Using a Virtual World for Robot Planning

D. Paul Benjamin
John V. Monaco
Yixia Lin
Christopher Funk

Pace University, 1 Pace Plaza, New York, New York 10038, 212-346-1012
benjamin@pace.edu

Damian Lyons

Fordham University, 340 JMH, 441 E. Fordham Rd., Bronx, NY 10458, 718-817-4485
dlyons@fordham.edu

ABSTRACT

We are building a robot cognitive architecture that constructs a real-time virtual copy of itself and its environment, including people, and uses the model to process perceptual information and to plan its movements. This paper describes the structure of this architecture.

The software components of this architecture include PhysX for the virtual world, OpenCV and the Point Cloud Library for visual processing, and the Soar cognitive architecture that controls the perceptual processing and task planning. The RS (Robot Schemas) language is implemented in Soar, providing the ability to reason about concurrency and time. This Soar/RS component controls visual processing, deciding which objects and dynamics to render into PhysX, and the degree of detail required for the task.

As the robot runs, its virtual model diverges from physical reality, and errors grow. The Match-Mediated Difference component monitors these errors by comparing the visual data with corresponding data from virtual cameras, and notifies Soar/RS of significant differences, e.g. a new object that appears, or an object that changes direction unexpectedly.

Soar/RS can then run PhysX much faster than real-time and search among possible future world paths to plan the robot's actions. We report experimental results in indoor environments.

Keywords: robot schemas, virtual world, Soar cognitive architecture, port automata

1. INTRODUCTION

A truly cognitive architecture has not yet been implemented in robotics. Robots have been programmed to perform specific tasks such as mowing the lawn or navigating in the desert, and these accomplishments can be impressive, but robots still cannot act autonomously to choose tasks and devise ways to perform them. Even when performing their allotted tasks, they lack flexibility in reacting to unforeseen situations. Currently, the design of important perceptual and decision-making structures is done by the programmers before the robot begins its task. The semantics for the symbols and structures the robot uses is determined and fixed by these programmers. This leads to fragmented abilities and brittle performance. The robots cannot adapt their knowledge to the task, cannot solve tasks that are even slightly different from those they have been programmed to solve, cannot communicate effectively with humans about their goals and performance, and just don't seem to understand their environment. This is a principal stumbling block that prevents robots from achieving high levels of performance on complex tasks, especially tasks involving interaction with people.

The ADAPT project (**A**daptive **D**ynamics and **A**ctive **P**erception for **T**hought) is a collaboration of three university research groups at Pace University, Brigham Young University, and Fordham University to produce a robot cognitive architecture that integrates the structures designed by linguists and cognitive scientists with those developed by robotics researchers for real-time perception and control. ADAPT is under development on Pioneer robots in the Pace University Robotics Lab and the Fordham University Robotics Lab. Publications describing ADAPT are [1,2,3,4].

ADAPT models the world as a network of concurrent schemas, and models perception as a problem solving activity. Schemas are represented using the RS (Robot Schemas) language [8,9], and are activated by spreading activation. RS provides a powerful language for distributed control of concurrent processes. Also, the formal semantics of RS [10] provides the basis for the semantics of ADAPT's use of natural language. We have implemented the RS language in Soar, a mature cognitive architecture originally developed at Carnegie-Mellon University and used at a number of universities and companies. Soar's subgoaling and learning capabilities enable ADAPT to manage the complexity of its environment and to learn new schemas from experience.

2. THE ADAPT ARCHITECTURE

Our approach is fundamentally different from other projects, which typically attempt to build a comprehensive system by connecting modules for each different capability: learning, vision, natural language, etc. Instead, we are building a *complete cognitive robotic architecture* by merging RS [6,8,9], which provides a model for building and reasoning about sensory-motor schemas, with Soar [5], a cognitive architecture that is under development at a number of universities. RS possesses a sophisticated formal language for reasoning about networks of port automata and has been successfully applied to robot planning. Soar is a unified cognitive architecture [7] that has been successfully applied to a wide range of tasks.

Soar’s model of problem solving utilizes a single mechanism of subgoaling and chunking to explain human problem solving performance; utilizing Soar as the basis of ADAPT permits us to unify the mechanisms underlying perception, language and planning. Furthermore, it permits us to explore possible interrelationships between learning in these areas, e.g. how learning language and learning perception may be related. Finally, it permits us to test our architecture on robotic versions of well-known cognitive tasks and explore how robot learning might be related to human learning.

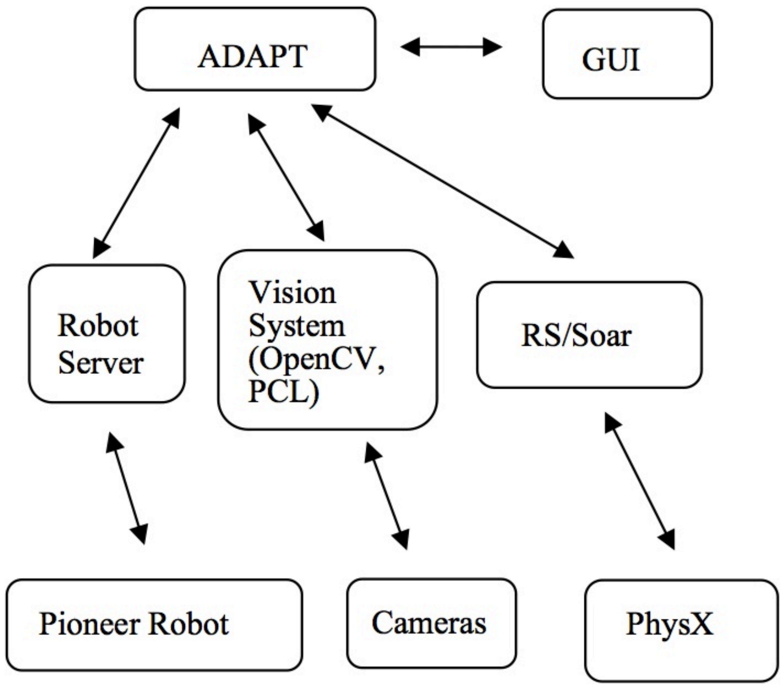

Figure 1. ADAPT architecture.

RS provides a powerful representational language for the system's dynamics, language and percepts; however, RS does not provide a mechanism for synthesizing the dynamics. Furthermore, RS lacks demonstrated cognitive plausibility, and in particular lacks a learning method.

We have implemented RS in Soar to take advantage of Soar's cognitively plausible problem-solving and learning mechanisms. Soar uses universal subgoaling to organize its problem solving process into a hierarchy of subgoals, and uses chunking to speed and generalize that process. Universal subgoaling permits Soar to bring all its knowledge to bear on each subgoal. Chunking stores generalized preconditions for search control decisions, so that in future tasks similar search control decisions are made in a single step.

3. Creating a Virtual World for Visualization

The central goal of our work is to develop effective methods for robots to comprehend their environment. Our approach models comprehension as a process of trying to recreate the observed dynamics by hypothesizing various sets of goals and beliefs for the agents, generating their dynamics based on these assumptions and comparing it with the observed dynamics. This knowledge-intensive approach to comprehension has a history within AI and in particular in machine learning.

We believe that the comprehension requires visualization. We view visualization as consisting of both a perceptual component and a reasoning component. The perceptual component is performed using the same perceptual mechanism that the robot uses to perceive its environment; the difference is that visualization perceives a simulation of the environment. Visual reasoning manipulates and superimposes representations that consist of a combination of symbolic knowledge and 3D animations.

This approach to comprehension requires the robot to be able to create different situations in which it can generate behaviors of robots, people and physical systems, and perceive the results of these behaviors. This requires implementing a virtual world that the robot can control.

ADAPT's virtual world is a multimedia simulation platform capable of realistic simulations of physical phenomena. It combines the various forms of map information found in most robots: topological, metric and conceptual information. ADAPT completely controls this virtual world, and can create arbitrary objects and behaviors in it, including nonexistent objects and behaviors that were not actually observed. Central to ADAPT's use of its virtual world is its ability to view these constructions from any point. This enables ADAPT to create visual representations with desired properties.

This approach to visualization is very different from previous work on reasoning about spatial relationships. ADAPT does not just turn spatial relationships into symbolic terms to be used in reasoning, but instead can reason visually about spatial relationships by constructing instances of those relationships, viewing them from various angles, and superimposing them.

In the current implementation, ADAPT's world model is PhysX. PhysX gives the robot the ability to create a detailed and dynamic virtual model of its environment, by providing sophisticated graphics and rendering capabilities together with a physics engine based on the PhysX physics engine. PhysX models a wide variety of dynamic environments, including modeling other agents moving and acting in those environments.

ADAPT uses this virtual world in a novel way. Typical robotics architectures connect their sensory mechanisms to their world models, so that sensory data is processed and modeled in the world model. The reasoning engine then operates on the world model to plan the robot's behaviors. This type of architecture treats perception as a separate process from the central reasoning, and typically the implementation reflects this, e.g. a computer vision module processes the vision data and puts symbolic representations of the recognized objects and their relationships in the world model, and the reasoning engine then manipulates these symbols to plan and learn. The reasoning engine does not process the sensory data.

In contrast, ADAPT's virtual world is not connected to its sensory processes. ADAPT's sensory data is placed directly in the reasoning engine (after some low-level processing); the reasoning engine's principal task in ADAPT is to reason about how to model the data. It does this in the following way:

It creates virtual entities and behaviors in PhysX.

It senses in the virtual world, using the same position and orientation as in the real world, and using the same sensors. For example, if ADAPT is modeling visual data, it grabs graphics input from PhysX, and if it is modeling sonar data, it grabs distance data from PhysX in the directions of the actual sonars.

It compares the virtual sensory data with the real sensory data, using a least-squares measure to find the degree of disagreement.

The reasoning engine searches alternative combinations of virtual entities and behaviors to attempt to minimize the measured disagreement. In this way, *perception becomes a problem-solving process*. This enables all the knowledge of the system to be brought to bear on perception, and unifies the reasoning and learning processes of problem solving with those of perception.

This search can be long and expensive; for this approach to comprehension to be practical, an effective speedup learning mechanism is required to store the results of this search. ADAPT contains a knowledge compilation method that stores generalized results of each successful search. One of the main research goals of our project is to quantify the effectiveness of this approach.

Visualization is also used in ADAPT for *predictive vision*: the robot predicts what it expects to see based on its virtual world and pays attention only to significant differences.

3. PREDICTIVE VISION

As an illustration of the use of this world model, let us consider ADAPT's novel approach to visual comprehension of its environment. ADAPT's vision system consists of two main components, a bottom-up component that is always on, and a top-down goal-directed component controlled by RS/Soar.

The bottom-up component is simple and fast. It does this by not producing much detail. The idea is for it to produce a basic stereo disparity map, a coarse-grained image flow, and color segmentation in real time. It runs on the robot's onboard computer using Intel's open vision library, and segments the visual data from the robot's two frame-grabbers. These "blobs" are transmitted together with stereo disparity data and optical flow to the off-board PC that is running RS/Soar, where it is placed into working memory. This component is always on, and its output is task-independent.

The top-down component executes the more expensive image processing functions, such as object recognition, sophisticated image flow analysis, and application of particular filters to the data. These functions are called in a task-dependent and goal-dependent manner by RS/Soar operators. This greatly reduces their frequency of application and speeds the operation of the vision system significantly.

These two components are not connected to each other; instead, the output of the bottom-up component is used by RS/Soar to determine when to call the top-down operations. RS/Soar compares the bottom-up output to the visual data predicted by the virtual world.

The virtual world can display the view that the virtual copy of the robot "sees" in the virtual environment. The output of this graphics "camera" in PhysX is segmented and sent to the MMD (Match-Mediated Difference), together with distance information and motion information. The MMD tests for significant differences between the expected view and the actual view, e.g. the appearance of a large new blob or a large change in optical flow. It aligns the real and virtual images with an affine map, then finds a set of matched key points and place a normalized Gaussian at each of them. The normalized match quality is the inverse of the distance between matched points divided by the sum of all match errors. We use this as a coefficient of the Gaussians to create the MMD measure. Any significant difference is placed in Soar's working memory, where it can cause an operator to be proposed to attend to this difference.

Soar controls all major aspects of perceptual processing, with the goal of constructing a virtual model of the environment that can be used in task planning. These aspects include focusing on regions of interest, choosing the depth of field, and deciding on the degree of detail for each part of the virtual model.

For example, if a new blob appears, an abstract operator will be proposed to focus on this blob and try to recognize it. If this operator is selected (if there is no more important operator to do at the moment) then RS/Soar will instruct the robot to turn its cameras towards this blob, focus at the appropriate depth, and obtain a point cloud from the visual input. Keypoints are extracted from the point cloud and used both in object recognition and to create a mesh that is registered with surrounding meshes.

Figure 2 shows two point clouds of a kitchen counter by this vision system. These are transformed into meshes. By joining small meshes together, the system can create larger meshes when necessary. Portions of the world that are not relevant to task goals are not rendered, greatly increasing efficiency.

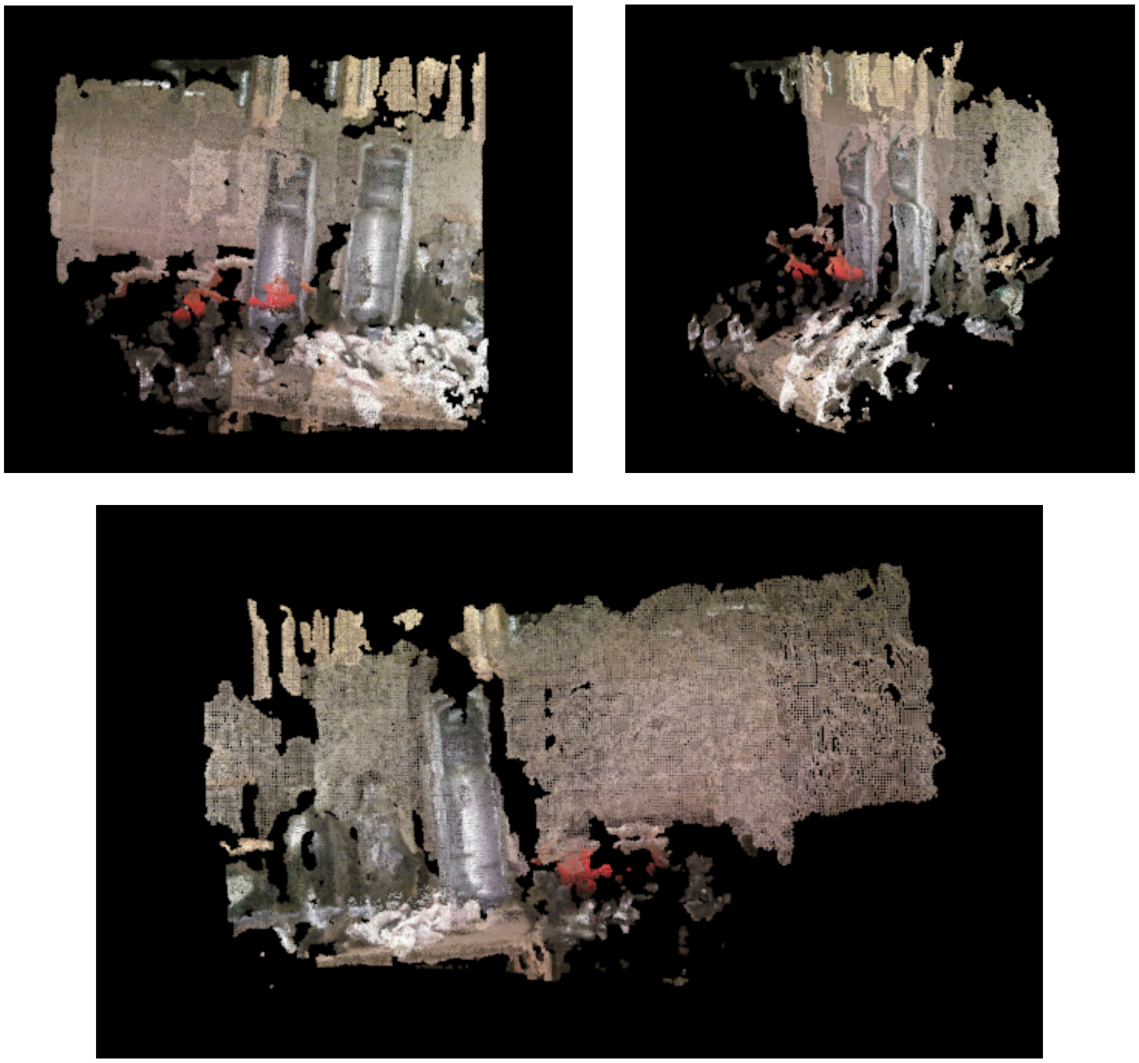

Figure 2. The two point clouds at top are registered and joined to create the bottom cloud, which is transformed into a PhysX mesh.

This approach also works for people, as shown in Figure 3. People are skeletonized in a manner similar to that used by the Kinect, and the skeleton is covered with mesh. Our initial work was with the Kinect, but now uses only stereo vision.

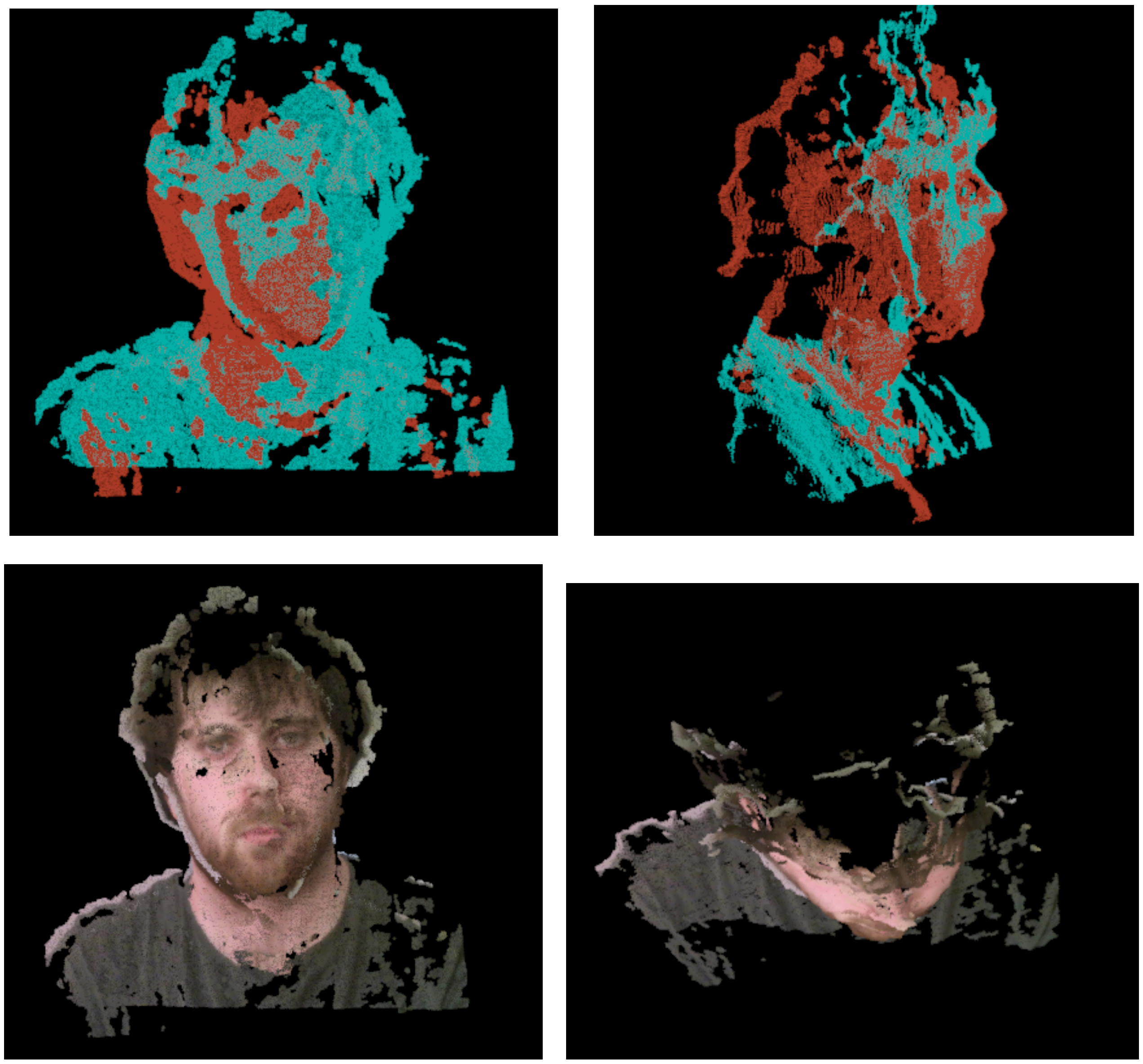
Figure 3. Point clouds for a face, registered and joined to create a 3D mesh for a skeleton. The two point clouds at top are joined to create the cloud seen below, frontal view at left and from above at right.

Once an object is recognized or rendered, a virtual copy is created in PhysX. The object does not need to be recognized again; as long as the blobs from the object approximately match the expected blobs from PhysX, ADAPT assumes it is the same object. Recognition becomes an explicitly goal-directed process that is much cheaper than continually recognizing everything in the environment. The frequency with which these expensive operations are called is reduced, and they are called on small regions in the visual field rather than on the whole visual field.

Thus, ADAPT's vision system spends most of its time verifying hypotheses about its environment, instead of creating them. The percentage of its time that it must spend attending to environmental changes depends on the dynamic nature of the environment; in a relatively static environment (or one that the robot knows well from experience) there are very few unexpected visual events to be processed, so visual processing operators occupy very little of the robot's time.

Figure 4 shows the overall flow of control of our vision system [11,12]. The real and synthetic images of the scene as viewed by the robot are compared. If the scenes are considered the same but from different viewpoints, then the viewpoint of the camera in the simulation is changed, and the simulation generates an image taken by the camera at the new location. If an unexpected object is seen in the real image, an object is introduced at the corresponding position in the simulated scene. The region of the real image responsible for the difference is used as video texture on the object and a new synthetic image generated. The information on whether there is no difference, an unexpected object, or an object missing between the image pairs is made available to action planning.

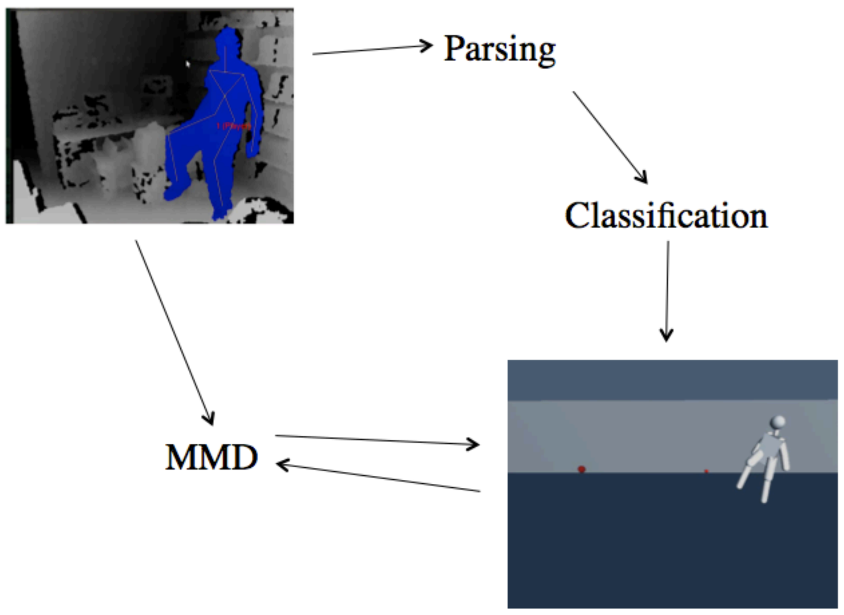

Figure 4. The input from the physical world (left above) is parsed, classified and rendered into the virtual world (below right). The MMD compares both inputs, detects differences and updates the virtual world.

This loop of difference detection and simulation modification is used to keep the simulation synchronized to the observed environment. For prediction purposes, the simulation can be allowed to 'fast forward' in time, so that the expected position, for example, of a target can be calculated and then compared to observations.

There is no knowledge acquisition bottleneck. We don't need to learn the physics rules. In addition, these rules are general, eliminating brittleness.

Perceived motion sequences are cut into basic units by finding points at which the preconditions of the sequence are identical with the postconditions. An example is shown in Figure 5. PhysX is used to simulate sequences in previous states to evaluate these preconditions and postconditions. Motion sequences are normalized and sampled to create a fixed length feature vector of positions and velocities. Then they are clustered using a hierarchical clustering algorithm.

Each type of motion sequence is then coded in RS.

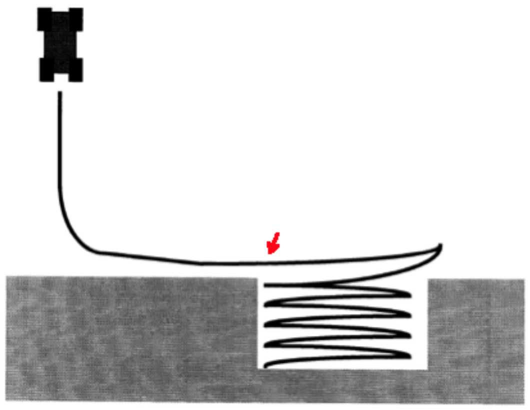

Figure 5. The car parking in an available spot follows a path that can be split at the arrow. The path after the arrow is executable in the initial state, but not anywhere afterwards until the red dot. This splits the car's path into two pieces.

4. SUMMARY

We have sketched the overall design of a new approach to the comprehension of dynamics that is part of a robotic cognitive architecture. A powerful 3D multimedia world model is used to render dynamics. This gives the robot the ability to visualize alternative evolutions of the dynamics and to classify them. The implementation of the basic components is complete. We have begun experiments on to evaluate the architecture.

Further information on this work, including video clips showing the robot moving under the control of schemas and the use of the world model, can be downloaded from the website for the Pace University Robotics Lab: http://csis.pace.edu/robotlab

REFERENCES

[1] Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, “Embodying a Cognitive Model in a Mobile Robot”, Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision, Boston, October, 2006.

[2] Benjamin, D. Paul, Damian Lyons and Thomas Achtemichuk, “Obstacle Avoidance using Predictive Vision based on a Dynamic 3D World Model”, Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision, Boston, October, 2006.

[3] Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, "Designing a Robot Cognitive Architecture with Concurrency and Active Perception", Proceedings of the AAAI Fall Symposium on the Intersection of Cognitive Science and Robotics, Washington, D.C., October, 2004.

[4] Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, " Cognitive Robots: Integrating Perception, Action and Problem Solving in Behavior-Based Robots", AAMAS-2004 Proceedings, pp. 1308-1309, 2004.

[5] Laird, J.E., Newell, A. and Rosenbloom, P.S., “Soar: An Architecture for General Intelligence”, *Artificial Intelligence* **33**, pp.1-64, 1987.

[6] Lyons, D.M. and Hendriks, A., “Exploiting Patterns of Interaction to Select Reactions”, Special Issue on Computational Theories of Interaction, Artificial Intelligence **73**, 1995, pp.117-148.

[7] Newell, Allen, *Unified Theories of Cognition*, Harvard University Press, Cambridge, Massachusetts, 1990.

[8] Lyons, D.M., “Representing and Analysing Action Plans as Networks of Concurrent Processes”, IEEE Transactions on Robotics and Automation, June 1993.

[9] Lyons, D.M. and Arbib, M.A., “A Formal Model of Computation for Sensory-based Robotics”, IEEE Transactions on Robotics and Automation **5**(3), Jun. 1989.

[10] Martha Steenstrup, Michael A. Arbib, Ernest G. Manes, *Port Automata and the Algebra of Concurrent Processes*. JCSS 27(1): 29-50, 1983.

[11] "Integrating Perception and Problem Solving to Predict Complex Object Behaviors", by Damian M. Lyons, Mohamed Chaudhry, D. Paul Benjamin, Marius Agica, John Vincent Monaco, Conference on Multisensor, Multisource Information Fusion, SPIE, April 2010.

[12] "Robot Video Tracking by Comparing Real and Simulated Video Scenes", by Damian M. Lyons and D. Paul Benjamin, Conference on Intelligent Robots and Computer Vision, SPIE, San Jose, Calif., January 2009.