An Approach to Intelligent Training on a Robotic Simulator using an Innovative Path-Planner

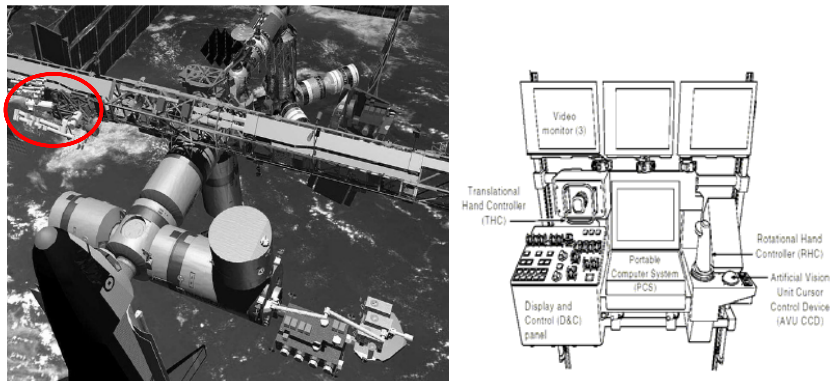
Roger Nkambou[1], Khaled Belghith[2], Froduald Kabanza[2]
[1] Université du Québec à Montréal
Montréal, Québec H3C 3P8, Canada
nkambou.roger@uqam.ca
2 Université de Sherbrooke,
Sherbrooke, Québec J1K 2R1, Canada
khaled.belghith@usherbrooke.ca; kabanza@usherbrooke.ca

Abstract. In this paper, we describe the open knowledge structure of *Roman Tutor*, a simulation-based intelligent tutoring system we are developing to teach astronauts how to manipulate the Space Station Remote Manipulator (SSRMS), known as "Canadarm II", on the International Space Station (ISS). We show that by representing the complex ISS-related knowledge in the form of a three-layered architecture with different levels of abstraction, and by using a new approach for robot path planning called FADPRM, it is no longer 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.

1. Introduction

This paper presents *Roman Tutor*, 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). Fig. 1-a illustrates a snapshot of the SSRMS on 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.


Video monitor (3)
Translational Hand Controller (THC)
Rotational Hand Controller (RHC)
Artificial Vision Unit Cursor Control Device (AVU CCD)
Portable Computer System (PCS)
Display and Control (D&C) panel

(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 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 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 [7-9]. 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 dealing with in our case [10].

That's why we designed and implemented FADPRM [11] 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 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.

FADPRM [11] 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; 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.

The ISS environment 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 that 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) [12], 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 [13], 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 that the planner can be called 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 [12] and AD* [13] 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).

We implemented FADPRM as an extension to the Motion Planning Kit (MPK)[12] 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's *dd* and a path's *dd* is a straightforward extension of collision checking for configurations and path segments. For this, we customized the Proximity Query Package (PQP)[14]. In the next section, we show how FADPRM is used as a tutoring resource within *Roman Tutor*.

3. Roman Tutor

3.1 Roman Tutor architecture

Roman Tutor's architecture contains six main components (Fig. 2): the simulator, the FADPRM path planner, the movie generator, the task editor, the student model and the tutoring sub-system.

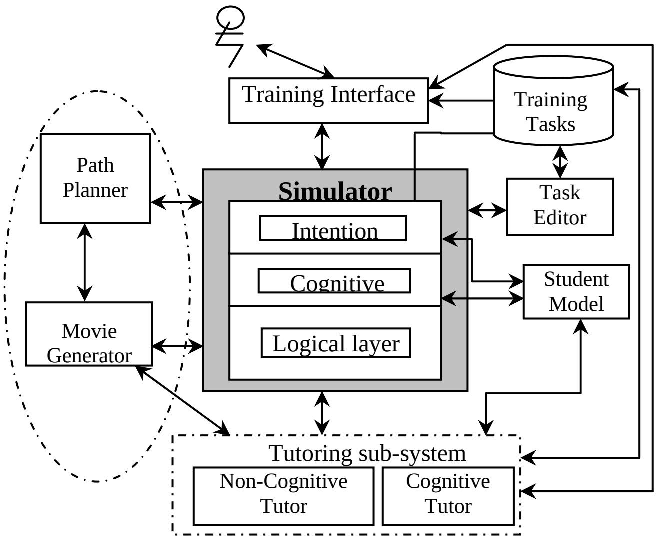


Fig. 2. Roman Tutor Architecture

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 a path planner, FADPRM, 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 logical level (level 1) of the simulator which is made up essentially of physical components in the environment such as the robot, the obstacles, the cameras and the zones, and some related low-level parameters and variables such as the configuration of the robot, the degree of desirability and whether there is a collision or not. This level is equivalent to the structure proposed by Forbus [15]. As we already said, a useful way to better support spatial reasoning is to extract and explicitly represent qualitative descriptions of shape and space. This is also known as *spatial aggregation* [5,6]. We did this by adding 2 other levels within the whole architecture for an elaborate cognitive support: the cognitive level and the intentional level.

The cognitive level (level 2) corresponds to an aggregation of the logical level in terms of zones and corridors of safe operation annotated by different domain knowledge elements. Corridors define portions of path the learner could raise during the manipulation of the robot from one place to another on the station. They are defined generally according to the geometry of the environment, the density and location of obstacles and to the choice of the spatial orientation predefined in the ISS environment (Which axis defines going down or up for example? The x-axis? The z-axis?). These corridors are annotated with high-level data providing an elaborated basis for the explanation process as said before, such as visibility through cameras and some environment-related knowledge (proximity to main obstacles, etc.). Zones define precise places of the station with particular characteristics we have to raise during the explanation process in order to draw the learner's attention on some specific knowledge in the environment such as narrow passages, or very small devices to which we have to pay a particular attention like antennas for example. The whole environment is thus aggregated into various areas annotated with appropriate knowledge in order to get more semantic richness in guiding the astronaut during displacements of the arm (Fig. 3).

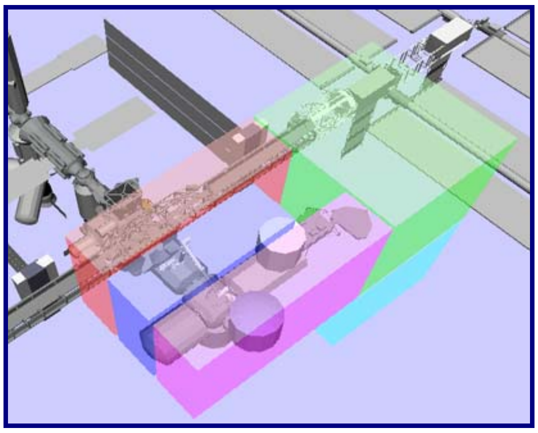


Fig. 3. Space aggregation into corridors and zones

The intentional level (level 3) contains structures of predefined tasks. The task's structure is a problem space made up a set of corridors and zones defined in level 2. The intentional level makes it possible to better follow the evolution of the astronaut in the execution of the selected task. More specifically, a task's problem space consists of a set of sequences of corridors the astronaut could undertake and a set of zones that could be related to these robot's displacements. For example, in a task for moving a load from one place to another, the learner could make the robot go through corridor 1 until reaching a specific element of the ISS, then follow corridor 2 until the robot becomes visible through the camera on the right corner of the Truss and reach the goal. Another path to execute the same task could be just going throw corridor 3 until reaching the final configuration while paying attention to zone 3 containing a tiny antenna.

The student model makes it possible to keep track of the learner's domain knowledge acquisition during trainings in order to better adapt the interaction. In addition to keeping a record on the performances of the learner, this model also stores environment-related knowledge the student acquired/understood and information on what procedures or concepts he used. It thus refers to levels 2 and 3.

The expert designs and publishes new tasks by accessing the Task Editor menu. The creation of a new task automatically requires the creation of a related abstract knowledge structure. The latter is defined at level 3 and refers to elements of level 2. We however left the possibility of creating new tasks directly in level 2 (without explicit task structures). Such a task does not allow a tutorial reasoning at the 3rd level of the simulator but is very suitable in the free practice mode where the astronaut could generate new tasks if needed.

The movie generator is a very important tutoring resource in the context of our simulator. It takes a plan computed by the path planner according to the demonstration needs and generates a movie illustrating the path proposed through appropriate cameras.

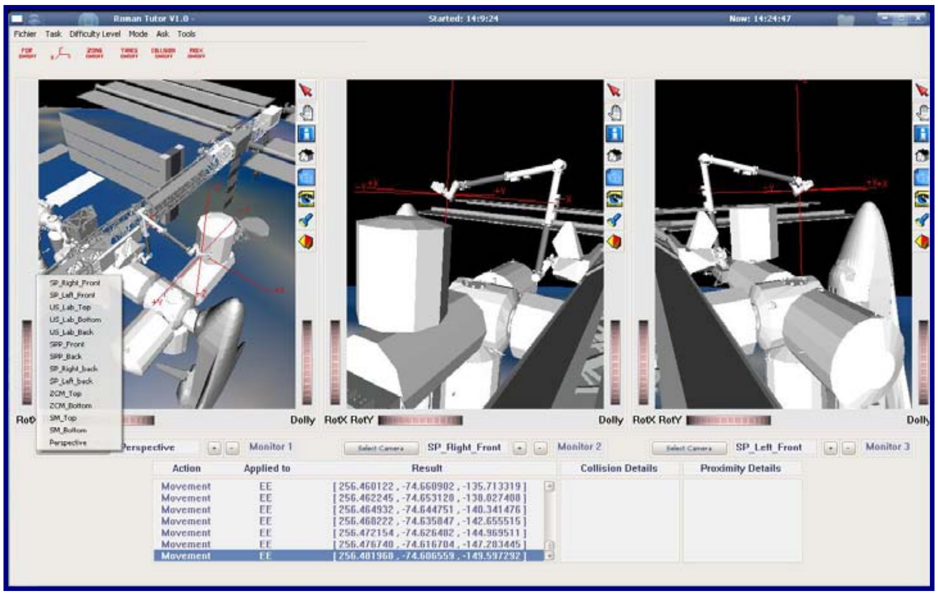
The tutoring sub-system consists of a cognitive tutor and a non-cognitive tutor. The non-cognitive tutor uses the FADPRM path planner to reason at level 1 of the simulator, whereas the cognitive tutor reasons starting from level 3 and allows thorough explanations during task execution. The rest of the paper focuses on the non-cognitive tutor.

3.2. Roman Tutor user interface

The *Roman Tutor* interface (Fig. 4) 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 link assigned to it. 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 astronaut. 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 about the current state: if there is a collision or not, the coordinates of the End-Effector, the position and the orientation of the cameras.



Action	Applied to	Result	Collision Details	Proximity Details
Movement	EE	[256.460122 , -74.660902 , -135.713319]		
Movement	EE	[256.462245 , -74.653128 , -138.027408]		
Movement	EE	[256.464932 , -74.644751 , -140.341476]		
Movement	EE	[256.468222 , -74.635847 , -142.655515]		
Movement	EE	[256.472154 , -74.626482 , -144.969511]		
Movement	EE	[256.476740 , -74.616704 , -147.283445]		
Movement	EE	[256.481960 , -74.606559 , -149.597292]		

Fig. 4. 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 inspecting the Trace window, he will find out to what this was due to 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 learner by guiding him or by giving him an illustrated video of the task he has to do. 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 with a given task consists of an infinite number of paths. Moreover, there is a need to generate new tasks on the fly without any cognitive structure. *Roman Tutor* brings a solution to these issues by using FADPRM as main resource for the tutoring feedback.

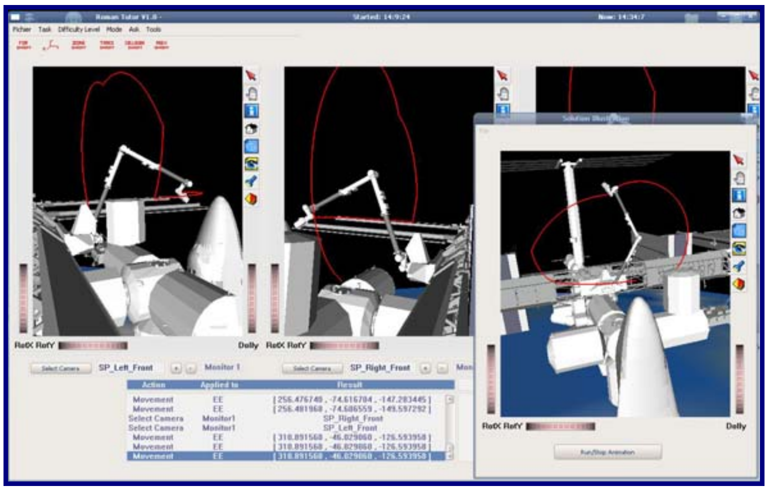
4.1. Training Tasks in Roman Tutor

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. In the 'Assisted' mode, the non-cognitive tutor invokes this type of activity when it notices a lack of understanding in the student profile about some environment-related knowledge during the displacements of the robot.

In the 'GoTo' task (Fig. 5), 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.



Action	Applied to	Result
Movement	EE	[256.476740 , -74.616704 , -147.283445]
Movement	EE	[256.481968 , -74.686559 , -149.597292]
Select Camera	Monitor1	SP_Right_Front
Select Camera	Monitor1	SP_Left_Front
Movement	EE	[310.891568 , -46.029060 , -126.593958]
Movement	EE	[310.891568 , -46.029060 , -126.593958]
Movement	EE	[310.891568 , -46.029060 , -126.593958]

Fig. 5. 'GoTo' Task in Roman Tutor

4.2. Continuous Tutoring Assistance in Roman Tutor

As we said in the previous section, the astronaut can choose between two modes: 'Free' and 'Assisted'. In the assisted mode, the non-cognitive tutor continuously assists the learner during his progression in the task.

In ‘GoTo’ tasks for example, 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 of the task he’s working on.

In the ‘Inspect’ Task, the learner has to reach several points on different positions of the station. The non-cognitive tutor calls the FADPRM Planner incrementally between these different positions and provides the learner with different indications that will help him follow the plan linking all these different points.

An analogue scheme is also used (both in the ‘Free’ and ‘Assisted’ mode) with the different questions in the ‘Ask’ menu. 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 explanations given that his action or sequence of actions is out of scope of the generated plan or may bring him to a dead end.

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

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 to a learner during training on a robot simulator. The capability of the path-planner built within the simulator’s architecture to predict and to determine what manipulations might achieve a desired effect makes it a useful component in training systems involving the ‘physical’ world. We also, detailed the architecture of the intelligent tutoring system *Roman Tutor* in which FADPRM is integrated.

This constitutes a very important contribution especially in the field of intelligent tutoring systems. In fact, 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.

Many extensions are under study to improve the performance of our intelligent tutoring simulator. The most important one will be to implement and test the cognitive tutor which exploits levels 2 and 3 of the simulator’s structure. This will allow us to validate the usefulness of these layers in cognitive tutoring. We will also improve the expressiveness of domain knowledge structures under SSRMS within the three layers of the simulator. This will lead to a better problem diagnosis process during training.

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 us with MPK.

References

1. Forbus, K.: Articulate software for science and engineering education. In 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 Int. Conf. on Artificial Intelligence in Education (2003)
5. Bailey-Kellogg, C., Zhao, F.: Qualitative Spatial Reasoning Extracting and Reasoning with Spatial Aggregates. AI Magazine, 24(4), 47-60, AAAI (2003)
6. Kuipers, B.: The Spatial Semantic Hierarchy. Artificial Intelligence 119: 191-233 (2000)
7. Latombe, J.C.: Robot motion planning. Kluwer Academic Publishers, Boston, MA, (1991)
8. Overmars, M.H.: Recent developments in motion planning. Computational Science - ICCS 2002, Part III, Springer-Verlag, LNCS 2331, (2002) 3-13
9. LaValle, S. M., Branicky, M. S., Lindemann, S.R.: On the relationship between classical grid search and probabilistic roadmaps. Int. Journal of Robotics Research 23: 673–692, (2004)
10. 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 (2004)
11. Belghith, K., Kabanza, F., Hartman, L., Nkambou, R.: Anytime Dynamic Path-Planning with Flexible Probabilistic Roadmaps. Proc. IEEE Int. Conf. on Robotics and Automation (2006)
12. 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. Springer Tracts in Advanced Robotics, Springer, 403-417 (2003)
13. Likhachev, M., Ferguson, D., Stentz, A., Thrun, S.: Anytime Dynamic A*: An Anytime Replanning Algorithme. Proc. of Int. Conf. on Automated Planning and Scheduling, 2005
14. Larsen, E., Gottschalk, S., Lin, M., Manocha, D.: Fast Distance Queries using Rectangular Swept Sphere Volumes. In Proc. of IEEE Int. Conf. on Robotics and Automation, 4:24-28, 2000
15. Forbus, K.: Using qualitative physics to create articulate educational software. IEEE Expert, 32-41 (1997)