MOTOR KNOWLEDGE REPRESENTATION

Giuseppe Marino Pietro Morasso Renato Zaccaria

Department of Communication, Computer and Systems Science, University of Genoa - Italy

ABSTRACT

The motor control problem is considered in the framework of knowledge representation. In the AI/Robotic world, a formal model for Motor knowledge should fill a gap between task planning and low level robot languages; such model should be able to "virtualize" the robot and the interaction with the environment so that the planner could produce (and rely on) high level abstract actions, characterized by high autonomy and skill. The paper discusses some general aspects about actions, actors, and scenes, and describes the NEM language, which is able to represent and animate humanoids in a scene and is meant to provide a software laboratory for experimenting with action schemas.

1. Introduction

In the AI/Robotic world, a formal model for motoric" knowledge (Marino et al., 1984) should fill a gap between task planning (Lozano Perez, 1982) and low level robot languages (Bonner and Shin, 1982); such a model should be able to "virtualize" the robot and the interaction with the environment so that the planner could produce (and rely on) high level abstract actions, characterized by high autonomy and skill. This problem is strongly related to that of modeling and simulating the human body and its movements (Badler and Smoliar, 1979). This paper briefly reviews a formal language, called NEM, for the representation and animation of "moving entities", or actors (e.g. humanoids), in a scene. Geometric, kinematic, dynamic aspects are all tightly interrelated for any skill. In general, however, it is convenient to distinguish, for an actor, two different levels of "knowledge" of the particular aspect of a skill: i) a qualitative/symbolic/explicit level, and ii) an implicit/analogic/quantitative one; NEM is intended to support the second level.

An action is a change in the relation between an ector and its surrounding. If we now associate with each feature of the scene a local system of coordinates (a frame (*)), the action can then be viewed as a stream of variations of some of the mutual relations between local coordinate systems, due to a stream of motor commands.

The whole scene can thus be described as a "forest" of frames, linked to features, grouped together conveniently when referring to a common <u>structure</u> (an actor, an object, a family or a part thereof). Ensembles of frames can only represent, in a direct way, the "skeletric" structure of actors or objects; if smoother and more detailed representations are needed, it would be necessary to associate appropriate "shape formation" attributes with frames and frame ensembles. This points out the problem of interfacing an action oriented system with solid modeling concepts (Binford, 1982) and with techniques of path planning and obstacle avoidance (Lozano Perez, 1982).

2. NEM: a language for representing actors, scenes, actions

The NEM language has been designed to provide a procedural, non-hierarchical representation of motoric knowledge. NEM is intended: i) to define and model (potentially) moving entities ("objects" or "actors"); ii) to describe the movement of an entity as a whole, or of groups or parts of entities.

Object/Actor representation is based on an atomic element called Geometric Frame (GF), by which Geometric Frame Structures (GFS) can be built. GFSs provide object/actor modeling in terms of skeletons of articulated chains, whose basic element is a tree of GFs. NEM objects are general and hierarchical: "limb", "man", "quadruped", "table", "crowd" are legal parameterized (families of) entities.

Moving objects are called Actors. An Actor's movement is described by means of scripts called Motions. Motions are abstract, non-hierarchical procedural descriptions of movement for single

(*) This kind of "frames" has nothing to do with Minski's: they come from analytic geometry.

objects, parts or groups of them, or generalizations of objects (families). A motion can adapt itself to a particular object in a family: for example, a script "animal^walk" could fit, at some level of detail, different actors like a "spider" or a "dog". Motions are built by composition, or specialization of other motions. Motions can express, at a certain level, geometric reasoning, both static ("is the book over the chair ?") and dynamic ("may John reach the book without walking ?"). Finally, motions allow expression of common sense motoric knowledge (that we may call "common skill knowledge"): naive physics concepts (such as "gravity", "equilibrium", "collision", "pushing" and so on) can be easily defined in terms of general virtual motions.

NEM has three components: i) an algebra, called Frame Algebra Notation (FAN), which manages GFSs; ii) atomic motor primitives and their semantics ("Primitives"); iii) rules of superimposition or composition of motions and primitives, called Constructs. They are the operators with which we can build motions.

2.1. Frames and Motions

The atomic datum in NEM is the frame, which corresponds to an orthogonal system of reference and is represented by means of a 4 x 4 homogeneous matrix. Homogeneous matrices express the translation/rotation of the given frame with respect to an ancestor frame.

Frames are used to identify significant points of an object/actor. For example, a simple pyramid can be defined by the following GFS:

@ ENV FRAME PYRAMID
@ PYRAMID FRAME VERTEX_1 VERTEX_2 VERTEX_3

where ENV is the "universe" frame (the environment) and "©" is an operator which refers a frame to its ancestor. The definition of a GFS and its corresponding initialization can be embedded into a parameterized definition block and instanciated several times.

At the lowest level, FAN semantics guarantees the computability of all spatial relations. However, FAN provides functions at different higher levels: GFSs can be dynamically generated and destroyed; frames can be assembled/disassembled into their components; functions can be defined and so on. For example, the script POS(ADAM"MOUTH @ EVE'R'HAND) gives the geometric relation existing at a certain instant between Adam's mouth and Eve's right hand. Finally, a frame can be referenced by its position inside a GFS (its "pathname") rather than by its name. Pathnames and related functions (such as the IS_DEF(<pathname> function), which tests the existence of a GFS element, allow motion scripts to fit families of similar structures.

A motion is a collection of three components: i) frames and declarations of variables, ii) motor primitives and iii) instances of other motions. Any component may be missing: for example, a motion can simply embed an object definition. Motions are usually active concurrently; frames and variables binding is dynamic.

The atomic element for motion construction is the primitive motor operator (PMO). PMOs are inspired by anthropomorphic mechanics and are defined at two levels: i) joint level, and ii) limb level. The PMOs of the former type move a frame, whereas the PMOs of the latter type affect chains of frames by specifying the motion of the "end effector" (they solve the inverse kinematic problem (Benati et al., 1982)). More abstract operators are also defined: for example, temporary linking an end effector with some moving frame (passive motions), or "reversing" a chain ("move the hip with respect to the foot").

Composite Motions are defined by composing PMOs and/or composite motions already defined. Compositions can be made in different ways:

sequential execution:

ACTION_l ; ACTION_2 j ACTION_3 ; ...

(where <ACTION> stands for a PMO or a motion;

parallel execution:

ACTION_l & ACTION_2 & ACTION_3 6 ...

guarded execution:

WHILE EVENT DO ACTION
WAIT EVENT DO ACTION
PERFORM ACTION_l EXCEPT EVENT THEN ACTION_2

The composition paradigm allows to overlap in time different spatio-temporal units, therefore providing an unlimited capability of trajectory formation: this expresses naturally a non-hierarchical, distributed approach to motor control (Hinton, 1984) which is conceptually akin to the object-oriented programming style (Weinreb and Moon, 1980). Guarded executions allow actors to synchronize through events, or to communicate each other through message passing.

3. Results

The NEM project is being implemented in the Unix environment. The NEM interpreter is written in C to obtain high efficiency. Preliminary versions of two interfaces are available: one interfaces NEM with FADL2 (a geometric modeling system) and the other with Prolog.

Figure 1 shows a graphic trace of the performance of a NEM script which represents a diving humanoid. It is worth noting that in this motor paradigm the motor actions are concurrent with the action of gravity. In the NEM script, two corresponding concurrent motions are activated; this is an example of physical laws implicitly embedded into the motoric knowledge. Moreover, "environmental" processes can be tested by "control" actors in order to tune action parameters.

Several other paradigms are being experimented (sitting, walking, picking ...) with the purpose of building a high level motor data base. With regard to the NEM-PADL2 interface, fig. 2 shows a NEM humanoid (in a "discobolus" posture) "dressed" with a PADL2 articulated solid model. Better schemes of human body representation are available (see Special Issue of IEEE Comp. Graph, and Appl.,vol.2 no.9,1982), but most of them are not articulated.

4. Final remarks

We stressed the potential capability of NEM to build high level, abstract pieces of motor activity, in spite of the simplicity of the GFSs, which are a low level atomic piece of knowledge. Moreover, NEM's facilities to express both static and dynamic geometric reasoning suggest that it could play a role of "co-planner", as to say, it could integrate both functions of analog modeling

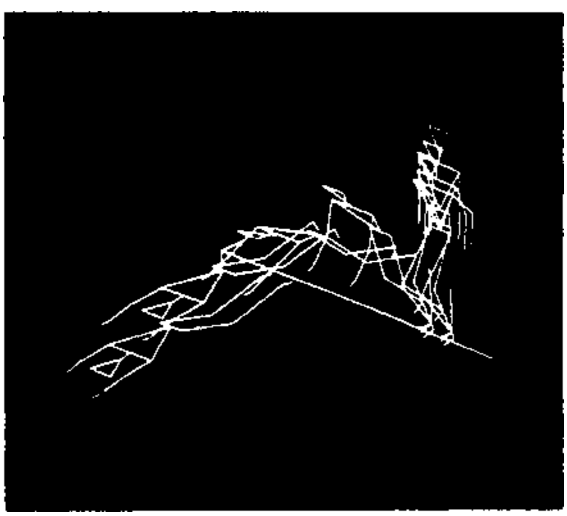

fig. 1

fig. 2

the (moving) world and of storing tasks/actions data base.

REFERENCES

BADLER NI, SMOLIAR SW (1979) Digital representation of human movement, ACM Comp. Surv. 11:1

BENATT M, GAGLIO S, MORASSO P, TAGLIASCO V, ZACCARIA R (1980) Anthropomorphic robotics. Biol. Cybern. 38:125-150

BONNER S, SHIN KG (1982) A comparative study of Robot Languages, IEEE Computer, 15:12

BINFORD TO (1982) Survey of model based image analysis systems. Intern. J. Robotic Research 1:18-64

HINTON G (1984) Some computational solutions to Bernstein problems. In WHITING BTA (Editor) Human motor action - Bernstein reassessed. Elsevier Science Publ. BV (North Holland): 373-412

LOZANO PEREZ T (1982) Task planning. In BRADY M et al (editors) Robot motion: planning and control. MIT Press, Cambridge, MA

MARINO G, MORASSO P, TROIANO E, ZACCARIA R (1984) Representing Motor Knowledge, Proc. AIMSA84 Conf., Varna, Bulgaria

WEINREB D, MOON D (1980) Flavors: Message Passing in the LISP Machine. Artificial Intelligence Laboratory Technical Report 602, MIT press, Cambridge, MA