Proceedings of the 1992 IEEE
International Conference on Robotics and Automation
Nice, France - May 1992

Behavior-Based Mobile Robot using Active Sensor Fusion

I. Kweon, Y. Kuno
Research & Development Center
Toshiba Corporation
1 Komukai, Toshiba-Cho, Saiwai-Ku
Kawasaki, 210, Japan

M. Watanabe, K. Onoguchi
Kansai Research Lab.
Toshiba Corporation
8-6, Motoyama-Minami-Cho, Higashinada-Ku
Kobe, 658, Japan

Abstract

We present a navigation system using multiple sensors for unknown and dynamic indoor environments. To achieve the robustness and flexibility of the mobile robot, we develop a Behavior-Based system architecture, consisting of multi-layered behaviors using multiple sensors: ultrasonic sensors and a video camera.

Basic behaviors required for navigation, such as, avoiding obstacles, moving towards free space, and following targets, are redundantly developed as agents and combined in a behavior-based system architecture.

We demonstrate the capabilities of our system in unstructured real office environments, using an indoor mobile robot developed by Toshiba.

1 Introduction

In mobile robot navigation, there are two distinctive approaches: (1) the conventional "SMPA (Sensing-Modeling-Planning-Action)" approach and (2) the "behavior-based" approach. The SMPA approach consists of modules that each sense the world, build two or three dimensional models, and plan actions for the robot using the model in a sequential manner [5]. All the modules must be complete and working before any action takes place. A serious problem in this approach is reliability. If one module fails, either due to software or hardware bugs, the entire system will break down.

On the other hand, the behavior-based approach decomposes the system into individual modules, each of which is responsible for one behavior to be performed by the entire system [1, 2]. Each behavior contains a complete path, from sensing to action, and is executed in a completely parallel manner. The behavior-based approach is more reliable than the SMPA approach. Even if one module fails, other behaviors can still produce meaningful actions for the robot. However, it shows inefficiency in achieving a mission because a behavior-based system with fixed priorities tends to adopt a lower-level reflexive behavior in a complex environment.

To solve these problems and to achieve the desired robustness and flexibility of the behavior-based system, we employ two approaches: a three-clustered system architecture and redundant behaviors using multiple sensors. The three-clustered system architecture consists of a manager (Motion-Executor) and three-clustered behaviors: the reflexive, purposive, and adaptive level behaviors. The reflexive-level group consists of agents which maintain minimal safety of the robot. The role of the purposive-level group is to achieve the global mission of the robot: to navigate the robot to the final goal efficiently. The adaptive-level agents save the failure of the purposive-level agents or deadlock situations. The details of the three-clustered system architecture are described in [11]. In this paper, we mainly describe redundant behaviors using both ultrasonic sensors and a video camera.

Multi-layered behaviors include the reflexive-level ***obstacle-avoider***, the adaptive-level ***free-space-explorer***, ***wall-follower*** and ***open-space-explorer***, and the purposive-level ***target-tracker*** and ***target-searcher***. Vision-based target tracking behaviors are complemented with sonar-based behaviors. For example, intersection tracking behavior is coupled with the sonar-based wall following behavior. Doorway tracking behavior is also associated with the sonar-based open-space-explorer. Such redundant behaviors using two different sensors can be very useful for detecting and recovering from failure of each behavior.

For each behavior, an Energy-based motion decision method determines the robot's position and orientation to minimize an Energy-based cost function. The Energy-based cost function consists of two terms: internal energy and external energy. Internal energy includes the smoothness constraint about the robot trajectory. The range measurements from multiple sensors act on the robot as external energy.

In the following sections, we first describe an active sensor fusion method to combine multiple sensor data, along with the principle of the Energy-based motion decision algorithm, which determines the robot location using range measurements. This principle is then exploited to develop range-based behaviors in Section 3. In Section 4, we describe a vision-based behavior, target tracking, which can guide the robot by tracking a specified target. In Section 5, we discuss behavior arbitration mechanisms and the Motion-Executor. In Section 6, we present the experimental results of a mobile robot equipped with four ultrasonic sensors and a video camera.

2 Active Sensor Fusion

In a behavior-based mobile robot, each behavior computes a motion command directly from sensor data without building any internal representation. Therefore, behaviors become more robust when multiple sensor data are used to compute motion commands. In this section, we present a method, based on active contour models, to compute the position and direction of the robot by fusing multiple sensor data.

To determine the best position and orientation of the robot using the range readings from multiple sensors, we make use of energy-minimizing curves, known as "snakes", which were introduced in [8]. In their original formula, snakes, $c(s)$, are computed by minimizing an energy functional E

$$\begin{aligned} E &= \int_s (E_{int} + E_{ext}) ds \\ &= \int_s \alpha \|c'(s)\|^2 + \beta \|c''(s)\|^2 + P(c(s)) ds \quad (1) \end{aligned}$$

where the primes denote differentiation and P is the potential associated to the external forces. The elasticity and rigidity of the model are controlled by α and β, respectively.

If c is a local minimum for E, it satisfies the associated Euler-Lagrange equation:

$$-(\alpha c')' + (\beta c'')'' + \nabla P(c) = 0. \quad (2)$$

In this formulation, each term appears as a force applied to the curve. A solution can be seen as realizing the equilibrium of the forces acting on the curve.

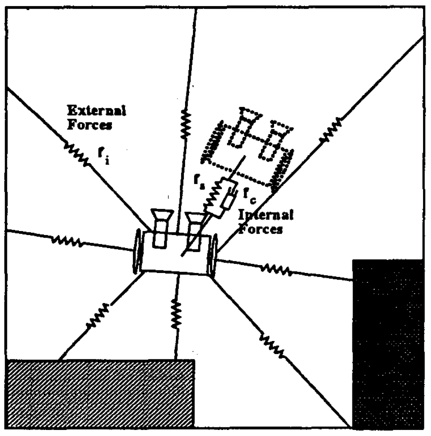

Figure 1: The Energy-based motion decision algorithm.

In the present work, we apply the same algorithm to compute the position and orientation of the robot, at which the equilibrium of all forces is maintained. Forces acting on the robot include internal and external forces:

- As internal forces, we consider a smooth force, $\mathbf{f}_s$, from the smoothness constraints on the robot's trajectory and a damping force, $\mathbf{f}_d$, from the robot dynamics [1],
- The range forces, $\mathbf{f}_i$, from multiple range measurements, pushes the robot to the location where equilibrium is achieved between the range forces (the safest position), and
- Other external forces, $\mathbf{f}_t$ (e.g., the target forces) can be added to attract the robot.

The location of the robot is presented by a 2D vector

$$\mathbf{v} = (x, y) \quad (3)$$

where (x, y) is the Cartesian coordinate of the robot's position on the floor.

Then, the internal smoothness force is computed by

$$\mathbf{f}_s = \alpha(\mathbf{v}_i^+ - \mathbf{v}_i) \quad (4)$$

where $\mathbf{v}_i^+$ is the ith location, computed from the smoothness constraints. and $\mathbf{v}_i$ is the current robot

[1] We can easily add another external force due to acceleration of the robot

position. α is a parameter to control the smoothness in the robot movement.

Based on simple trigonometry, it is easy to compute the robot's location, $\mathbf{v}_i^+$, satisfying the smoothness constraint by [10]

$$\begin{aligned} \mathbf{v}_i^+ &= (x_i^+, y_i^+)^t \qquad (5) \\ &= \begin{pmatrix} 2x_{i-1} - x_{i-2} - 2(y_{i-1} - y_{i-2})\tan(\frac{\Delta\theta}{2}) \\ 2y_{i-1} - y_{i-2} - 2(x_{i-1} - x_{i-2})\tan(\frac{\Delta\theta}{2}) \end{pmatrix} \end{aligned}$$

where $\Delta\theta$ is the angle between the two lines formed by three robot locations.

The internal damping force is given by

$$\mathbf{f}_d = d(\mathbf{v}_i - \mathbf{v}_{i-1}) \qquad (6)$$

where $\mathbf{v}_i$ and $\mathbf{v}_{i-1}$ are the current and previous robot locations, and d controls the effect of the dynamics of the robot.

The range force for each sensor is simply computed as

$$\mathbf{f}_i^j = k(\mathbf{s}_j - \mathbf{v}_i) \qquad (7)$$

where $\mathbf{s}_j = (x_s^j, y_s^j)$ represents the Cartesian coordinate values for the jth range reading, and k is the range force parameter which controls the effect of the range force on the total energy functional, E.

From (4), (6), (7), a new location of the robot, satisfying the force equilibrium condition from multiple sensors, is simply computed by

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \sum_{j=0}^{n} \mathbf{f}_i^j + \mathbf{f}_d + \mathbf{f}_s \qquad (8)$$

where n is the number of sensors. The new location is controlled by three parameters: α, d, and k for the smoothness constraint, damping force and image force, respectively.

3 Range-Based Behaviors

Many mobile robot systems use acoustic sensors, because they provide an inexpensive means to obtain range information around the robot by computing the echo travel time. The ultrasonic ranging sensor has been also proven to be very effective for indoor navigation.

Using the Energy-based motion decision algorithm and ultrasonic sensors, we present four range-based behaviors: free-space exploring, obstacle avoiding, obstacle-boundary following, and open-space exploring.

3.1 Free-Space-Explorer

The free-space-explorer pushes the robot to the largest open space, which corresponds to the safest area for the robot. To achieve this goal effectively, the free-space-explorer uses the Energy-based method, with the range force parameter, k, at around unity. This results in the range forces from sensors having a major influence and the robot internal energy having a small effect. In this case, the robot moves to the location where equilibrium between range forces is achieved.

3.2 Obstacle-Avoider

The obstacle-avoider only uses range measurements reflected from nearby obstacles, because any range measurements that are greater than a specified distance guarantees the robot safety. For the obstacle-avoider, the range forces from nearby obstacles act as repulsive forces on the robot (Remember that, in the original Energy-based motion decision, the range forces act as the attractive forces). The repulsive range forces pushes the robot to a new location.

The range force is inversely proportional to the range measurement. In other words, nearer obstacles provide the greater repulsive forces. In (8), the range force , $\mathbf{f}_i^j$, is replaced by a repulsive range force:

$$\mathbf{f}_i^j = \begin{cases} -k\mathbf{s}_j & \text{if } \rho_j < \rho_s \\ 0 & \text{otherwise} \end{cases} \qquad (9)$$

where $\mathbf{s}_j$ is a Cartesian coordinate value of the jth range measurement ρ_j and ρ_s is the minimum safety distance for the robot.

3.3 Obstacle-Boundary-Follower

The obstacle-boundary-follower uses an algorithm similar to the obstacle-avoider. In the case of the free-space-explorer and the obstacle-avoider, the range forces from nearby obstacles were either attractive or repulsive. For the obstacle-boundary-follower, the range measurements from nearby obstacles are used to compute tangential forces. In (8), the range force ($\mathbf{f}_i^j$) is replaced by a tangential range force:

$$\mathbf{f}_i^j = \begin{cases} k(\mathbf{s}_j - \mathbf{s}_{j-1}) & \text{if } \rho_j < \rho_{th} \text{ and } \rho_{j-1} < \rho_{th} \\ 0 & \text{otherwise} \end{cases} \qquad (10)$$

where $\mathbf{s}_j$ is a Cartesian coordinate value of $j-th$ sonar measurement ρ_j, and ρ_{th} is a threshold distance for nearby obstacles.

The robot will only be pushed by these tangential forces. As long as good range measurements are available without any systematic error, such as specular reflection for sonar, the robot will never collide with obstacles, because it will always move along the tangential directions of nearby obstacles.

The failure of this behavior can be easily detected by the null motion command from this behavior which corresponds to two possible situations: (1) there are no obstacles to follow and (2) obstacles are not detected due to sensor error. Additional sensing from the deliberate motion of the robot or other sensor is required to detect the real failure. In the present work, we use the exploratory rotational motion of robot to obtain multiple range measurements at one robot location. These multiple range measurements are then used to confirm the failure of the behavior.

3.4 Open-Space-Explorer

In indoor navigation, a robot often needs the open-space-explorer which can find an exit (e.g., a doorway) in a room by just using robust reflexive behaviors, without using any sophisticated vision systems or algorithms. We present the open-space-explorer based on the free-space-explorer.

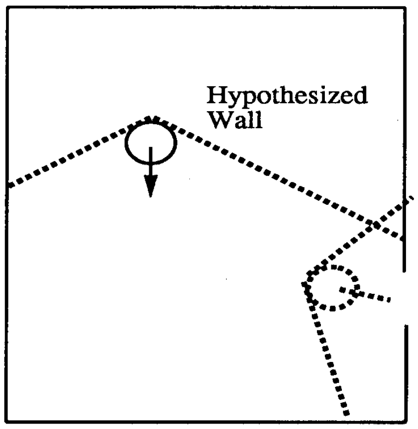

Figure 2: The open-space-explorer using the free-space-explorer.

The basic idea is to hypothesize an oblique wall, immediately behind the robot (as shown in Figure 2) and to update the actual range measurements with the synthetic range readings reflected from this hypothesized wall. Then, with these updated range readings, a new robot location is computed by the algorithm of the free-space-explorer as described in Section 3.1.

How can it be determined whether the robot goes through a doorway or not? For example, Gat proposed a "squeezing" behavior for the robot to wiggle through a very narrow doorway [4]. However, this method only works when the robot knows that it is approaching a doorway. Our approach is to use high-level heuristics about the environment as well as low-level sensor information. First, it computes the change of range readings from the left and right side sonars. When the robot passes through a doorway, large fluctuations in both the left and right sonar readings must be observed. When this large fluctuation is detected, it checks the existence of an intersection or a wall to confirm the completion of the behavior.

4 Vision-Based Behaviors

In most examples of indoor mobile robot navigation, one of the most important capabilities is to recognize some features, such as intersections and walls, and then track them to guide the robot - target tracking.

The target-tracker extracts two kinds of information from an input color image: a steering angle directing the robot towards a target and verification of the tracking accuracy [10]. The robot normally controls its steering, merely by using the steering angle computed from the tracker. However, this steering angle includes some errors, because of inaccurate motion of the robot. The tracking trajectory is used to verify whether the robot exactly moves toward the target or not. If the robot moves exactly toward the center between targets, the trajectories of these two targets on the image will diverge from the image center to the image boundary. Using these lines, the tracker checks whether the robot accurately follows the commands or not. When both targets disappear from an image frame at the same time, the tracker assumes that its mission has been completed. If only one target disappears, the tracker commands the robot to turn in the direction of this target, until the target is again visible. Then it repeats the same procedure until it accomplishes the tracking.

In the following sections, we describe two tracking behaviors: wall tracking and intersection tracking.

4.1 Wall-Tracker

Part of an indoor environment can usually be described by sets of parallel 3-D lines (e.g., walls and corridors). In the image, they form a vanishing point with associated lines [7].

We can extract vanishing points by using the fact that a set of parallel lines in the world intersect at a vanishing point in the image, as shown in Figure 3. However, the image lines will never meet at one point

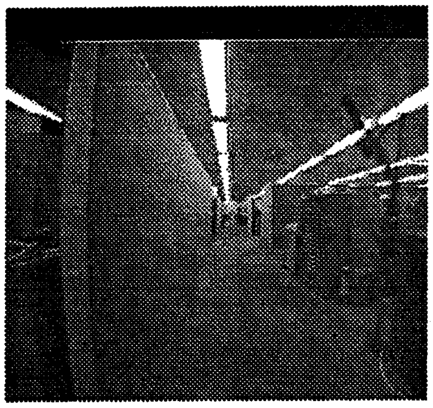

Figure 3: A typical indoor scene.

because of image noise and the errors in localizing lines in the image. We represent the uncertainty in edge location using a 1-D Gaussian distribution along the direction perpendicular to each edge line. Therefore, the search for vanishing points can be easily accomplished by finding a small neighborhood in the image plane intersected by a sufficient number of straight lines. An example of the extracted vanishing point for Figure 3 is shown in Figure 4.

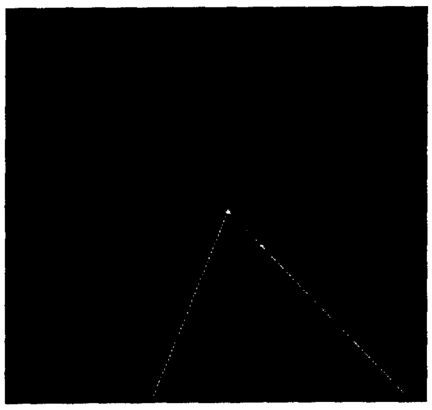

Figure 4: Extracted vanishing point and the associated lines.

Since this vanishing point specifies the 3-D orientation of the parallel lines, a mobile robot can be aligned with the parallel lines using this vanishing point. The relation between the camera and world coordinate systems is specified by the image coordinates of this van-

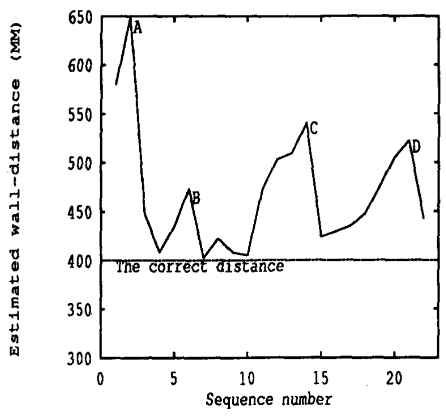

Figure 5: The computed distances from the wall by wall-following behavior.

ishing point [9]:

$$\mathbf{p_V} = (x_v, y_v) = (f\frac{\tan\alpha}{\cos\theta}, f\tan\theta) \qquad (11)$$

where f is the focal length of the camera, and θ and α indicate the pan and tilt angle of the camera with respect to the world coordinate system. Therefore, the angle needed to align the robot with the wall - steering angle - becomes:

$$\theta = \arctan\frac{y_v}{f}. \qquad (12)$$

We have done a series of navigation experiments in a real office environment, shown in Figure 3. Using the extracted vanishing point, the wall-following behavior commanded the robot to move along the corridor. Figure 5 shows the correct distance (i.e., half of the corridor width) and the computed wall distances by wall-following behavior. When the computed distance shows a large discrepancy with respect to the correct distance (e.g., at the points A, B, C, and D in the figure), the behavior controls the robot to the center of the corridor. The robot navigated a distance of 10 meters along a narrow corridor without any failure in repeated trials.

4.2 Intersection-Tracker

In an indoor environment, when a behavior-based robot fails to accomplish its mission, such as finding

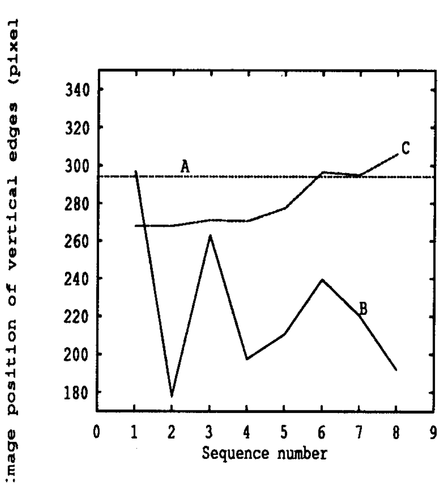

Figure 6: The average column value of two vertical edges, tracked by intersection tracking behavior.

a specific target, it may be a better strategy to find and move to a nearest intersection rather than wandering around.

In the present work, we use a simple heuristic - intersections exist at the end of walls - to extract an intersection. Vertical edges on the walls are candidates of intersections. The algorithm works in the following steps:

1. First compute a vanishing point;
2. Find two corresponding straight lines on the floor, which form the vanishing point;
3. Extract vertical lines on each straight line, located at approximately the same distance from the robot;
4. Track the two vertical lines by a real-time tracker.

The steering angle of the robot is determined by

$$\tan\theta = \frac{C_{avg} - C_i}{f} = \frac{\Delta C}{f} \tag{13}$$

where f indicates the focal length, C_i is the column value of the image center, and C_{avg} is the average column value of two vertical edges.

Figure 6 shows the computed average column value of two vertical edges from the experiments. The straight line A represents the column value of the correct image center. The curve B indicates the average column position of two vertical edges before the robot moves. The discrepancies between A and B, corresponding to ΔC, are used in the intersection-tracking behavior to control the robot. The curve C shows the resulting average column positions, corrected by the intersection-tracking behavior. Under ideal conditions, we should not observe any errors between the correct image center (the line A) and the corrected average column position of the two vertical edges (the curve C). However, when vertical edges are distant (e.g., 10 meters from the robot) in the beginning of the experiment, large errors (30 pixels) are observed. As the robot approaches the vertical edges, the error is reduced to less than 5 pixels.

5 Integration

To be a useful system in a complex indoor environment, the robot must be able to carry out a global mission by selecting an appropriate behavior in sequence among many available behaviors. The integration of multiple behaviors into such a coherent system raises three important issues: the representation of a global mission and behavior arbitration.

5.1 Mission interface

Any typical indoor environment (e.g., an office building) can be represented by a combination of rooms, corridors, and intersections [6]. Based on this fact, we observe that the robot can demonstrate a goal-directed navigation capability only with three target tracking behaviors: doorway tracking, wall tracking, and intersection tracking behavior. Therefore, we can represent a global navigation mission as a sequence of these three tracking behaviors. Currently, this sequence is generated by a human operator. Using common sense-like rules, a list of adaptive-level candidate behaviors is also generated for the recovery of the failure of each target tracking behavior. For example, when the robot is in a room and a doorway tracking behavior fails, the open-space-explorer is the best alternative behavior to get out of the room. If the open-space-explorer is not activated or fails, the wall-follower provides the behavior to get out of the room by following the walls of the room.

5.2 Behavior arbitration and failure recovery

There are two kinds of behavior arbitration in our system: arbitration between different-level groups and that inside each group [11]. We use the fixed priority order among three groups: A behavior from the reflexive-level group has the highest priority because of the safety of the robot except in the case of deadlock situation. For efficient goal-directed navigation, the purposive-level group has higher priority than the adaptive-level while tracking or searching is successfully executed. If a failure of the purposive-level behavior or a deadlock situation occurs, an appropriate behavior from the adaptive-level group is selected from the specified list of behaviors.

Different behavior arbitration mechanisms for the reflexive-level, purposive-level, and adaptive-level work in parallel to determine the motion command. For the reflexive-level group, priority-based arbitration is adopted according to the reliability of each sensor. If the behavior from an agent with higher priority emerges, others from agents with lower priority are ignored.

For the purposive-level group, the priority of each paired target tracker and target searcher is determined according to the mission. This arbitration is done by the sequence of tracking behaviors to be executed.

The adaptive-level behavior can be invoked either when a target tracking behavior failed or a deadlock situation occurred. For the failure of the tracker, an appropriate behavior is selected from the list of adaptive behaviors of the current tracking behavior. Detecting the failure of the target tracking behavior is done by comparing the motion command from the target tracker with that from the best adaptive-level behavior.

For a deadlock situation, the arbitration of the adaptive-level behavior can be easily done if the robot can detect the deadlock situation. For the detection of the deadlock situation, work is still under progress.

6 Experimental Results

We have carried out a series of real navigation experiments in an unmodified office environment, using BIRDIE, which is equipped with 18 ultrasonic sensors and a color camera [11].

Missions for the robot are specified as a sequence of behaviors to be accomplished: (1) go through a doorway, (2) turn to the left after passing a doorway, and (3) go to an intersection by following a long flat wall on the left-hand-side. The motion executor monitors all the behaviors to confirm whether or not this sequence of behaviors is accomplished. Work for a more general mission level interface is still underway.

Figure 7 shows a diagram of an experiment in a real office environment. The robot should accomplish the mission, given as a sequence of behaviors, without colliding with any obstacles. In this experiment, five behaviors (obstacle-avoider, open-space-explorer, obstacle-boundary-follower, intersection-tracker, intersection-searcher) were working in parallel.

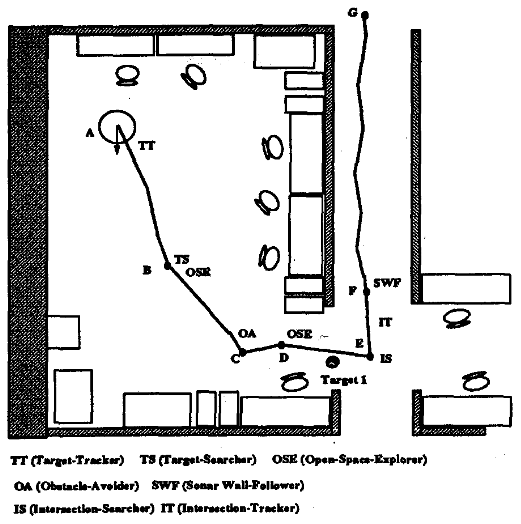
G
A
TT
B
TS
OSE
F
SWF
IT
OA
OSE
E
C
D
IS
Target 1
TT (Target-Tracker) TS (Target-Searcher) OSE (Open-Space-Explorer)
OA (Obstacle-Avoider) SWF (Sonar Wall-Follower)
IS (Intersection-Searcher) IT (Intersection-Tracker)

Figure 7: An experimental result from a real mobile robot navigation.

At starting position, A, the robot found and tracked a red target, $Target1$, located at a doorway to simulate the doorway tracking behavior. The sonar-based open-space-explorer confirmed the resulting motion command from the target-tracker. While the robot was approaching B, $Target1$ was removed and the target-tracker failed to track the target. The motion-executor immediately detected this failure from the discrepancy between the two redundant behaviors, the simulated doorway-tracker and the sonar-based open-space-explorer. Since the target could not be found by the target-searcher, the motion-executor used an actuator command computed from the most appropriate adaptive-level behavior, the open-space-explorer, to reach C. At C, however, the obstacle-avoider detected a static obstacle and commanded the robot to D. The motion-executor again used a command computed from the open-space-explorer, because the first goal, going through open space, was not yet accomplished. When the robot passed through the doorway, from D to E, the successful accomplishment of the first goal was confirmed. At E, the intersection-searcher used the vanishing point to extract a candidate of intersection on the left hand side of the robot. From E, two redundant behaviors, such as the intersection-tracker and the sonar-based obstacle-boundary-follower, were working in parallel. From E to F, however, the intersection-tracker commanded the robot since there was no wall for the sonar-based obstacle-boundary-follower to follow. From F, the two redundant behaviors commanded the robot to move along a wall until it reached an intersection G. The accomplishment of the second goal was confirmed when both targets disappear from an image frame at the same time.

7 Conclusions

We have demonstrated that a navigation system, consisting of a few basic behaviors, along with a proper behavior selection mechanism can provide sufficient navigation capability for a mobile robot working in indoor environments. The three-clustered system architecture with the redundant adaptive-level behaviors have improved the robustness and flexibility of the behavior-based system by providing an efficient mechanism to recover the failure of the purposive-level behaviors.

We have also demonstrated that Range-based behaviors, using an efficient Energy-based motion decision algorithm, are very robust and flexible for mobile robot navigation in unmodified office environment, through real world experiments. From our preliminary experimental results, we have also found that vision-based behaviors using vanishing points are robust against errors of preprocessing (e.g., edge detection) and image noise. We are now developing other vision-based behaviors such as obstacle-avoidance and target-approaching. The obstacle-avoider computes the surface orientation and the time to contact from the image velocity field [3].

Work is currently underway to improve our mobile robot, BIRDIE, by adding other sensors, such as infrared sensors, touch sensors, and a digital compass.

References

[1] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2, April 1986.

[2] R. A. Brooks. Intelligence without reason. Technical report, Artificial Intelligence Lab., MIT, 1991.

[3] R. Cipolla. *Active Visual Inference of Surface Shape.* PhD thesis, Dept. of Engineering Science, Univ. of Oxford, 1991.

[4] E. Gat. Robust low-computation sensor-driven control for task-directed navigation. In *IEEE Robotics and Automation, Scramento*, 1991.

[5] Y. Goto and A. Stentz. The cmu system for for mobile robot navigation. In *IEEE Robotics and Automation*, March 1987.

[6] M. Habib and S. Yuta. Efficient navigation system consistution for indoor autonomous mobile robot. In *International Symposium on Advanced Robot Technology*, 1991.

[7] Kanatani. Reconstruction of consistent shape from inconsistent data. *Int. J. Computer Vision*, 3, March 1989.

[8] Kass, A. Witkin, and Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1, 1987.

[9] I. Kweon, K. Onoguchi, M. Watanabe, and Y. Kuno. Vision-based behaviors for indoor mobile robots. In *Information Processing Society of Japan (IPSJ) Workshop on Computer Vision*, 1992.

[10] I. Kweon, M. Watanabe, K. Onoguchi, and Y. Kuno. Behavior-based mobile robots using multiple sensors. In *The First Korea-Japan Joint Conference on Computer Vision*, 1991.

[11] M. Watanabe, K. Onoguchi, I. Kweon, and Y. Kuno. Architecture of behavior-based mobile robot in dynamic environment. In *IEEE Robotics and Automation*, 1992.