Vision-Based Behaviors for Multi-Robot Cooperation

Yasuo Kuniyoshi* Jukka Riekki* § Makoto Ishii† Sebastien Rougeaux‡
Nobuyuki Kita* Shigeyuki Sakane* Masayoshi Kakikura†

* Autonomous Systems Section, Intelligent Systems Division, Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba City, Ibaraki 305, JAPAN

† Tokyo Denki University, 2-2 Kanda Nishiki-cho, Chiyoda-ku, Tokyo 101, JAPAN

‡ Institut d'Informatique d'Entreprise, 18 Allee Jean Rostand, 91000 Evry, FRANCE

Abstract

This paper presents some advanced examples of reactive vision-based cooperative behaviors: (1) Chasing and posing against another robot among others, (2) Unblocking the path of another robot by removing an obstacle, (3) Passing an object from one to another. These behaviors are demonstrated using real mobile robots equipped with CCD cameras, in a complex environment, and with no central controller or explicit communication among the robots. The action observation is based on real time processing of optical flow analysis and stereo tracking (ZDF). An extended behavior-based architecture for "cooperation by observation" is presented. The core extension consists of a mobile space buffer and an image space buffer with manipulable markers which control the internal flow of information, thereby coordinating parallel behaviors and achieving purposive tasks in complex and dynamic environments.

1 Introduction

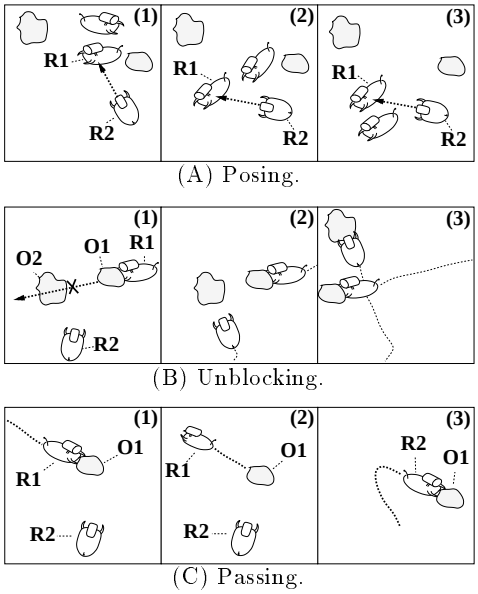
In decentralized multiple mobile robot systems, it is important to minimize the amount of explicit communication among the robots. However, the performance of complex cooperative behaviors depends on the information available about other robots [1, 2]. Therefore, each robot must collect such information using its own sensors if little or no communication is allowed.

Cooperation without explicit communication has been proposed [3] for cooperative navigation tasks. But, in order to achieve active cooperation in more complex tasks, we need sophisticated means for collecting richer information about other robots' actions.

Recently, a couple of methods have been presented for visual recognition of human tasks [4, 5]. Also, application of such methods as a mean for "implicit communication" in multi-agent domains has been suggested [6, 1].

We are aiming at achieving complex cooperative tasks with no centralized controller, based on (primarily visual) observation of other agents' actions, namely, "Cooperation by Observation" [7].

As a first step, we are focusing on "routine" [8] cooperative behaviors, e.g. stereotyped cooperative behavior patterns which we exhibit routinely in our everyday life. Such behaviors can be realized in a near-reactive framework. Moreover, since they are stereotyped, there is little need for explicit communication, and the necessary common knowledge is fixed and limited.



(A) Posing.

(B) Unblocking.

(C) Passing.

Fig. 1: Cooperative Behavior Patterns. The three basic cases are chosen and realized. (A) Chasing the target robot among others keeping a constant distance. (B) Helping the other transporter robot by removing the obstacle. (C) Passing a cargo from one robot to another.

This paper presents a concrete design of an agent architecture which is capable of autonomous execution of the basic cooperative behavior patterns shown in Fig. 1. These behavior patterns are demonstrated using real mobile robots in a complex environment without explicit communication among the robots.

2 Basic Cooperative Behavior Patterns

We have chosen and implemented three basic patterns of *Cooperation by Observation*, as shown in Fig. 1.

In this paper, we concentrate on the behaviors of the observer/helper robot "R2". Although these are the simplest

§ Jukka Riekki is visiting ETL as an STA fellow from the University of Oulu, FIN-90570 Oulu, Finland.

cases, they include no physical coupling between "R1" and "R2". Thus, they are good examples where visual observation is inevitable.

In pattern (A), "Posing", the observer robot is always oriented toward the target robot and keeps a constant distance from it by maneuvering its mobile base. We assume a complex dynamic environment, so the observer robot must not be distracted by other static or moving objects. Posing serves as a base behavior for vision-based cooperation, since it keeps the helper robot attending to the target robot.

Pattern (B), "Unblocking", is analogous to a common event in our everyday life, that is, if someone with a handful of baggage approaches a closed door, someone should open the door for the person. The robot "R1" is pushing "O1" and on its way ahead, there is an obstacle "O2". The helper robot "R2" anticipates the possible collision between "O1" and "O2", and helps "R1" by pushing "O2" aside from the path.

Pattern (C), "Passing", is an example of "bucket brigading" type of cooperation. The robot "R1" brings a cargo "O1" and leaves it in front of another robot "R2". The relay robot, "R2" waits until "R1" releases "O1", and then approaches and pushes the "O1" to carry it further.

3 The Agent Architecture

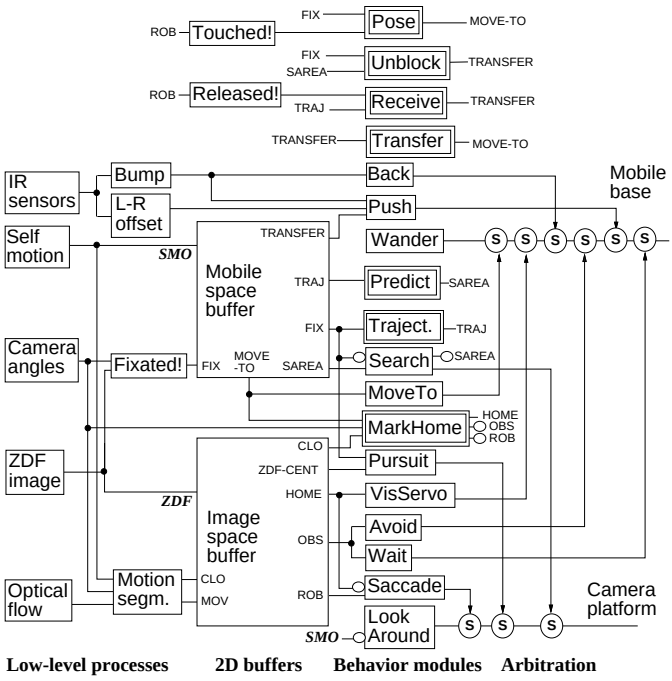


Fig. 2: The Organization of Vision Based Behaviors. The lines denote continuous flow of data. A module (box) takes input from its left side and outputs to the right or bottom. Small labels in capital letters denote markers. Circles labeled "s" are suppressor nodes. Circles at the input port denotes inhibition for modules and deletion for markers. Double frame boxes are abstract modules which manipulate markers/regions.

Figure 2 shows the organization of our agent architecture. It is partly inspired by the architectures proposed by Brooks [9] and Chapman [8], but with a significant extension using two types of buffers. The basic structure consists of four stages, low-level processes, intermediate representations (2D buffers), behavior modules, and arbitration network.

We assume that a robot has a power-wheel-steering mobile base with precise measurement of self motion (translational and angular velocity), a stereo gaze platform and two adjacent proximity (IR) sensors at the front edge. We also assume a flat ground plane throughout the workspace. Currently, we have implemented and tested the three cooperative behaviors independently. The architecture shown in Fig. 2 is a conceptual design which integrates all the implemented modules.

3.1 Low-level Vision

Among various visual information, motion and depth are most important for an observer in a multi-agent environment. We adopt optical flow for motion detection, and stereo Zero Disparity Filter (ZDF) [10] for depth selectivity. Figure 3 shows the two processes.

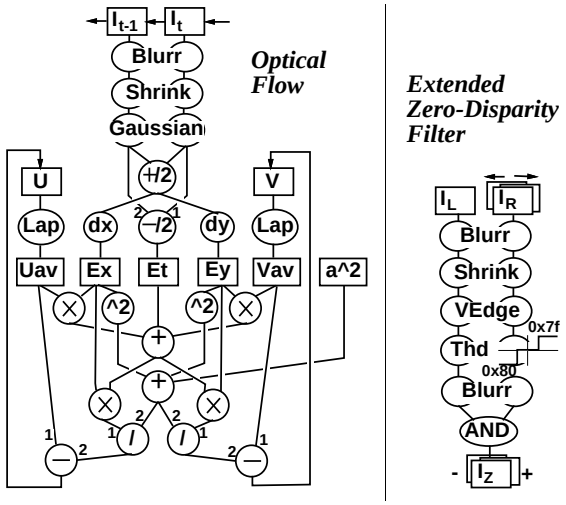
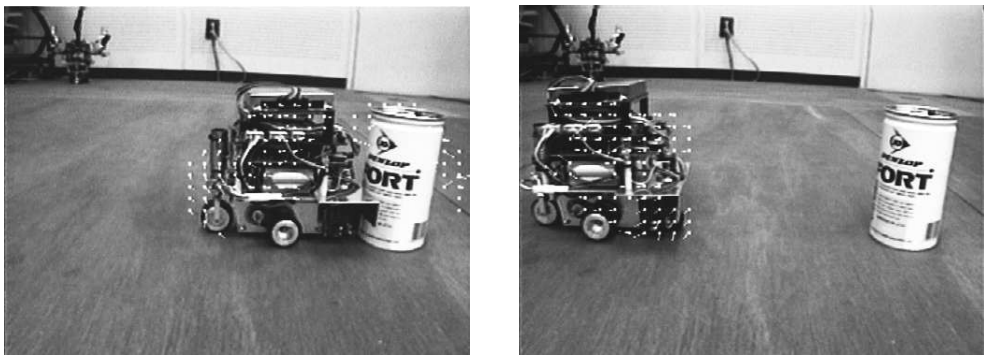


Fig. 3: Low-level Vision Processes. The circles denote computation and boxes denote data buffers. The numbers at the input lines denote the order of operands. "Lap" denotes Laplacian averaging filters.

3.1.1 Optical Flow The optical flow computation is based on the incremental version of the iterative scheme by Horn&Schunk [11]. The current values of optical flow (u,v) are stored in the buffers "U" and "V". Each time the spatial and temporal image derivatives Ex, Ey, Et are obtained from the successive input images, one or two iterations are executed to update the flow values. With our current implementation, the flow values are updated at 10Hz.

Figure 4 shows an example result of optical flow patterns extracted from the cooperative task situations.

3.1.2 Motion Segmentation Fig. 5 shows our method for motion detection. In order to cope with the significant noise in the optical flow as well as the intrinsic ambiguity, we currently assume strictly limited self motion whose velocity is precisely known to the system; Either pure forward translation, pure rotation, or stationary.



(a) (b)

Fig. 4: Optical Flow Patterns of a Release Action: (a) The robot is pushing a can to the right. (b) The robot has released the can and is moving left.

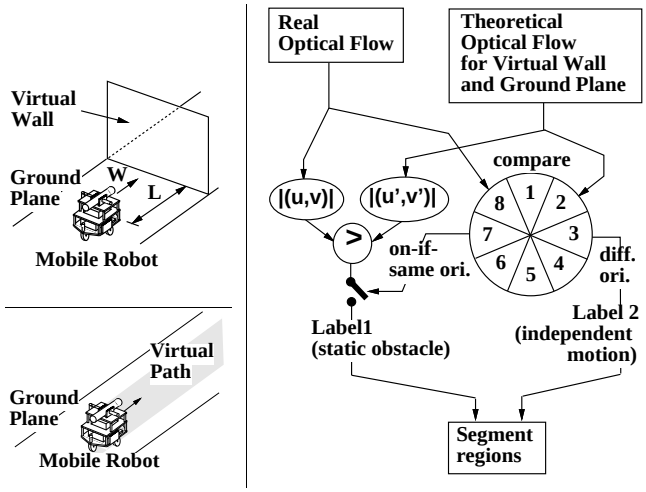


Fig. 5: Motion segmentation. The virtual wall and the ground plane (upper left) are used to precompute a theoretical background flow. The algorithm (right) detects independently moving objects and static objects closer than the virtual wall. For obstacle avoidance, a virtual path (lower left) is projected onto the image plane and used as a critical region for obstacle detection.

If the robot is stationary, motion segmentation is done by first thresholding flow intensity and then labeling by vector orientation. If the motion is pure rotation, the background flow can be completely calculated and subtracted from the real flow. In case of pure translation, we assume a "virtual wall" at a constant critical distance L ahead of the robot. L is typically chosen so as the robot must immediately start avoiding when an object closer than the virtual wall is detected. Static objects beyond the virtual wall are considered uninteresting. Theoretical flow is precomputed and stored for the virtual wall and the ground plane. The real flow vectors are compared with the theoretical flow in two steps; First, in quantized orientation. If they differ, the real flow vector is labeled as representing an independently moving object (with a velocity component orthogonal to the gaze line). If they match, their intensity is compared. If the real flow vector is larger, it is labeled as representing a close static obstacle. Regions are segmented using the labels and rough depth information is computed based on the camera angles and the ground plane assumption.

While navigating through the environment, the robot constructs its trajectory with straight lines and small arcs. It detects obstacles using the pure translation case on straight trajectories while on the small arcs (curves), it temporarily becomes "blind". When observing behavior (e.g. a releasing action) of other robots, the observer robot uses the stationary case in order to extract reliable motion information. The pure rotation case is used to search for a moving robot in order to start behavior observation.

3.1.3 Extended Zero Disparity Filter With vergence-controlled stereo cameras, we can construct an active tracking system which always captures the target images at the centers of both right and left CCDs. In this case, the target images have zero stereo disparity. The ZDF extracts zero disparity image features by extracting vertical edges and simply taking "AND" of the left and right edge images [10]. The result contains only those features lying on a "horopter", a circle defined by the two camera centers and the current fixation point. The original ZDF extracts a "strict" horopter, but we extended it by incorporating a limited search around the zero disparity [12]. Our extended ZDF is capable of tracking cross-horopter movement based on a vergence error value extracted from a "thick" horopter. By constraining the images to the limited regions around the image centers, we can realize a strong selectivity for the 3D region around the target. Figure 6 shows an example result of the ZDF.

3.2 Intermediate Representations

The mobile space buffer (MSB) and the image space buffer (ISB) are placed between the low level processes and the behavior modules. They are used for several purposes: (1) To represent spatial relationships among objects. (2) To select an image region (attention) to which the behavior modules should react. (3) As a short-term memory for remembering important local objects which falls off the view range as the robot moves or succades. (4) To coordinate different behavior modules by sharing common data among them. (5) To arbitrate behaviors by restricting their input data flow through a limited shared resource (markers).

MSB and ISB are indirectly connected via the stereo gaze

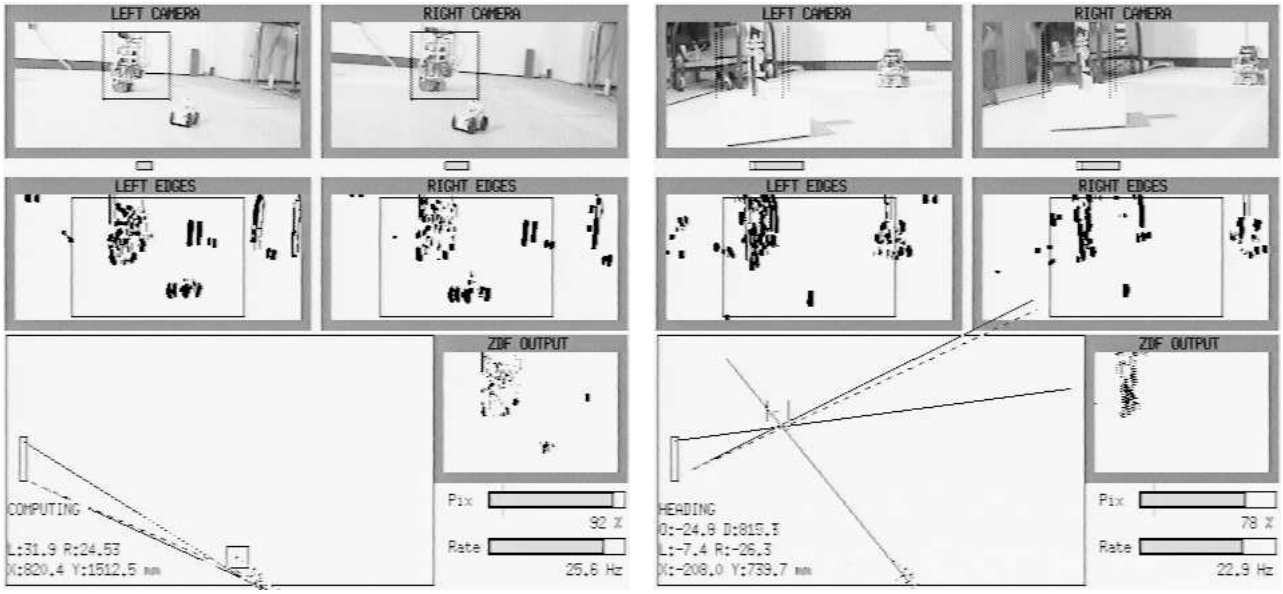


Fig. 6: Tracking a mobile robot (left). Trajectory estimation and obstacle detection (right). In the both figures, raw stereo images are shown at the top, extracted edges in the middle, ZDF outputs at bottom-right, and results of computation are shown at bottom-left. The entire processing is done at near frame rate (20 – 30Hz).

angles and the attentive modules.

3.2.1 Mobile Space Buffer The mobile space buffer (MSB) is a 2D egocentric qualitative local map. It is a network of a limited number of computing elements called "markers". Several types of markers are used: A point marker holds a position which can be continuously updated based on the stereo fixation process. A sequence marker holds a set of point markers as an ordered set. A relation marker holds two point markers. Their distance and orientation are constraints, i.e. they are used to maintain a constant relation between the point markers. A line marker associates a set of point markers with a line (LMS fitted). All the markers are represented in a local coordinates fixed on the mobile base. Thus, they are continuously updated to compensate self motion (*SMO* input in Fig. 2).

The FIXATION (FIX) marker represents the current fixation point. It becomes "active" with its position continuously updated when the fixated region contains an object, e.g. the ZDF output is positive ("Fixated!"). When FIX becomes active, it searches for other markers in MSB with the same position. If one is found, it is combined with FIX to continuously update its position until FIX is deactivated. TRAJECTORY (TRAJ) is a line marker. SEARCH-AREA (SAREA) is a sequence marker. It specifies the trajectory of the fixation point during a guided visual search. TRANSFER is a relation marker. The start point of the relation is associated with the object to be transferred. The direction and the distance of the transfer are calculated by the behavior that creates the marker. The end point is calculated automatically. MOVE-TO is a relation marker. The end point of the relation is associated with a reference position (an object or a virtual point). The start point of the relation, the goal position of self motion, is calculated automatically. The relation between these two points is specified by the behavior that creates the marker.

3.2.2 Image Space Buffer The image space buffer (ISB) contains image based features in the (right) camera coordinates. The features are represented as markers (regions and points) and continuously updated by the low-level vision processes.

ZDF-CENT marker holds a center of mass position of the ZDF output image. CLOSE-OBJECTS (CLO) and MOVING-REGIONS (MOV) are the regions labeled as close objects and independently moving objects by the motion segmentation process. OBSTACLES (OBS) are computed from CLO and MOV by intersecting them with the virtual path projected on the image plane (Fig. 5, lower-left.). ROBOT (ROB) represents other moving robots and is computed by subtracting OBS from MOV. HOME is computed by the module MarkHome as a part of CLO which corresponds to the MOVE-TO marker in MSB. It is used as a homing marker for the optical flow based (image space) steering servo VisServo.

3.3 Behavior Modules

The behavior modules are classified into four categories: Mobility, Visuo-ocular, Attentive, and Abstract.

3.3.1 Mobility Behaviors Following behaviors control the mobile base directly; "Push" is activated when a TRANSFER marker is active and the robot touches something. "Push" drives the robot in its current direction. It makes small modifications to the direction based on IR sensor data in order to keep the object in front of the robot. "Back" drives the robot backwards for a preset distance when the robot touches an object. "Back" is subsumed by "Push", so "Back" starts to control the robot after the transfer task is completed. "Wait" stops the robot whenever an obstacle comes too close to the robot. "Avoid" avoids both static and moving obstacles. This behavior is currently subsumed by "Push", because it is dif-

ficult to avoid obstacles while the robot is pushing an object. In such cases, the robot stops by the "Wait" behavior. Some other robot will hopefully come to push the obstacle away (the unblocking behavior). "MoveTo" controls the robot towards the position and orientation specified by the MOVE-TO relation marker. When the cameras are fixating on the target, the MOVE-TO marker is continuously updated and "MoveTo" works as a feedback loop. Otherwise, when the target is not fixated, "MoveTo" still works as an open loop control based on short term memory of the target position. When the robot reaches the goal, the MOVE-TO marker is cleared. "VisServo" is activated by the HOME marker in the ISB, and it steers the robot towards the marked object by quantizing the HOME position in three values (Left, Front, Right) and switching the steering angle to (-5, 0, 5)[deg] respectively. "VisServo" subsumes "MoveTo", because "VisServo" is always based on direct sensor data. "Wander" is a base behavior which takes over when the robot is idle. The robot moves around (with occasional halts) the workspace and looks for some other robot to help (the "LookAround" behavior).

3.3.2 Visuo-ocular Behaviors These behaviors control the cameras: "Succade" directs the cameras to a newly found moving robot. It takes an inhibition input from the HOME marker, so "Succade" is deactivated while visual servoing. It is also subsumed by "Persuit" and "Search", which means it is deactivated when the robot is paying attention to something. This restriction is too strong and should be relaxed in the future versions. "Persuit" continuously directs the cameras to fixate the zero disparity target as long as the ZDF output (FIX) is positive. "Search" is activated by the SAREA marker. It quickly scans the fixation point along the search area by swinging the cameras. The ZDF output is continuously monitored, and if something is found, the search behavior is switched off. The SAREA marker is cleared on completion. "LookAround" is the base behavior. When the robot is idle and stationary (*SMO* inhibition), it rotates the camera and scans the workspace for moving robots.

3.3.3 Attentive Modules These behaviors create markers that change the attention of the robot: "Trajectory" ("Traject.") records the recent temporal sequence of FIX positions and updates a TRAJ marker in MSB. "Predict" takes a TRAJ marker and predicts a future trajectory by line fitting, and specifies a search area (SAREA) in which an obstacle blocking the other robot might be found. "MarkHome" checks if the CLO in ISB matches the MOVE-TO marker in MSB. If they match, it places a HOME marker on the CLO region which is henceforth tracked within ISB. At the same time, "MarkHome" clears the portions of OBS and ROB which corresponds to the HOME marker, so that "Succade", "Avoid", "Wait" do not react to the region marked as HOME.

3.3.4 Abstract Behaviors The "abstract" behaviors coordinate lower behaviors by manipulating markers.

"Transfer" is activated by the TRANSFER marker, and it guides the robot to approach and push the target object in the distance and direction specified by the marker. "Transfer" computes an appropriate approach trajectory, as shown in Fig. 7. The trajectory is composed of two approach paths.

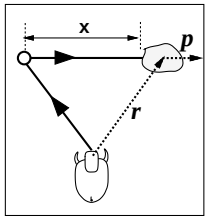


Fig. 7: Approach Trajectory to the Target. The circle denotes the posing position, x is the minimum distance required for correcting the self orientation by visual servoing, r denotes the target location and p denotes the desired pushing direction.

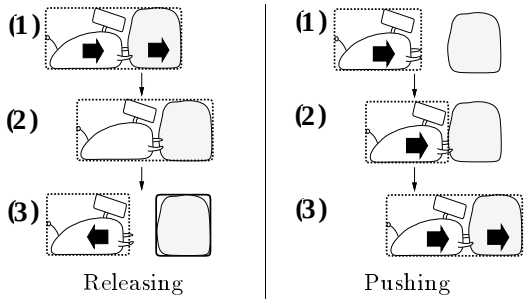


Fig. 8: Detecting Release and Pushing Actions. The key features are sudden changes of the size of the bounding boxes of the segmented optical flow fields.

First, a MOVE-TO marker with direction p is placed on the upper-left corner ($r - \mathrm{x}p$) on the path, where r is a positional vector of the target, p is the desired push direction, and x is the minimum distance necessary for correcting the robot's approach trajectory by visual servoing (determined experimentally, currently set to 0.5[m]). Then after the robot reaches the point, the marker is replaced to the target object. If the target is found using optical flow, visual servoing is automatically invoked as described earlier. Finally, when the robot touches the object, the IR sensors goes on and the robot pushes the object away.

"Unblock" is activated when the FIX marker is in SAREA. It places a TRANSFER marker at the current FIX position with a direction orthogonal to SAREA. "Transfer" behavior takes care of the rest of the task.

"Receive" is activated when a releasing action is perceived, and initializes a transfer task similarly to the "Unblock" behavior. "Released!" detects a sudden change in the optical flow pattern, as shown in Fig. 8. Initially, when the target robot is pushing an object, a large homogeneous flow region is observed (1). When the robot stops and releases the object, the intensity of the flow vectors goes off to zero (2), but the size of the region is kept unchanged. When the robot leaves the object and starts to go back (3), a drastic change is observed, e.g. the flow region shrinks suddenly and the average velocity orientation is reversed. This defines the releasing action. The object left behind is segmented by differentiating the shape of the flow regions before and after releasing. "Receive"

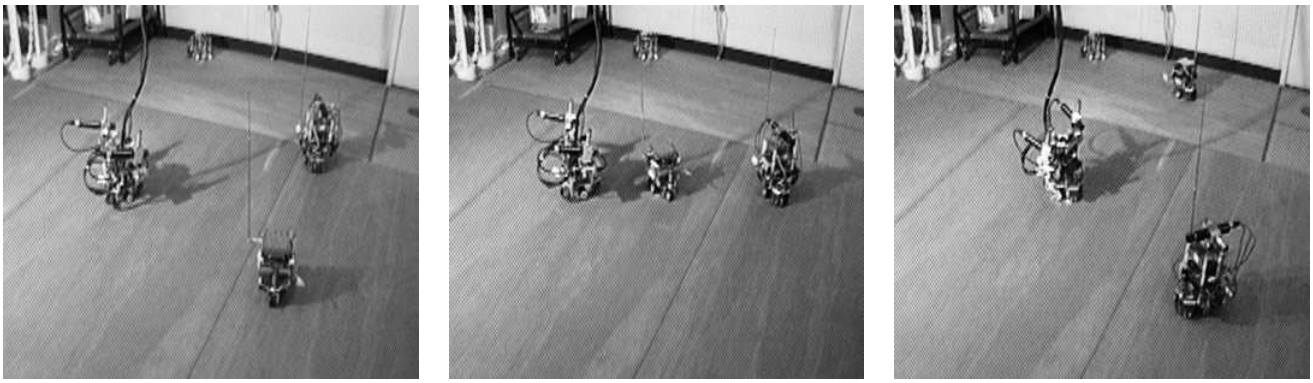
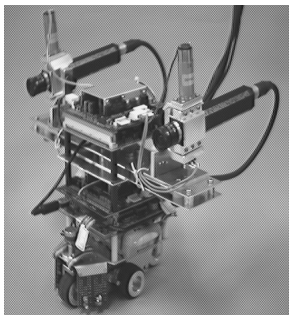


Fig. 9: Posing behavior. The chaser (left) follows the target (moving downward). Even when the distractor (moving upward) comes in between the chaser and the target, the chaser is not disturbed and locks onto to the target.

marks the region with a TRANSFER marker. Figure 8 also shows a method for detecting the beginning of a pushing action ("Touched!"). This method was originally demonstrated by Tistarelli and Sandini [13].

"Pose" is activated whenever the FIX marker is valid. It places a MOVE-TO marker (with a default distance) at the FIX position. "Pose" stops the robot when the target robot touches something in order to facilitate behavior observation.

4 Experiments



Total spec	:	
length	:	0.20[m]
width	:	0.30[m]
height	:	0.28[m]
weight	:	2.5[kg]
max.speed	:	0.20[m/s]
Gaze platform:		
DOF(verge)	:	2
baseline	:	0.25[m]
range	:	±30[deg]
precision	:	±0.4[deg]
max.speed	:	50[deg/s]

Fig. 10: A mobile robot with a stereo gaze platform.

Fig. 10 shows one of our mobile robot equipped with a stereo gaze platform. The gaze platform has 2 DOF for vergence control. The robot has two onboard processors, one for controlling the mobile base and another for the gaze platform. Image processing is currently done on a remote host processor which consists of pipeline image processors (Datacube Maxvideo system), and a CPU board running a real time operating system. The system configuration is given in Fig. 11.

So far, we have implemented and tested the three cooperative behaviors separately. The results are given below.

Figure 9 shows snapshots from the experiment of the "Posing" behavior. The mission of the chaser robot is to keep a fixed relative position (distance and rotation) with regard to the target robot. There is a distractor robot moving around, that sometimes comes in between the chaser and the target. The chaser must lock onto the target. This behavior supports

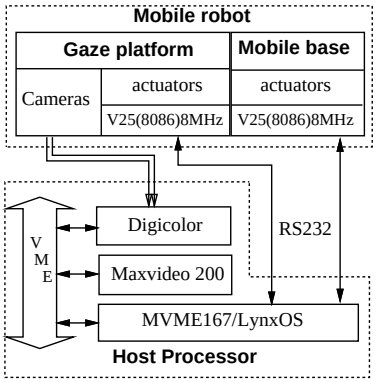


Fig. 11: Overview of the prototype system.

other cooperative vision-based behaviors in complex, dynamic environments. From the viewpoint of social structures, the chaser and the target form an ordered pair in this case, and the distractor is isolated, not spatially, but at the level of cooperative behaviors.

In this experiment, only the chaser was computer controlled, where as the others were manually controlled. The purpose of the experiment was to test the stereo tracking and visual servo routines. They turned out to be quite robust.

We have made a simplified experiment of the unblocking task discussed before. In this experiment, the transporter robot (R1) does not carry an object (O1) (See Fig. 1). This simplification is justified because the purpose of this experiment is to verify the visuo-motor routines designed for the helper (R2).

Fig. 12 shows a result. The helper robot successfully found the anticipated obstacle and helped the transporter robot by pushing away the obstacle.

The realized "Pass" behavior pattern is shown in Fig. 13–14. Figure 13(a)–(c) shows the result of motion segmentation and release detection. The extracted optical flow patterns were shown earlier in Fig. 4. The released can is detected and the receiver robot starts to follow the computed approach trajectory (d). This is done in an open-loop manner using the mobile

space buffer because the can goes out of sight temporarily. After a "Pose" behavior, the can becomes visible again (e), then the robot approaches it by visual servoing (f) – (g). Figure 14 shows a bird-eye view of the entire situation.

5 Discussions

A concrete system design and real experiments are presented for the three basic patterns of "Cooperation by Observation", a framework without explicit communication and centralized control.

The presented architecture is strongly inspired by the "Sonja" system [8] with intermediate buffers for controlling the information flow and coordinating parallel behaviors in order to achieve purposive tasks. The coordination is realized not by explicit control but through manipulating markers grounded on sensor data flow. So, it fits well to the behavior based architecture. The important extensions we made are the development of attentive procedures such as "Predict" and "Search" for the 3D domain, and the integration of image space data with mobile space data based on active vision framework. The

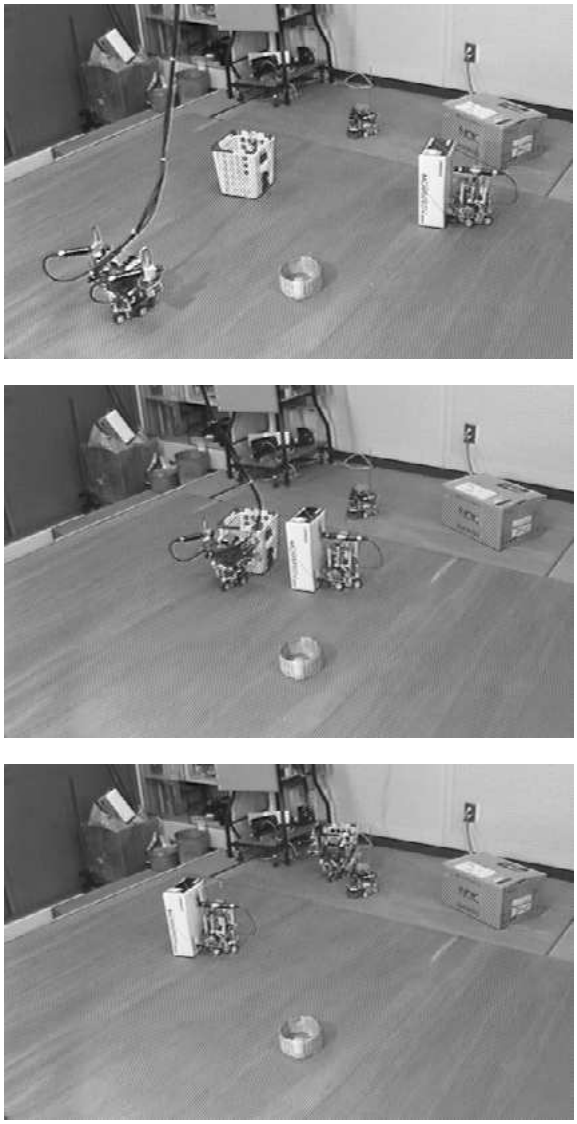


Fig. 12: Unblocking behavior: The binocular robot (helper) on the left watched the monocular transporter robot pushing a block from right to left, estimated the path and found the possible obstacle on the path (top figure), then quickly approached the obstacle using visual servo (middle figure), and successfully pushed it away to help the transporter (bottom figure).

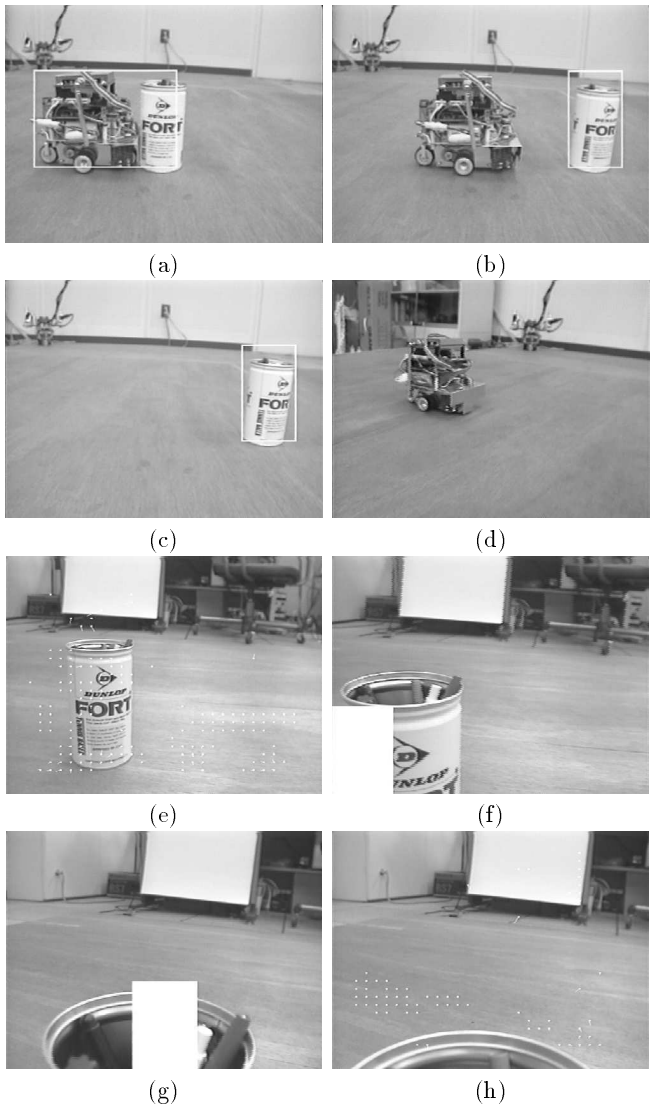


Fig. 13: Vision Processing during a Receiving Action; A view from the receiver robot. (a) A segmented optical flow region including the robot and the can. (b) A release action is detected. (c) The released can is segmented. (d) Starting an open loop MoveTo behavior. (e) MoveTo behavior completed and recaptured the can. (f),(g),(h) VisServo behavior based on optical flow. The white box indicates the HOME marker.

intermediate buffers also have an implicit behavior arbitration functionality, because a limited number of markers are shared among parallel behavior modules. This greatly simplifies the arbitration network at the final stage.

Our current architecture immediately reacts to a known behavior pattern such as releasing an object. This is justified because currently we deal with routine, fixed types of cooperation. But behavior observation often suffers from ambiguity, e.g. whether the other robot wants to avoid an object or fetch it. The next step would be to consider how to combine limited inter-robot communication and common knowledge with behavior observation in order to control the high level behaviors.

The use of optical flow and the zero disparity filtering turned out to be quite powerful for "cooperation by observation" in complex multi-robot environments, as shown in the experiments. In order to extract meaningful information from low-level vision, the active vision paradigm [14] played a significant role. Active vision also played an important role in purposive behavior coordination. The Mobile Space Buffer and the Image Space Buffer are combined through the camera gaze angles and visual attention. Currently we assign a low priority to the "Succade" behavior such that it does not take over while the robot is paying attention to something. This restriction is too strong and we are trying to improve it. If we succeed, our robot will be aware of changes in its surrounding while consistently achieving a purposive task.

References

[1] L. E. Parker. Designing control laws for cooperative agent teams. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 582–587, 1993.

[2] M. J. Mataric. Minimizing complexity in controlling a mobile robot population. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 830–835, 1992.

[3] R. C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *J. Robotic Systems*, 9(3):351–364, 1992.

[4] Y. Kuniyoshi, M. Inaba, and H. Inoue. Seeing, understanding and doing human task. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1992.

[5] K. Ikeuchi and T. Suehiro. Towards an assembly plan from observation. Technical Report CMU-CS-91-167, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA 15213, USA, 1991.

[6] Y. Kuniyoshi and H. Inoue. Qualitative recognition of ongoing human action sequences. In *Proc. IJCAI93*, pages 1600–1609, 1993.

[7] Y. Kuniyoshi et al. Cooperation by observation – the framework and basic task patterns –. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 767–774, 1994.

[8] D. Chapman. *Vision, Instruction and Action*. MIT Press, 1991. ISBN 0-262-03181-7.

[9] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Trans. Robotics and Automation*, RA2:14–23, 1986.

[10] P. von Kaenel, C. M. Brown, and D. J. Coombs. Detecting regions of zero disparity in binocular images. Technical report, University of Rochester, 1991.

[11] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 1:185–203, 1981.

[12] S. Rougeaux, N. Kita, Y. Kuniyoshi, S. Sakane, and F. Chavand. Binocular tracking based on virtual horopters. In *Proc. IROS'94*, 1994.

[13] M. Tistarelli and G. Sandini. Dynamic aspects in active vision. *CVGIP: Image Understanding*, 56(1):108–129, 1992.

[14] J. Aloimonos. Purposive and qualitative active vision. In *Proc. Int. Workshop on Active Control in Visual Perception*, 1990.

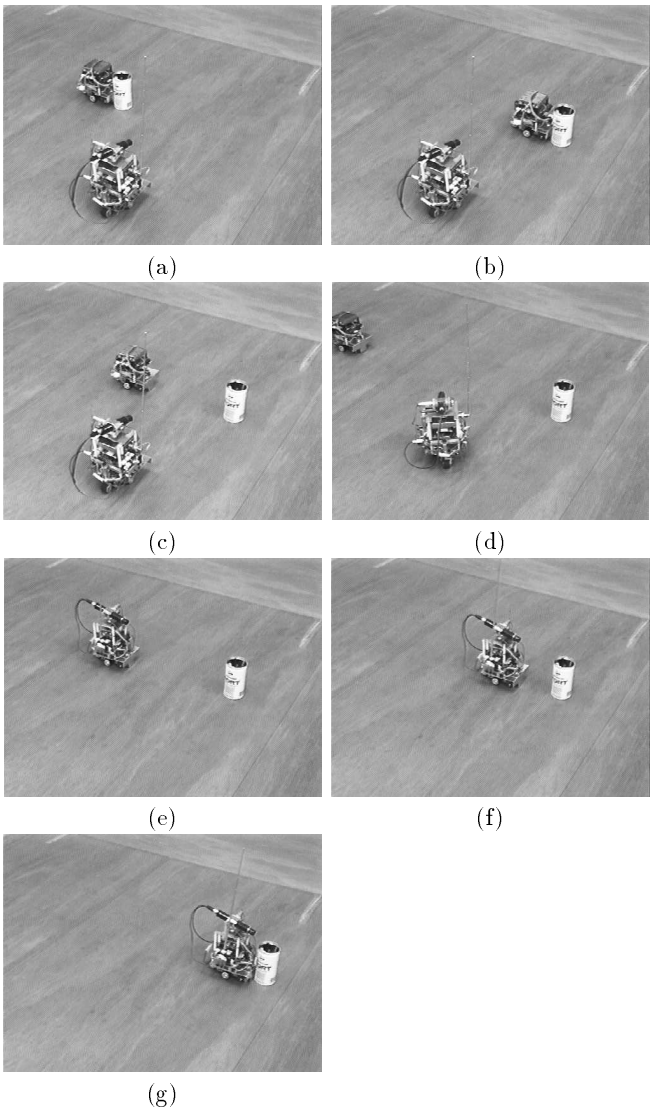


Fig. 14: Passing Task. (a)(b) A robot is pushing a can from left to right. (c) The first robot releases the can and returns to fetch another can, and the helper robot recognizes the situation. (d) The helper robot determines the pushing direction, and starts to follow a computed approach trajectory. (e)(f) The helper robot approaches the can using visual servo. (g) Pushing the can in the same direction as the first robot did.