Introductory Samba

Jukka Riekki

The Samba architecture is targeted at controlling a mobile agent in a dynamic environment. It contains the following types of modules: Sensors, Actuators, Markers, Behaviors, and Arbiters (see Fig. 1).

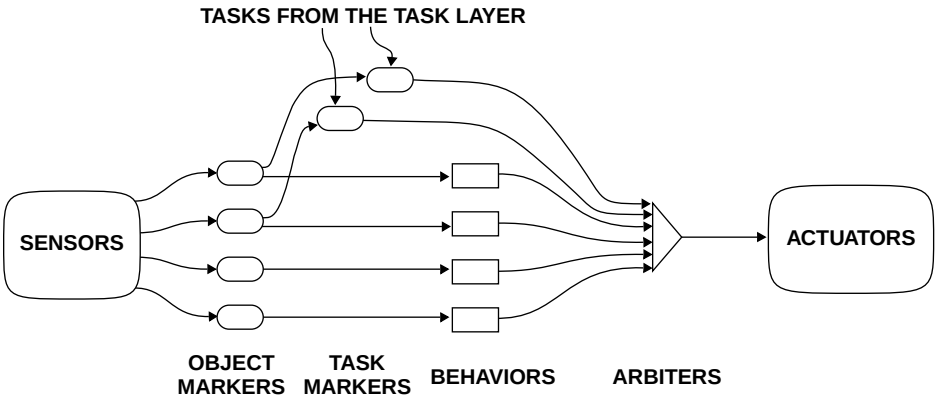

Fig. 1. The Samba architecture. In addition to the signals drawn in the figure, all modules can receive signals from sensors and object markers.

The control system is connected to the external world through Sensors and Actuators. Actuators change the local environment as reasoned by the behaviors. Sensors produce data about the local environment. A sensor can be also virtual, in which case it receives data from the other modules and processes it further. Object markers are simple processing elements that ground task related data on sensor data flow and communicate it to behaviors. Task markers form an interface between the reactive Samba system and the task layer reasoning the tasks.

Behaviors transform sensor and marker data into commands to actuators. Behaviors produce primitive reactions to objects, such as “go to an object” and “avoid an object”. The reactions are in the form of primitive action maps. A separate primitive action map is calculated for each primitive task of reaching or avoiding an object.

An action map specifies for each possible action how preferable the action is from the perspective of the producer of the map. The preferences are shown by assigning a weight to each action. The most common action map type is a velocity map. Velocity maps are three dimensional: an action is described by a velocity vector (direction, speed) and for each action there is a weight. The weights form a 2-dimensional surface in the polar coordinate system (direction, speed). In soccer also impulse maps are utilized. An impulse map presents weights for all the possible different impulses (direction, magnitude) the agent can give to the ball. It specifies the preferences for the different actions for kicking the ball to reach an object.

A weight for an action is calculated based on the time to collision. The shorter the time needed to reach an object, the heavier the weight for that action. For actions that do not result in a collision, the weights are calculated by approximating to what amount the action progresses the task of reaching the object. The resulting map contains only positive weights. A map of this kind is called a Goto map, because the performance of the action with the currently heaviest weight on such a map causes the agent to reach the corresponding object. An example of a Goto map is shown in Fig. 2. The map contains a ridge of heavy weights because the object is moving.

An action map containing negative weights is called an Avoid map. It is calculated either by negating the corresponding Goto map or by a separate procedure. As only positive weights cause actions to be performed, an Avoid map does not alone trigger any motion, but only prevents some actions. In other words, an Avoid map prevents collision with an obstacle when a target is being reached with the help of a Goto map. Fig. 3 shows an Avoid map.

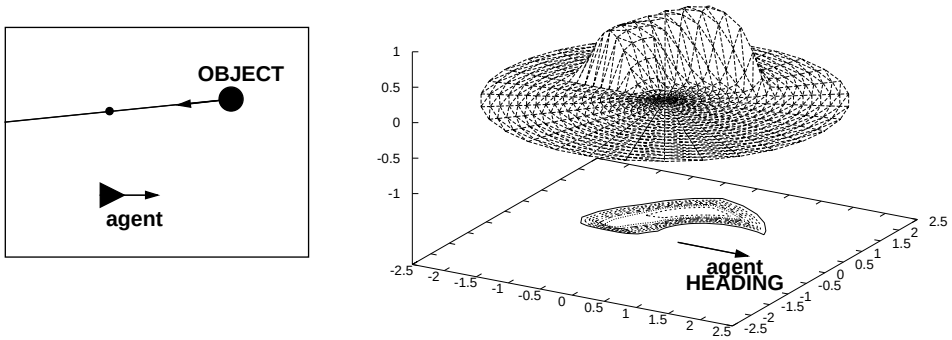

Fig. 2. Left: The situation. Right: An example of a Goto map. The agent reaches the object at the location marked with the small dot, if it executes the action with the heaviest weight on the Goto map.

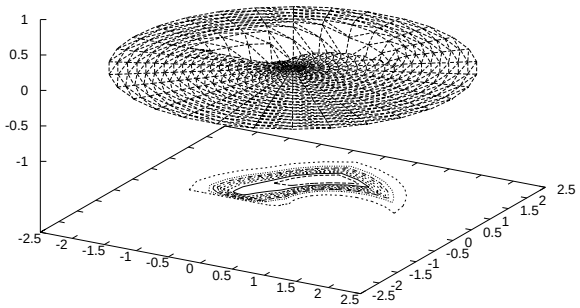

Fig. 3. An Avoid map corresponding to the Goto map shown in Fig. 2.

An arbiter selects and combines commands. The arbiter combines primitive Goto and Avoid maps into composite maps by the Maximum of Absolute Values (MAV) method. In this method, the weight with the maximum absolute value is selected for each action. In other words, the shortest time it takes to reach an object is selected for each action and the corresponding weight is stored on the composite map. Thus, the sign of a weight on a composite map specifies whether an obstacle or a target would be reached first if a certain action were performed. The absolute value of the weight specifies the time to collision with the object that would be reached first. Fig. 4 illustrates the MAV method for compiling action maps.

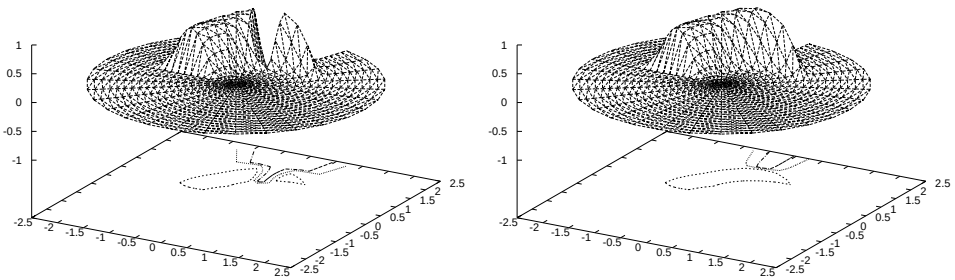

Fig. 4. A Goto map for a moving target and an Avoid map for a stationary obstacle compiled with the MAV method. Left: The obstacle is in front of the trajectory of the target, and actions in that direction are therefore forbidden. Right: The obstacle is behind the trajectory, and no actions are hence forbidden.

A task is performed by sending the action with the heaviest weight in the appropriate composite map to the actuators. The task to be executed at each moment is selected by the arbiter. A simple arbiter goes through the received composite maps in a predefined order and selects the best action from the first composite map producing an action good enough. A more versatile arbiter considers the current situation when selecting the composite behavior.