Representing Robots in Building Construction Simulation

Rudi Stouffs
Stephen R. Lee
Ramesh Krishnamurti
Irving J. Oppenheim
Department of Architecture
*Carnegie Mellon University**

In this paper we elaborate on a general representation for robots in building construction, to simulate the robots' capabilities to operate within different building projects and in cooperation with human labor crews. We introduce motion rules as a way to capture the diversity in the different robot types in a uniform manner. We explore a motion language to model the behavior of robots in the real-world environment of building construction. The motion language builds upon the motion rules, and is used to specify the actions and operations of a manipulator when faced with different tasks and/or situations. The motion language, when combined with a simple collision avoidance algorithm for path planning that is based on a 'generate and test' procedure, allows for the simulation of robots in building construction.

INTRODUCTION

Building construction offers challenging problems in robotics for various reasons, including, site characteristics, man-robot cooperation and robot diversity. Of particular interest to us is this last issue, namely the use of several different and distinct robot agents in the construction of buildings.

The simulation of building construction provides a valuable tool for construction planning and management. The use of robot agents adds an extra dimension to this simulation. Besides allowing for the construction process to be visualized through consecutive projections of the building during construction, it allows for a manifestation of the flow of construction material from storage to assembly, and the manner of assembly itself. Furthermore, simulating the use of robots leads to a more precise analysis of the difficulty and time involved in the transportation of construction material on the site. These inclusions require a comprehensive representation of a diverse set of robots, as well as a general three dimensional motion planning algorithm to determine the necessary robot motions.

The simulation takes a building construction task plan as input. This is a detailed plan describing the construction elements and the construction process as a task schedule. A task is to be performed either by a human crew, or by a robot. The simulator then translates each task into a robot motion plan, i.e., a sequence of motion steps, using a rule-based description of the robot agents. The motion plans reflect the respective robot's motional capabilities and limits, and avoid any collision. The simulation also rejects impossible tasks and allows for a variety of different robot types. The result of the simulation consists of a graphical visualization of the motion plans, and of the building under construction.

The simulation can be used to make observations on the feasibility of each task and to maintain a running measure of construction time and cost. It can also be used to study the productivity of alternate construction plans and alternate resource mixes or robot types (Stouffs et al. 1993).

For the demonstration of the simulator we chose the construction of a typical Japanese precast concrete residential building. Figure 1 illustrates a rendering of three stories of a typical 5 story, 40 unit building; Figure 2 shows the floor plan of a single unit.

* Contact address: Department of Architecture, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh PA 15213, U.S.A.; tel: (412) 268-2360; fax: (412) 268-6129; e-mail: ramesh@arc.cmu.edu.

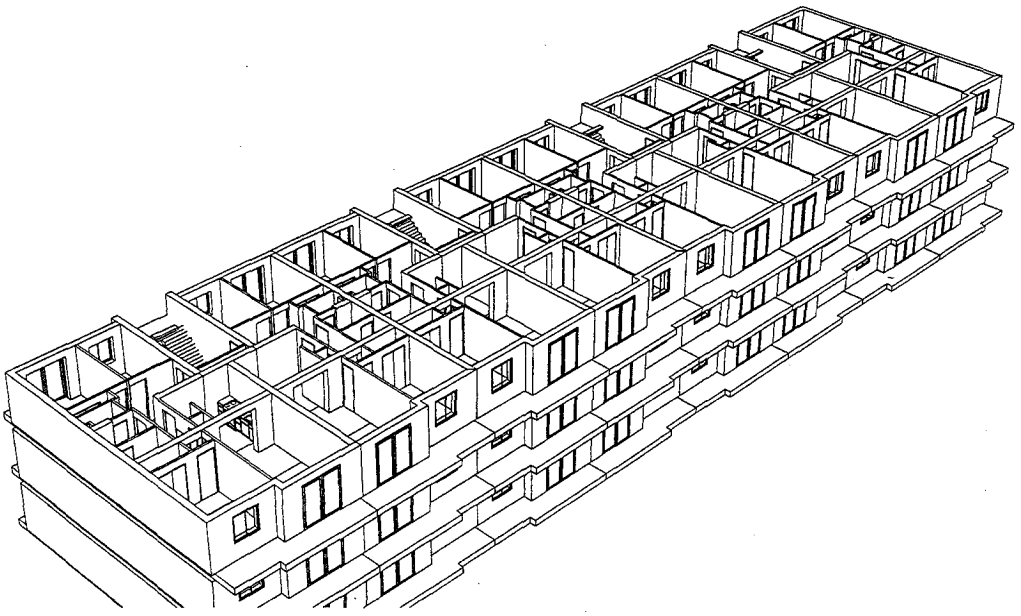

Figure 1: Typical 40 unit building under construction.

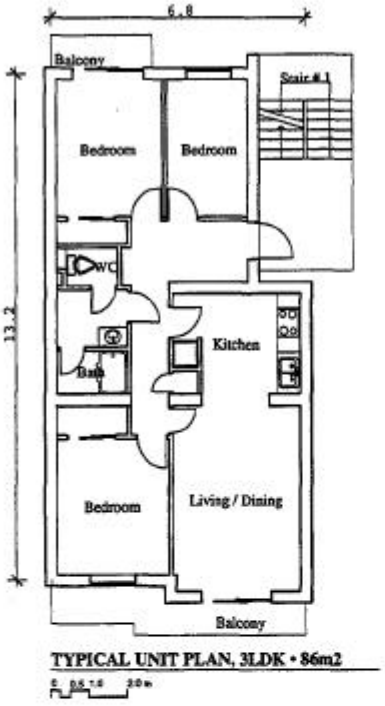

Figure 2: Floor plan of a single unit.

ROBOT MOTIONS

Robots in building construction include any form or type of automated mechanical manipulator characterized by spatial motion, that is employed in the construction process. Robots in building construction may vary greatly in size, mobility, purpose (the tasks or operations they are intended for) and adaptability to different situations within a real-world environment.

In particular, we consider robots for construction that are characterized by their ability to lift, move and place sizable construction components from the delivery location to the placement location. As examples, we consider two different robot types: a robot *crane* for handling heavy components; a robot *towmotor* for palettized materials. These two types are illustrated in Figure 3 and Figure 4, respectively.

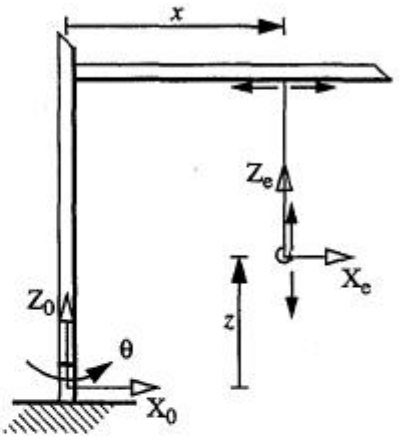

Figure 3: A robot crane.

The description of a robot encompasses a representation of its configuration, i.e., position and orientation, over time, its motional capabilities and its configurational limits.

In order to describe the configuration of a body in space we rigidly attach a coordinate system or *frame* to the body (Craig 1989). A minimal description of a robot involves the following three frames: the universal frame $\{U\}$ represents the outside world and serves as an absolute reference; the base frame $\{B\}$ defines the configuration of the robot's base; the tool frame $\{T\}$ defines the configuration of the robot's end-effector (see Figure 5). The object frame $\{O\}$ defines the configuration of the material being handled.

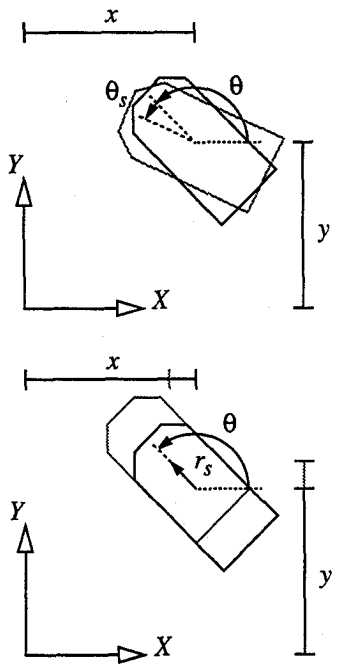

Figure 4: A robot towmotor with (above) stationary rotational motion and (below) translational motion along its axis.

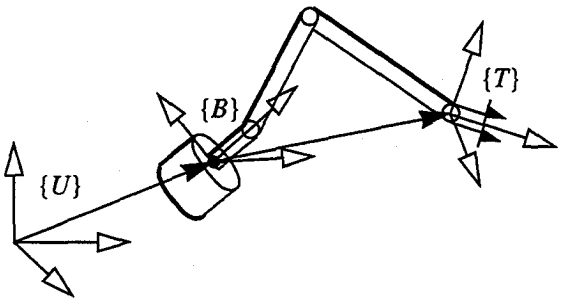

Figure 5: Three frames to describe the configuration of a robot.

A *transform* describes one frame or coordinate system relative to another. The following two transforms are selected to express the configuration of a robot: ${}^{U}_{B}\mathrm{T}$ describes the base frame relative to the universal frame; ${}^{B}_{T}\mathrm{T}$ defines the tool frame with respect to the base frame. The object frame is described relative to the robot's tool frame by the transform ${}^{T}_{O}\mathrm{T}$. Transforms can be composed, e.g., the composite transform ${}^{U}_{O}\mathrm{T} = {}^{U}_{B}\mathrm{T}\ {}^{B}_{T}\mathrm{T}\ {}^{T}_{O}\mathrm{T}$ describes the object frame relative to the universal frame.

The ability of these transforms to change over time and, as a result, the relative configuration of the bodies they describe, is captured in a set of *control parameters*. For each transform, these parameters correspond to the respective degrees of freedom (DOF) of the body relative to its reference frame. Physically, the relationship between the end-effector and the base of the robot is defined by a set of links and joints, generally formed into an open kinematic chain. These joints can be either rotational (change of orientation) or prismatic (translational change). The number of DOF of the robot's end-effector, with respect to the base and for an open kinematic chain, corresponds in most cases to the number of joints. The set of valid control parameters consists of x, y, z, ψ, ϕ and θ, where x, y, z define translations parallel to the respective axes and ψ, ϕ, θ define rotations about the X-, Y- and Z-axis, respectively.

MOTION RULES

In the case of a mobile robot, the motional capabilities of the base of the robot often are dependent on the current configuration of the base and, thus, defined locally. However, the configuration itself is defined relative to a reference frame which is fixed with respect to the body's motion, and is thus defined globally.

Motion rules are conceived to describe a robot's motional capabilities possibly independent of its configuration. Foremost, however, they form a tool to embody the discretization of time in the simulation process. A motion rule generally defines the value of a control parameter at time $t+\Delta t$, where Δt denotes the time step, in terms of the control parameter's values at time t and their initial and step values. The initial values are the values of the control parameters at time 0. Under the assumption of constant velocity, for each control parameter we can specify a motion step as a constant value. Each parameter also has a minimum and maximum value specified in accordance with the relative workspace of the body. This workspace is roughly the volume of space which the body can reach in at least one orientation, relative to its reference frame (Craig 1989).

Examples of motion rules are:

(1) $p \leftarrow p\_initial$
(2) $p \leftarrow p + p\_step$
(3) $x \leftarrow x + x\_step \cos\theta - y\_step \sin\theta$
(4) $y \leftarrow y + x\_step \sin\theta + y\_step \cos\theta$

Rule (1) specifies a constant value to a parameter p, e.g., for a fixed base. Rule (2) may specify the motion for a base with allowable translational motion, if $p \in \{x, y, z\}$, and of a ball joint with allowable rotational motion, if $p \in \{\psi, \phi, \theta\}$. If $p = \theta$, then rule (2) specifies the (stationary) rotational motion of a towmotor (see Figure 4). Rules (3) and (4) specify the translational motion of the towmotor in the XY-plane.

MOTION PLANNING

In the simulation, we are concerned with the role that a particular robot plays in the construction process and with its place in and its relation to the environment. That is, we are interested in the interactions of the robot with the site, the building under construction, the human labor crews, and other robots involved in the construction. This relation and, therefore, the robot's behavior, is constrained by physical obstacles, task-specified and other interactions, and safety and other considerations.

A *motion language* is developed that serves as a means to specify this behavior and, in particular, to specify the order in which the allowed motion steps should be proposed, for each robot or body. The choice of a specific motion step is dependent on the particular situation of the robot at the moment, according to the specified behavior. When a motion step has been chosen, all motion rules that contain this motion step are invoked, and the corresponding control parameters are updated.

Such a motion scheme fits easily within the framework of a local *path planning* method. In path planning we distinguish between *global* and *local* techniques. Global techniques are generally based on the concept of a configuration space (C-space) introduced by Lozano-Pérez and Wesley (1979). The disadvantages of this approach are due to the fact that C-space has high dimensionality for bodies with many DOF, and the technique is computationally expensive. It is also inefficient when dealing with a dynamically changing environment, as is the case for building construction. Local strategies may function both in task space and in configuration space. They are generally based on artificial potential functions (Khatib 1986), but exhibit a limitation that is due to the appearance of local minima.

In order to deal with a dynamically changing environment it is often useful to follow a local approach where obstacles are avoided as they are encountered (Lumelsky 1986). In this *motion planning* approach a path is built up from single motion steps in a bottom-up fashion. Such a technique may be either goal-driven or path-driven.

Lozano-Pérez and Wesley (1979) also present a simple collision avoidance algorithm that is based on the 'generate and test' paradigm. Adapted to motion planning, the algorithm takes the form:
(1) compute the volume swept out by the moving object for the proposed motion step;
(2) determine the intersection of the swept volume and the surrounding obstacles;
(3) if an intersection exists, propose a new motion step.

BEHAVIORAL LANGUAGE

The design of a general motion language for robot behavior is important, if one wants to express the appropriate behavior for a robot with respect to each of the constraints adopted, without restrictions on the types of robots that may be involved in the construction process. The language is used to specify the translation process of high-level pick-and-place tasks into specific, discrete, robot motions.

The basic building blocks of the motion language are the *motion command elements*. Motion commands may be grouped into series of motion steps using *control structure elements*, namely, *which*, *chain* and *each*. The *which*-structure groups different modules, each marked by a label, from which a selection is made depending on the particular situation. Each module specifies a series of actions, each of which is subject to verification of its validity in light of the surrounding obstacles. Series of motion commands are '*chain*ed' together, that is, for each series, commands are invoked only when necessary and, whenever possible, an attempt is made to try the previous command. The *each*-structure specifies a set of commands of which only one is active at any time. Only on failure is it replaced by the next command in the set.

The motion commands are *move_to* and *move_dir*. The former specifies a single step in the direction of a (sub)goal for the specified parameter. If the goal value equals the current value, no step is performed. The *move_dir* command explicitly specifies the direction of motion for a single step.

The language also allows for the use of variables. The language is completed with a set of functions that may be used within assignments or as command arguments:

goal(p)	returns the goal value of parameter *p*.
r_goal(p)	determines the goal value for parameter *p* relative to the current configuration.
dir(p, v)	determines the direction specified by *v* with respect to the current value of *p*.
axis(p, v)	reduces *v* to between -90° and 90° with respect to the axis defined by the angle *p*.
min(p) *max(p)*	returns the minimum (maximum) value specified for parameter *p*.
min(p, v) *max(p, v)*	returns the minimum (maximum) of the current value of *p* and *v*.
var(p)	returns the value of the variable associated with parameter *p*.

As an example, given any obstacle, the expected behavior of a crane should be to raise the load above the height of the obstacle, if possible, and to complete its trajectory subsequently, only lowering the load when the goal position has been reached. A corresponding motion plan for an automated crane hoist is as follows:

```
                                        move clear of the goal height
chain { move_to (z, max(z, 30 + goal(z))) };
                                        move closer to the base
chain { move_to (x, min(x, goal(x))) };
                                alter the orientation towards the goal
chain { move_to (θ, goal(θ)),
                                while staying clear of any obstacles
        move_dir (z, dir(z, max(z))) };
                                alter the displacement towards the goal
chain { move_to (x, goal(x)),
                                while staying clear of any obstacles
        move_dir (z, dir(z, max(z))) };
which {
    transport:                  when transporting an object
                                        stop clear of the goal
        chain { move_to (z, 30 + goal(z)) };
    default:                                              else
                        lower the end-effector to the goal height
        chain { move_to (z, goal(z)) }; };
```

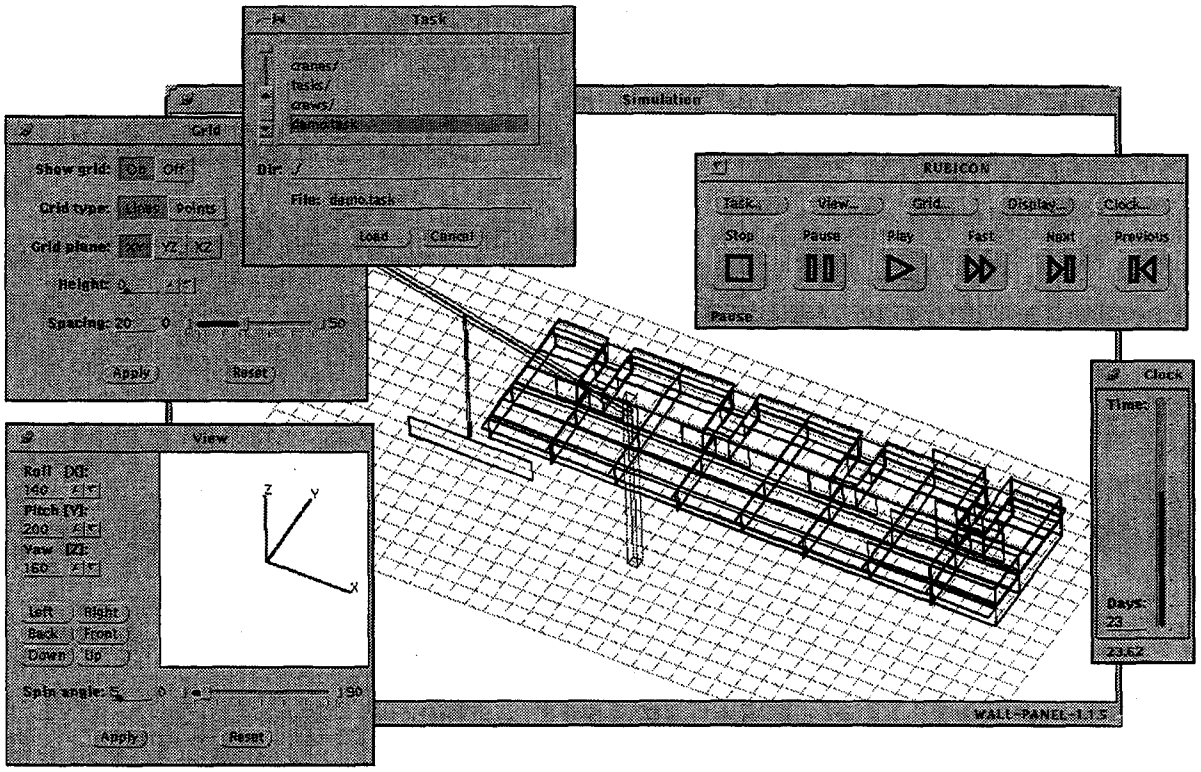

Figure 6: Snapshot of the RUBICON graphical user interface.

THE RUBICON PROGRAM

The findings of this research have been implemented in a simulation program, RUBICON. The program can be used to study different task plans, report on different robot types, study alternate robot-human crew mix examples, and produce measurements in terms of time or cost. Working examples include an automated crane hoist and an automated tow-motor active in the construction of a typical precast concrete residential building. The results of the simulation will assist an engineer or planner to decide on the appropriate mix of robots and human labor crews in the planning stage of a building construction project.

The input to the RUBICON program consists of two kinds of files. One file contains the task plan, that is, the description of the components and of the construction process as a sequence of tasks each of which is performed by a human or robot crew. For each robot agent specified in the task plan, a motion file is read in that contains a description of the motional capabilities of the robot and the motion rule set describing its intended behavior, expressed in the motion language specified above.

The output of the RUBICON program is a graphical simulation of the construction process as specified in the task plan, with a visualization of the motional actions of the robot agents and of the transportation of the construction components by these agents, and with a specification of the process time. The output is controlled through an audio/video-like control panel with buttons for *play*, *fast* play, *next*, *previous*, *pause* and *stop*. *Fast* play is achieved by unmaterializing the robot agent; *next* and *previous* instantly jump to the next, respectively previous, construction component.

Other interface panels allow the user to choose and load a new task plan and corresponding motion files, alter the 3D-viewing parameters, alter the displayed grid, and view the current process time (see Figure 6).

Figures 7 through 11 illustrate an examplar simulation of a single task: the placement of a wall-panel in the second unit of the first floor of the building under construction.

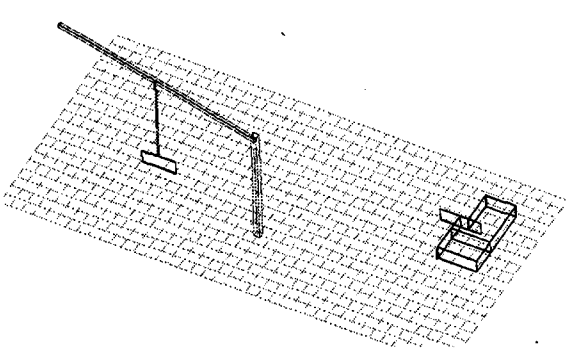

Figure 7: Snapshot of the simulation, step 1: The panel is picked up from the truck site or delivery location, after a human crew has taken care of connecting the panel to the crane hoist's tool.

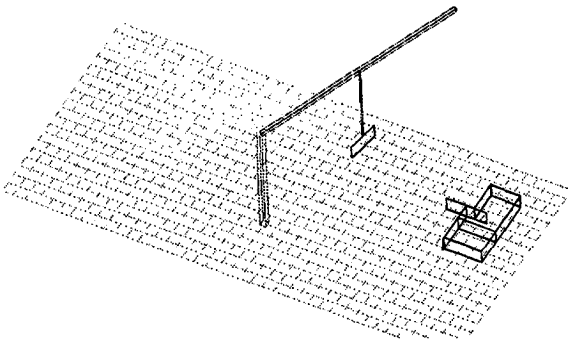

Figure 8: Snapshot of the simulation, step 2: The robot crane uses rotational motion (about its axis) to move the panel from the delivery location closer to its final position. Obstacles are avoided by raising the panel.

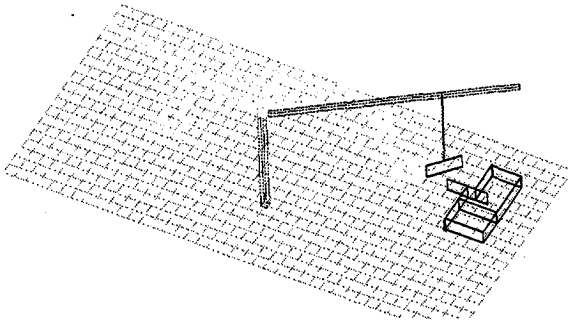

Figure 9: Snapshot of the simulation, step 3: The robot crane uses (radial) translational motion to move the panel above its final position. Obstacles are avoided by raising the panel.

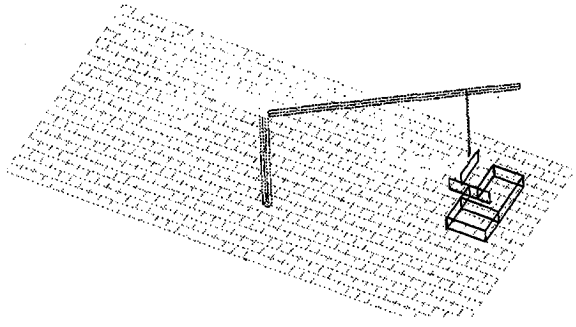

Figure 10: Snapshot of the simulation, step 4: While still attached to the robot's tool, the panel is rotated to its final orientation by a human crew.

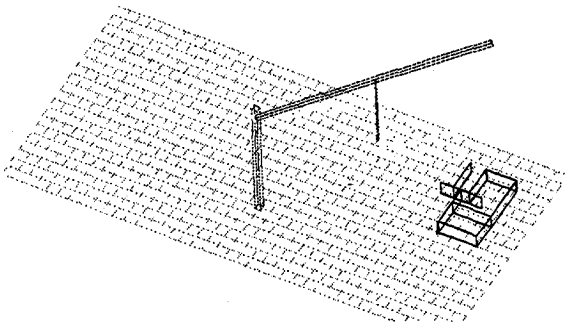

Figure 11: Snapshot of the simulation, step 5: Aided by the human crew, the panel is lowered to its placement location and disconnected from the robot's tool. The robot crane returns to the truck site for its next task.

CONCLUSION

Within the simulation described, our particular interest lies in finding a collision-free path for a robot from one location to another, given the dynamically changing environment of a construction site. For this purpose, we developed a framework for representing different robot types and for expressing their behavior.

Motion rules are used as a tool to embody the discretization of time in the simulation; they serve as a way to specify the motional capabilities of a manipulator in terms of a set of control parameters, the latter representing the degrees of freedom of the manipulator; and they supply a way to solve the discrepancy that may occur between specifying a robot's configuration on one hand, and controlling its motion on the other hand.

A motion language for robot behavior is presented that allows to describe disparate robot types each operating in a multitude of different situations. The language is used to specifiy robot motions for use in a general motion planning algorithm.

Two simple robot examples were used in the study: a crane and a towmotor. The simulation program, RUBICON, is demonstrated for a residential building example constructed with precast concrete panels.

RUBICON may serve as a tool for construction company engineers to perform studies on real project task plans; engineers/developers may use RUBICON to develop better task planners.

ACKNOWLEDGEMENTS

The development of the RUBICON prototype described in this paper was funded in part by the Japan Research Institute.

REFERENCES

Craig J.J. 1989. *Introduction to Robotics: Mechanics and Control* (2nd ed.). Addison-Wesley Publishing Company, Reading, MA.

Khatib, O. 1986. "Real-time obstacle avoidance for manipulations and mobile robots". *The International Journal of Robotics Research* 5: 90-98.

Lozano-Pérez, T. and M.A. Wesley. 1979. "An algorithm for planning collision-free paths among polyhedral obstacles." *Communications of the ACM* 22: 560-570.

Lumelsky, V.J. 1986. "Continuous motion planning in unknown environment for a 3D cartesian robot arm." In *IEEE International Conference on Robotics and Automation.* IEEE, Piscataway, N.J., 1050-1055.

Stouffs, R.; R. Krishnamurti; S.R. Lee and I.J. Oppenheim. 1993. "Construction process simulation with rule-based robot path planning." Abstract accepted for *International Symposium on Automation and Robotics in Construction.* CII, Austin, TX.

Zozaya-Gorostiza, C.; C. Hendrickson and D.R. Rehak. 1989. *Knowledge-Based Process Planning for Construction and Manufacturing.* Academic Press, San Diego, CA.