A Multi-Agent Analogical Representation for Physical Objects[*]

Luca Maria Gambardella and Marc Haex

IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Corso Elvezia 36 - CH - 6900 Lugano
Phone: +41 91 22 88 81 Fax: +41 91 22 89 94
Email: luca@idsia.uu.ch marc@idsia.uu.ch

Abstract

The topic of this paper is a representation model for solid objects used for physical simulation purposes and for planning in a robot assembly system. The system combines analogical representation and multi-agent modelling, using a bottom-up representation for objects based on analogical agents. We call these agents analogical because they are mapped, they interact and they reason directly on the workspace representation, which is a discrete grid. The agents contain local geometrical and physical constraints and they cooperate to satisfy them while moving in the direction of an external force and interacting with other objects in the workspace. An emergent functionality of the simulation of a block moving in a complex environment is the solution of the stability problem.

1 Introduction

Simulation is a frequently used technique in many fields. In planning systems it is a valuable method to check whether the execution of the planned actions will lead to a successful state without having to try them in the real environment. This paper deals with the simulation of physical objects, i.e. objects moving according to some force and colliding with other objects in the environment. Consider for instance an object falling on the edge of a table: it will hit the table, rotate around some pivot touch point and fall further down until it hits another object or the ground. In this paper we will present a system modeling this kind of behaviour in 2 dimensions with polygonic solid shapes. We will describe a bottom-up representation for these objects, consisting of autonomous agents, and show how the global behaviour emerges from the interaction of these agents. We will call these agents analogical because they are mapped, they interact and they reason directly on the workspace representation, which is a discrete grid.

This kind of analogical simulation has previously been applied to other physical systems like liquids [DKS91] [GM89] and strings [GGM89]. The use of analogical representation is fostered by the nature of the problem. Simulating complex physical systems using exclusively symbolic information would result in low accuracy when detailed spatial knowledge is needed.

A key example of an analogical simulation program is WHISPER, described in [Fun80]. It is able to detect and simulate instabilities in a blocks world using diagram representations. WHISPER has similar functionalities to those of our system

[*]This research is supported by Swiss National Project NFP23 "Automatic Assembly based on Artificial Intelligence". The project is a collaborative effort of Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (Lugano), Institut de Microtechnique Ecole Polytechnique Federale de Lausanne and Institute de Mathematiques et Informatique Universite de Neuchâtel.

but uses centralized high-level reasoning on low-level distributed analogical representation to create the envisionment of object configurations. Our system tries to avoid this global centralized control by distributing among the constituent agents the necessary local behaviour to obtain a correct global result. We will obtain the same result, not by explicitly describing the movement of an object; the required functionality emerges from internal communications and interaction with the environment. The importance of the use of analogical representations in autonomous agent organisations is explained in [Ste89].

The next chapter describes the different kinds of agents that our representation consists of. The third one explains how agents cooperate to obtain the desired behaviour. This is followed by a discussion of the main characteristics, the limits and the possible applications of the analogical agents approach.

2 Agent Architecture

Before explaining the different kinds of agents, it is important to notice that the underlying workspace is a discrete grid; this means a two dimensional array of cells. Each object occupies a number of cells according to its size and shape. Objects are considered to be two-dimensional, solid, non-elastic polygons. The cells making up the contour of the polygon contain two kinds of agents, namely node agents on the nodes of the polygon and contour agents on the edges between the nodes. These two kinds of agents are organised in a two-level hierarchy in which the contour agents are subordinated to the node agents in the sense that they serve as analogical sensors and their position depends on the geometrical information contained in the node agents. Node agents contain the necessary local rules that determine the geometrical shape and the non-elasticity property of the objects. For reasons of efficiency, the cells that fill up the polygon remain empty. A third kind of agent which is not a real part of an object, but which is physically attached to it is the force agent. It represents the qualitative force acting on the object and will play an important role in the simulation behaviour of the agents. By a qualitative force, it is meant that it only gives an indication of the direction of the force. Notice that the granularity of the workspace grid influences the accuracy of our model, because it determines the unit of the size that an agent can move. The next paragraphs will describe these agents, giving their knowledge and positional constraints.

Figure 1: A node agent and its geometrical information.

2.1 Node Agents

Node agents are positioned on the angular points of the polygonal object. The necessary geometrical information about the object is distributed over these agents. Each node agent n contains the following information slots (see Figure 1):

- A position (x, y) denoting a cell in the workspace grid.
- Links to the two neighbouring node agents: n_{b1} and n_{b2}.
- The distances d_1 and d_2 from n to resp. n_{b1} and n_{b2}.
- The angle α between the two neighbouring node agents.
- The orientation ϕ of the latter angle. This is a variable.

Figure 2: A contour agent and its geometrical information.

- A link to the two direct contour agents of n, as described in the next paragraph.

d_1, d_2 and α are set at creation time and remain constant all the time. They represent the constraints for a node agent and they are used to describe the geometrical shape of the object. The constant angle α expresses the non-elasticity constraint for the object and d_1, d_2 tell that the object is not extensible. The angle ϕ describes the orientation of the fixed angle in the workspace. At each time step the angle and the distances between n and its two neighbour agents have to be equal to α and to d_1 and d_2. Each node agent checks its constraint by asking the two neighbour node agents for their position and from computing the actual distances d'_1, d'_2 and angle α'. When these constraints are not satisfied the agent can correct them by changing its position or by asking n_{b1} or n_{b2} to adapt their position in order to obtain the correct value for d'_1, d'_2 and angle α'. Another constraint is that each cell can only contain a single agent (i.e. a node or contour agent), which implies that the node agent will always check a cell before it tries to occupy it.

2.2 Contour Agents

Contour agents are positioned on the edges between the node agents. Each contour agent c contains the following information slots (see Figure 2) :

- A position (x, y) denoting a cell in the workspace grid.

Figure 3: A force agent and its geometrical information.

- A link to the two neighbouring contour agents c_{n1} and c_{n2}.
- A link to the two node agents at the ends of the edge on which c is positioned.
- A distance d to one of the two node agents mentioned in the previous slot.

The constant d represents a positional constraint for the agent. Each time the object moves, the contour agents have to recompute their new position using the constant distant d and the links to the node agents. When it tries to change position it will make sure that it does not occupy a cell already occupied by another object. The purpose of contour agents is to act as analogical sensors, i.e. to detect contact between the object and other objects represented as filled cells in the workspace.

2.3 Force Agent

The force acting on an object is represented by a force agent, which is positioned at the cell containing the point of application of a force. For instance, for an object falling under gravity this will be the center of mass. The knowledge of a force agent f is contained in the following slots (see Figure 3):

- A position (x, y) denoting a cell in the workspace grid.

- A qualitative force vector $\vec{f}$ denoting the direction of the force.
- A constant angle ϕ and distance d denoting the relative position of the force agent to the object.

The force agent's positional constraint is relative to the node agents and is contained in the constants ϕ and d. After each timestep the actual values for this angle and distance are computed and if necessary the agent's position is corrected. The force vector can be either static or could be changed by an external controller. Also, in order to consider multiple moving objects, a force propagation protocol between colliding obstacles could be considered.

2.4 The communication protocol

In order to satisfy their constraints and their goal, which we will describe in the next chapter, communication between the agents will be necessary. Between node agents and contour agents this occurs according to an actor protocol, which means that agents can only communicate directly with those agents that they know about, and that communication is done by message passing. These communication abilities are called links in the previous paragraphs. Notice that between the node agents a doubly linked circular list exists, as for the contour agents. Node agents have links with their direct neighbouring contour agents, which implies that by message passing each node or contour agent can reach every other node or contour agent. Contour agents can use a shortcut link to the node agents of their edge, for performance reasons. The force agent can communicate with every other agent and vice versa.

3 Agent Behaviour

In this chapter we will describe the behaviour of the agents to obtain global movement of an object according to a force applied to it. We will consider one object moving in the workspace containing static obstacles. The point in which the force is applied is considered to be the center of mass of the object. Forces applied at points other than the center of mass result in a torque and rotational movement, even in free space.

3.1 Unconstrained Translation

When a force is applied to an object a force agent is created and attached to it. Now, the goal of the node agents is to try to move in the direction of the force. The node agents will try to occupy the neighbouring cell in the workspace grid according to the required direction, consequently the contour agents will change their position in order to remain on the correct edge position. If every agent successfully changes position, i.e., does not try to occupy a filled cell, the result will be a one cell translation of the object. The positional constraints of the node agents will remain satisfied and no complicated communication will be needed to resatisfy them and the force agent will start a new movement instruction.

3.2 Stability Problem

When an agent tries to occupy a cell that is already occupied by another object, it will not change position. If this happens for one or more agents the agent configuration will not conform to the object initially modeled. At this point the agents will have to negotiate to satisfy their local constraints and to obtain a new configuration conforming to the initial geometry and the movement caused by the force. Because we deal with only one moving block in a workspace, containing only static obstacle blocks, we can distinguish two situations. The first is that the moving object is completely blocked by the others. The second possibility is that the object is only partially blocked and it will start rotating around a pivot touch point, which is the touch point closest to the center of mass. This is the so called stability problem. We will show how we obtain this global behaviour, without really having the high-level notion of stability or rotation around a pivot point, but by cooperation among the agents in the bottom-up representation. The general idea is that two agents will be selected, one on each side of the force vector (see Figure 4 and 5), from which the reconstruction protocol will be initiated. These two agents are called fixed and are found in the following way. First each contour agent which was unsuccessful in trying to occupying its desired cell, looks to se whether both its neighbours are in the same condition. If this is not the case the agent knows that it is the last one in a series of touching agents (and thus candidate for a pivot point). In Figures 4.a and 5.a these are the contour agents which are marked black. These candidate pivot points communicate with the force agent and compute their distance from the force vector. For each side of the force vector the agent having the shortest distance is marked as fixed. In Figures 4 and 5 this leads to the marking of the contour agent which is labeled f_1. If this results in two fixed agents a reconstruction protocol is started. In the case of only one fixed agent the nearest node agent at the other side of the force vector is selected to be the second fixed agent (Node agent f_2 in Figure 4). These two agents determine how the positions of the rest of the agents are corrected. This is done in the following way: the fixed agents communicate to their neighbours to change their position in a way to satisfy the constraints. These in turn do the same with their neighbours. When the loop is closed all constraints are satisfied and a new movement can be started. Notice that the first case of finding two fixed contour agents, each on one side of the force vector, agrees with a stable situation while the second case means an unstable situation. Examples of resulting configurations after this correction are shown in Figures 4 .b and 5 .b.

4 Characteristics, Applications and Limits of the System

The main difference between our multi-agent approach and high-level approaches, like the WHISPER system by Funt [Fun80] , is that we do not explicitly code the global physical behaviour, but it emerges from the interaction between agents.

Figure 4: An unstable situation. 4.a Shows an agent configuration after a collision with a static obstacle. Agents f_1 is first chosen to be the fixed agents, Node agent f_2 is chosen to be the second fixed agent. 4.b shows the resulting configuration after the reconstruction.

Figure 5: A stable situation. 4.a Shows an agent configuration after a collision with a static obstacle. Agents f_1 and f_2 are chosen to be the fixed agents. 4.b shows the resulting configuration after the reconstruction.

In WHISPER a high level reasoner first checks a blocks configuration for instabilities, choses a pivot point and simulates the rotation explicitly. Our system does not have an idea of instability or rotation, agents always apply the same behaviour of moving in a certain direction and if necessary recovering from an abnormal situation. In order to distinguish between a stable or unstable configuration the global behaviour must be interpreted externally by looking at the workspace grid or internally by monitoring the behaviour of individual agents. For instance stability, it is sufficient to observe the movement of the agents. This explains the need for an interpreter module to extract information relating to the status of a simulation system which uses distributed representation and control. In [Gam91] the use of an analogical string simulation in an automatic assembly system applying a planning, simulating and interpreting loop is described.

A possible application of this physical block simulation in robot assembly could be the simulation of an object following a path, defined by the planner and represented as a series of forces in the workspace. The system could monitor the object, see where it touches and evaluate the result as either successful or not and use this as a basis for recovery planning or execution of the plan in the real environment.

For this kind of application the current functionality will be sufficient. For other applications it will be necessary to cover more complex functionalities like velocity, acceleration object surface properties and other dynamic features of a physical object. To be able to take these features into account for simulation the current qualitative knowledge of the agents needs to be extended with a more quantitative one.

5 Conclusions

We have described a model of physical objects for simulation purposes which relied on autonomous agents, bottom-up descriptions, constraint satisfaction and the use of local and analogical information. The result of the simulation of a moving block in a complex environment implicitly solves the stability problem, not by global reasoning, but by cooperation between autonomous agents. Analogical representations are used to represent both the workspace and the changes that are made in it. We mixed this strength of analogical representations with the power of autonomous agent systems which lies in the capabilities of agents to cooperate and communicate to satisfy their local constraints. An implementation of the model serves as a basis for further research. One goal is to expand the model to be used with multiple moving objects and propagation of forces. Another goal is to use the model in robot assembly, in which we simulate a grabbed object moving in its workspace.

References

[DKS91] Jo Decuyper, Didier Keymeulen, and Luc Steels. A qualitative model for the behavior of liquids in daily-life circumstances. In *First European Workshop on Qualitative Reasoning About Physical Systems*, Genova, Italy, January 1991.

[Fun80] Brian V. Funt. Problem-solving with diagrammatic representations. *Artificial Intelligence*, 13(3):201–230, 1980.

[Gam91] Luca Maria Gambardella. Simulation and planning with multiple representations. In *AI, Simulation and Planning in High Autonomy Systems*, Cocoa Beach, Florida, April 1991.

[GGM89] Luca Maria Gambardella, Francesco Gardin, and Bernard Meltzer. Analogical representation of naive physics. In *Second Workshop on Qualitative Physics*, Paris, France, 1989.

[GM89] Francesco Gardin and Bernard Meltzer. Analogical representations of naive physics. *Artificial Intelligence*, March 1989.

[Ste89] Luc Steels. Cooperation between distributed agents through self - organisation. In Yves Demazeau and J.P.-Műller, editors, *Decentralized A.I., Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Cambridge, England, August 1989.