# A Hierarchical Behavior-Based Approach to Manipulation Tasks

Zbigniew Wasik and Alessandro Saffiotti
Center for Applied Autonomous Sensor Systems
Dept. of Technology, Örebro University
S-70182 Örebro, Sweden
zbych@aass.oru.se, asaffio@aass.oru.se

## Abstract

Typical mobile robots can be customized to perform a variety of different tasks by combining in different ways a set of basic control modules, or *behaviors*. By contrast, most current systems for manipulation are still designed for just one specific task. In this paper, we propose a hierarchical behavior-based system that can perform several vision-based manipulation tasks by using different combinations of the same set of basic behaviors. Behaviors can run concurrently, and they are arbitrated through "if-then" rules. We show experiments involving object tracking, grasping and placing, both with static and moving objects.

## 1 Introduction

Most current systems for (vision-based) manipulation are designed for just one specific task. Depending on the task and on the arm configuration, the design process can be pretty complex, especially when the task requires the use of a visual servoing approach [3]. This contrasts with the dominant design style in the mobile robotics community. Mobile robots are typically equipped with a set of basic control modules, or "behaviors", that perform atomic types of operations. The designer can then customize the system to perform a variety of different tasks by simply instantiating and combining these behaviors in different ways [12]. The goal of the present paper is to propose a behavior-based approach for manipulation tasks, and to illustrate its use to solve some different tasks involving both static and moving objects.

A few behavior-based approaches to manipulation have been already proposed in the literature. Several of these systems are based on Brooks' subsumption architecture [1], including Connell's early can retrieving robot [2] as well as systems based on fuzzy logic control [4] and on force-based control [15]. Other systems use finite state machines to sequence behaviors. For instance, Pettinaro [10] mostly uses sequential activations of generic arm behaviors to perform assembly tasks. De Giuseppe [5] uses sequential activation of fuzzy behaviors to perform a pick-up task.

All the above approaches are based on a winner-take-all strategy for behavior combination: only one behavior is in the control of the manipulator at any given time. In our previous work [14] we have proposed a system where several vision-based behaviors can be simultaneously active and are fused together using a fuzzy logic technique to achieve a overall coordinated motion of the arm to grasp static objects. We have shown that our approach results in increased modularity, smoother trajectories, and greater reactivity to unexpected events.

In this work, we show that our previous behavior-based system can be used to solve a variety of different manipulation tasks by combining the same basic behaviors in different ways. Moreover, we show that our behaviors can operate in moderately dynamic environments, in which the objects of interests are moving around, for example to perform tracking or grasping of a moving object. Differently from common approaches to visual tracking (e.g., [8, 7, 6]) our system does not need a predictive model of the object's motion: rather, we rely on the reactivity of fuzzy behaviors to adapt the execution to the changing position of the object. In case a predictive model is available, however, it can be incorporated in our behavior-based approach to better cope with faster object dynamics.

In the next Section, we recall the basic elements of our behavior-based system for manipulation, described in more detail in [14]. In Section 3, we show how these elements can be used to solve four different manipulation tasks: track, grasp, place, and repeated pick-and-place. In all cases, the position of the objects is not know *a priori*, and objects may be moving. Finally,

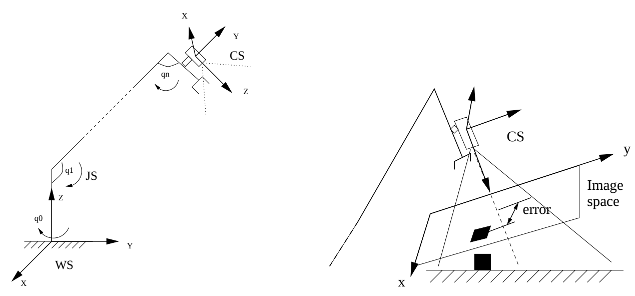

Figure 1: The arm and camera configuration.

we describe experiments performed on a real robot arm, which show that our system features the desired properties of reactivity, smoothness, and modularity.

## 2 Manipulation behaviors

A robot arm is a complex plant with usually more than 3 degrees of freedom, and it is typically controlled by giving desired joint angles (in joint space, JS) or by giving the desired external coordinates of the gripper (in work space, WS) [13]. In environments which are dynamic or not known *a priori*, visual feedback is often used as the major sensory input. In this case, all object's parameters are measured in an additional space: the image space (IS). The relation between the IS and the WS (or the JS) depends on the arm-camera configuration. Fig. 1 illustrates this relation for the "eye-in-hand" configuration considered in this paper.

### 2.1 Basic behavior set

To design a behavior-based control system, we must define a set of basic behaviors such that: (i) each behavior in the set implements a *simple* control strategy; and (ii) a large variety of tasks can be performed by *combining* these behaviors in different ways. While there are *de-facto* standard sets of behaviors for mobile robots, the literature offers few examples of behaviors for manipulation.

In this work, we use the following set of basic, unitary behaviors for simple manipulation tasks, extended from the set defined in [14].

**Center** Move the gripper in order to keep the object in the center of the image.

**Zoom** Move the gripper forward or backward in order to keep the size of the object to a given set-point.

**Surround** Adjust the position of the gripper so that the object is between the fingers.

**Catch** Grasp the object and move up a little.

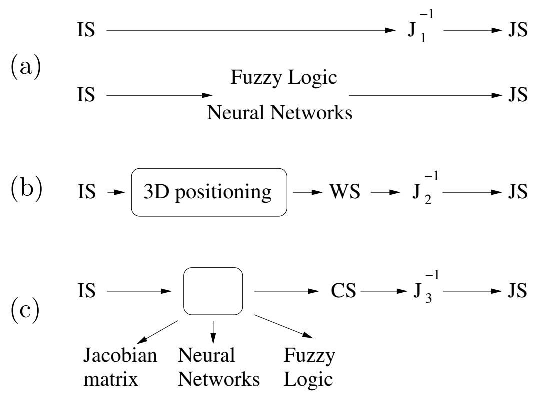

Figure 2: Approaches to visual servoing. (a) Image-based. (b) Position-based. (c) Our approach.

**Search** Visually explore the workspace by moving the gripper until an object appears in the image.

**FindFreeSpace** Move the gripper toward a free position on the pallet.

**PutDown** Release the object on the pallet.

**GoToPos** Move the gripper to a predefined position.

**GoHome** Bring the arm to its home configuration.

### 2.2 Vision-based behaviors

Some of the above behaviors only consider the position of the gripper and can be implemented by directly generating controls to the joints. Other behaviors, however, are defined in terms of the relative position between the gripper and a given object, and thus require input from sensory feedback, e.g. a camera. These behaviors take as input features in the image space (IS), like the size and position of the target object in the image, and must eventually generate controls in joint space (JS) [3]. In most approaches to visual servoing, the controller either maps the IS directly into the JS (image-based visual servoing), or it maps it to the WS which is then mapped into the JS (position-based visual servoing) — see Fig. 2.

In [14] we have proposed a different approach to implement vision-based control using an intermediate space, which we call the camera space (CS). The camera space is a 3D Cartesian space with its origin at the center of the camera and its $z$ axis along the optical axis of the camera, as shown in Fig. 1. The controller (behavior) maps input from the IS to control values defined in the CS. These are then transformed to joint controls by a Jacobian transformation [13], which is the same for all vision-based behaviors. Thus, each mapping from IS to CS defines a specific visual behavior. These mappings can be defined in several ways (Fig. 2): in our approach, we use fuzzy rule-based control systems [9]. The main motivation for this is that, once we introduce the notion of camera space, it becomes very easy for a designer to write intuitive control rules that implement basic vision-based motions like "center" or "zoom." No analytical model of the system is needed to do so. For example, the following rules, defined by the designer, implement the Center behavior:

**IF** object at left **THEN** GO(LEFT)
**IF** object at right **THEN** GO(RIGHT)
**IF** object is upper **THEN** GO(DOWN)
**IF** object is lower **THEN** GO(UP)

where the condition 'at left', 'at right' and so on are relative to the image. The rule conditions are evaluated from the features in IS, and produce a truth value between 0 (fully false) and 1 (fully true). Each rule affects the corresponding control variable (the first two the $x$ axis in CS and the other two $y$) by an amount that depends on the truth value of its condition: as a result, smaller or larger adjustments to the camera position will be generated depending on how much the object is close to the center of the image.

## 2.3 Complex behaviors

Complex behaviors are built by *combining* simpler behaviors together. In our approach we select behaviors in a purely reactive way using situation-action rules, and we fuse the output of concurrent behaviors by using mechanisms from fuzzy logic [11, 12].

We write complex behaviors using fuzzy meta-rules of the form

**IF** <condition> **THEN** USE(<behavior>)

For each such rule, the controller evaluates the truth value, in range [0,1], of <condition> and activates <behavior> at a level that corresponds to this value. Several behaviors may be active at the same time: in this case, their outputs are fused by a weighted combination according to the respective activation levels. Fusion is performed on each control variable independently. It is important that controls from vision-based behaviors are fused in the CS space (see [14]).

## 3 Using the behaviors

In this Section, we show that the above behaviors are sufficient to perform a variety of different vision-based manipulation tasks by combining them together in different ways. In each case, basic behaviors will be combined in a hierarchal way to form increasingly complex behaviors [12]. Basic behaviors will be graphically indicated by boxes with a thick border.

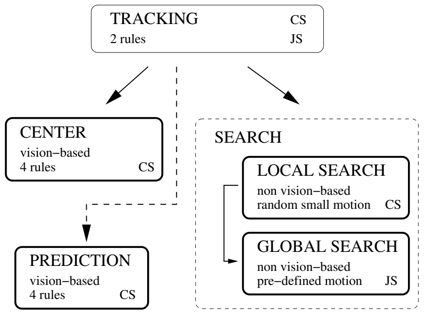

Figure 3: Implementation of a "Track" task using our behaviors

### 3.1 Tracking

The first task we consider is tracking of an object moving with unknown but relatively slow dynamics. A typical industrial application would be a robot arm that needs to catch objects that are irregularly placed on a transporter and may move on it, e.g. stones. The arm can track the objects until they enter in its workspace, and then grasp them without a need for further search.

The tracking task can be simply achieved by the composition of two basic behaviors: Center and Search. The following meta-rules implement the Tracking behavior, shown in a graphical form in Fig. 3 (The "CS" and "JS" labels indicate the space in which controls are generated).

**IF** object in image **THEN** USE(CENTER)
**IF** object not in image **THEN** USE(SEARCH)

As in the case of the rules for basic behaviors, the rule preconditions are evaluated on the input given by sensory feedback. The Center behavior, mentioned in Section 2, keeps object in camera's view. If the object is not found in the image, however, the Search behavior is activated. This behavior first searches a small area around the last known position of object by generating a random small motion in CS. This usually allows to reacquire the object if it moved out of the field of view, or if it was temporarily occluded, or in case of transient failures of the object detection routines. If after a while the object is still not visible, the Search behavior starts a systematic search by moving the camera to a set of pre-defined positions.

The reactive combination of Center and Search can effectively track objects that move slowly compared to the dynamics of the arm. In order to cope with faster moving objects, we may want to incorporate a simple Prediction behavior in our task: this behavior is similar to Center, except that it takes as input not the instantaneous position of the object but its future position computed by linear extrapolation. Since the control values produced by concurrent behaviors are averaged by the fuzzy fusion mechanism, we can incorporate the Predict behavior by simply adding it to the existing task, instead of replacing (or modifying) the Center behavior. This simplifies the incremental upgrade of an existing system.

## 3.2 PickUp

A more complex example of hierarchical composition of behaviors is provided by the grasping task shown in Fig. 4. The following meta-rules implement the top level behavior for this task, called PickUp.

**IF** if object is not in view
  **THEN** USE(SEARCH)
**IF** object is far from the gripper and in view
  **THEN** USE(APPROACH)
**IF** object is close to the gripper and in view
  **THEN** USE(SURROUND)
**IF** object is between fingers of the gripper and in view
  **THEN** USE(CATCH)
**IF** object is caught
  **THEN** USE(GOTOPOS)

This task uses basic behaviors from the set of unitary behaviors mentioned above plus a complex behavior, Approach, which is in turn built up from basic behaviors. This hierarchical design results in a smaller amount of rules and in the possibility to do incremental testing and tuning. The purpose of the Approach behavior is to bring the gripper close to the object of interest without loosing it from the image. The overall PickUp behavior generates a smooth motion of the camera/gripper that simultaneously keeps the object in the view of the image and brings the gripper closer to the object in order to catch it. As in the previous case, a predictive model of the object motion can be added by including an additional Predict behavior.

## 3.3 Place

A Place task can be easily designed by the composite behavior shown in Fig. 5. The goal of this behavior is to leave a grasped object at a free space on the pallet.

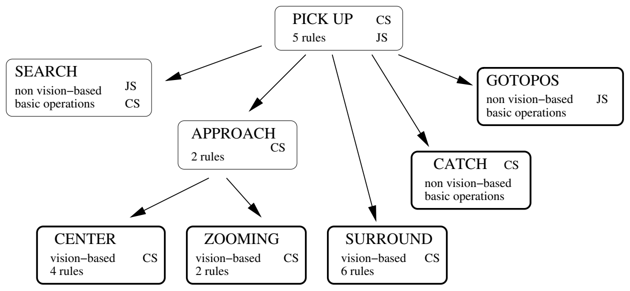

Figure 4: Implementation of a "PickUp" task using our behaviors

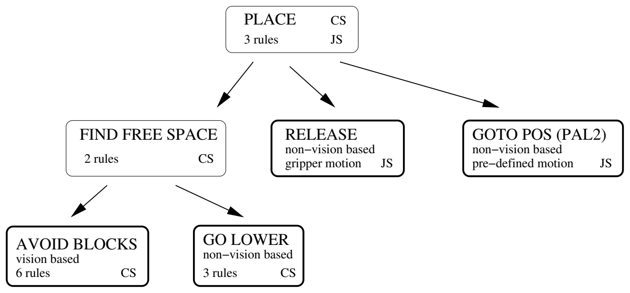

Figure 5: Implementation of a "Place" task using our behaviors

This behavior uses the basic behavior FindFreeSpace behavior mentioned in Section 2. When implementing the system, however, we have found it convenient to further decompose this latter behavior into two simpler ones:

**Avoid blocks** Center image on the largest area that contains no object

**Go Lower** Go toward the pallet and stay at proper distance above it.

This decomposition simplifies the design and makes it easier to modify this behavior to adapt to a different type of pallet.

## 3.4 Pick-and-Place

Finally, we can compose the above PickUp and Place behaviors to perform a complex Pick-and-Place task that picks up all the blocks found in one pallet and brings them to a second pallet. The number and position of the blocks are not known, and blocks can be moving. This task can be implemented by the simple finite state machine shown in Fig. 6. Switching from one state to the next happens if and only if the previous behavior has successfully completed its task, e.g., the PickUp behavior has caught a block. Recall the PickUp uses the Search behavior in order to cope with the unknown position of the blocks.

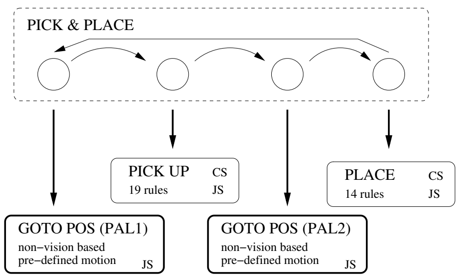

Figure 6: Implementation of a "Pick-and-Place" task using our behaviors

## 4 Experiments

We have implemented all the above behaviors on a 5DOF Scortec ER1 arm, shown in Fig. 7, and we have tested them in hundreds of experiments where the tasks were: 1) tracking the selected moving blocks, 2) to pick up all blocks from a pallet beside the arm and place them on another pallet.

In each experiment, the blocks were placed at random positions and moved randomly on the pallet. The image features used by the vision-based behaviors were the size, centroid and minor/major axis of all blocks (see Fig 7 right). Images were taken by a web camera mounted on the top of gripper.

Fig. 8 shows an execution of the Track behavior. The figure shows the trajectory of the gripper in the 3D space, together with its projections on the $xy$, $xz$ and $yz$ planes (dotted lines). The solid line on the $xy$ plane shows the trajectory of the object. At the beginning the arm follows the block and keeps it in the view. When the block disappears the Track behavior starts to search the block around the last known position. After the block is reacquired, the arm continues to move in order to keep it in the center of the field of view of the camera. This is done by composition of the translation and tilt motion of the gripper/camera. Note that we could track the object even without the Predict behavior, which was not used in these experiments since the objects moved in a random way and because we wanted to stress the reactivity of the system. This is different from most other visual servoing systems [8, 7, 6].

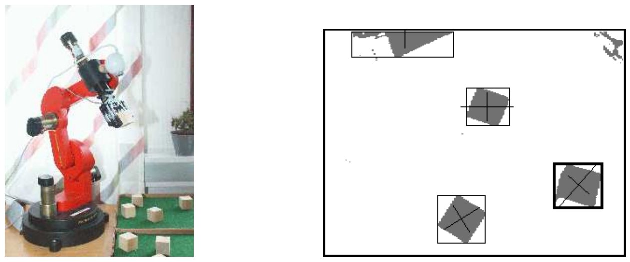

Figure 7: Left: The used arm. Right: The view after image processing.

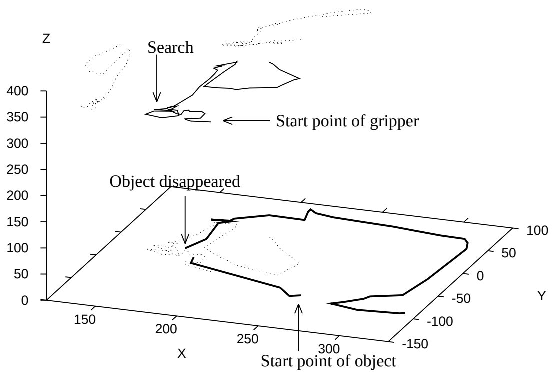

Figure 8: Performance of Track behavior

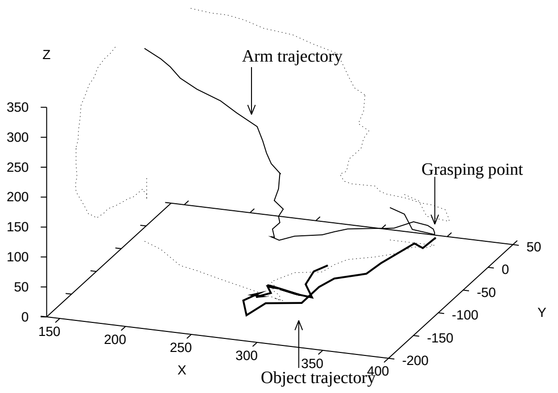

Figure 9: Performance of the PickUp behavior

In the next test the grasping task was examined. Fig. 9 shows the result of typical run. In order to test the advantages of the reactive rule-based arbitration at the beginning a block is placed in a random position and stays there for a while. After the block is found, the arm starts to move toward it. Normally the arm would grasp the static block and bring it up [14]. However, the block starts to move away from its position, and the arm tracks it and comes closer to it, until it can finally grasp it. Notice that this situation would be problematic if the behaviors were organized, say, in a finite state machine [10, 5].

The last example, shown in Fig. 10, illustrates the performance of the Pick-and-Place behavior. For graphical simplicity, we show this example with only one block. Like in previous examples the block was moving

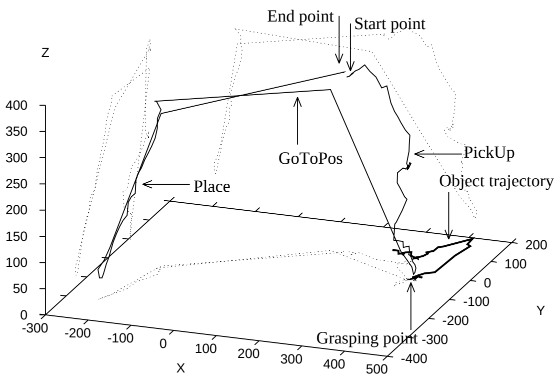

Figure 10: Performance of the Pick and Place behavior

on the pallet, as indicated in the plot. First the arm grasps the block using the PickUp behavior. Then, it goes up to a position safe from collisions and then moves to a position from where the second pallet is seen. The Place behavior is now executed, which puts the caught block at a free spot in the second pallet. At the end the arm goes back to the initial position ready to move the other blocks.

## 5 Conclusions

Our approach allows a designer to build complex controllers by combining in a hierarchical way a number of simple behavior-producing units. The key to design simplicity is that each "behavior" is only meant to achieve a simple, elementary goal under a limited set of conditions. The system then can be customized to perform different, possibly complex tasks by instantiating and combining these behaviors in different ways [12].

We have shown that the set of behaviors proposed in [14], and originally used to pick-up static objects, can be extended and used to solve several common manipulation tasks, including: search, track, pickup, place, and move blocks from one pallet to another. All tasks involve the use of visual feedback, and can be performed on both static and moving objects. Experiments show that the resulting system is highly reactive and that it produces smooth trajectories.

The behavior-based approach together with the hierarchical organization have greatly simplified the design task. Behaviors are designed and debugged incrementally, from the basic ones to the more complex ones. All tasks re-use the same set of basic behaviors, resulting in a small overall set of rules. For instance, the full PickUp behavior consists of 19 fuzzy rules, compared with, e.g., the over 120 used in [5].

## Acknowledgements

This work was partially supported by the Swedish KK Foundation.

## References

[1] R. Brooks. A layered intelligent control system for a mobile robot. *IEEE J. Robotics Automat.*, RA-2:14–23, 1986.

[2] J.H. Connell. A behavior-based arm controller. *IEEE Trans. on Robotics and Automation*, 5(6):784–791, 1989.

[3] P. Corke. Visual control of robot manipulators – a review. *Robotic and Autonomous Systems*, 7:1–31, 1993.

[4] P. Dassanayake, K. Watanabe, and K. Izumi. Fuzzy behavior-based control for a task of three-link manipulator. In *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 776–781, 1999.

[5] R. De Giuseppe, F. Taurisano, C. Distante, and A. Anglani. Visual servoing of a robotic manipulator based on fuzzy logic control. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1487–1494, 1999.

[6] W.E. Dixon, E. Zergeroglu, Y. Fang, and D.M. Dawson. Object tracking by a robot manipulator: a robust cooperative visual servoing approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 211–216, 2002.

[7] D. Kragic, A.T. Miller, and P.K. Allen. Real-time tracking meets online grasp planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 2460–2465, 2001.

[8] N. Papanikolopoulos, P.K. Khosla, and T. Kanada. Vision and control techniques for robotic visual tracking. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 857–864, 1991.

[9] K.M. Passino and S. Yurkovich. *Fuzzy Control.* Addison-Wesley Longman, Inc., 1998.

[10] G.C. Pettinaro. *Basic Set of Behaviours for Programming Assembly Robots.* PhD thesis, University of Edinburgh, 1996.

[11] A. Saffiotti, E. H. Ruspini, and K. Konolige. Blending reactivity and goal-directedness in a fuzzy controller. In *Proc. of the 2nd IEEE Int. Conf. on Fuzzy Systems*, pages 134–139, San Francisco, California, 1993.

[12] A. Saffiotti and Z. Wasik. Using hierarchical fuzzy behaviors in the robocup domain. In D. Maravall C. Zhou and D. Ruan, editors, *Autonomous Robotic Systems.* Springer-Verlag, Berlin, DE, 2002. In press. Online at http://www.aass.oru.se/˜asaffio/.

[13] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators.* Springer, 1959.

[14] Z. Wasik and A. Saffiotti. A fuzzy behavior-based control system for manipulation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1596–1601, 2002. In press. Online at http://www.aass.oru.se/˜asaffio/.

[15] T.G. Williams and Hardy N. Behavioural modules for force control of robot manipulators. In *Proc. IEEE Int. Symp. on Robot Control*, 2000.