2004)
9. Belghith, K., Kabanza, F., Hartman, L., Nkambou, R.: Anytime Dynamic Path-Planning with Flexible Probabilistic Roadmaps. In Proc. of IEEE International Conference on Robotics and Automation, ICRA (2006)
10. Sanchez, G., Latombe, J.C.: A single-query bi-directional probabilistic roadmap planner with lazy collision checking. Int. Symposium on Robotics Research (ISRR'01), Lorne, Victoria, Australia, November 2001. Published in Robotics Research: The Tenth Int. Symp. R.A. Jarvis and A. Zelinsky (eds.), Springer Tracts in Advanced Robotics, Springer, 403-417 (2003)
11. Likhachev, M., Ferguson, D., Stentz, A., Thrun, S.:. Anytime Dynamic A*: An Anytime Replanning Algorithme. In Proc. of International Conference on Automated Planning and Scheduling, June (2005)
12. Larsen, E., Gottschalk, S., Lin, M., Manocha, D.: Fast Distance Queries using Rectangular Swept Sphere Volumes. In Proc. of IEEE International Conference on Robotics and Automation, 4:24-28 (2000)
13. Forbus, K.: Using qualitative physics to create articulate educational software. *IEEE Expert*, May/June, 32-41 (1997)