Multi-touch Vector Field Operation for Navigating Multiple Mobile Robots

Jun Kato
The University of Tokyo, Tokyo, Japan
jun.kato@ui.is.s.u-tokyo.ac.jp

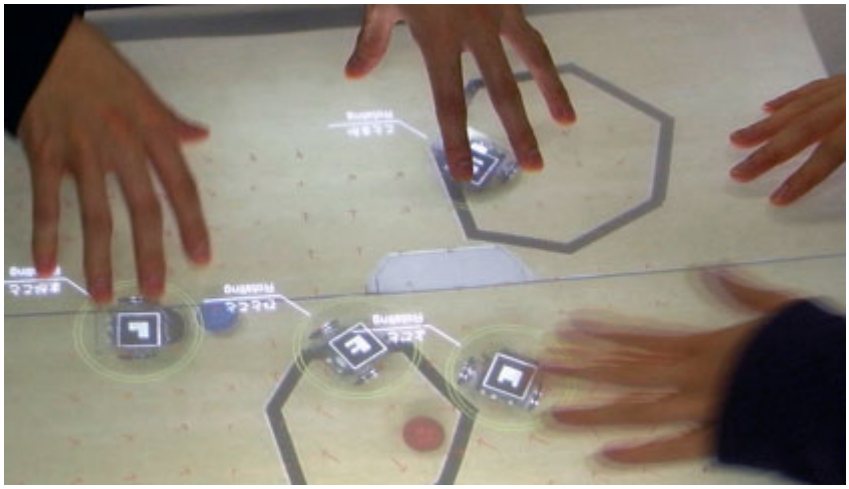

Figure.1: Users can easily control movements of multiple robots simultaneously.

Abstract

Existing user interfaces for controlling mobile robots in real-time are generally for only one robot. When we use these interfaces for multiple robots, we must conduct commands for each of them. We present a multi-touch interface with a top-down view from a ceiling-mounted camera for controlling multiple mobile robots simultaneously. Users touch and move their hands on the screen to specify the 2-dimensional vector field, and all the robots move along the vector field. We also combined the vector field with a collision avoidance method using a potential field. An informal user test is conducted to compare characteristics of our vector field control and a traditional interface of specifying waypoints of individual robot. The test users successfully operated multiple robots simultaneously with our interface. The results suggested that the combination of our interface and other user

interfaces for robot navigation is more promising, and more sophisticated visualization is desired.

1.Problem and Motivation

Using multiple robots simultaneously to handle tasks is desirable because they can do various tasks with greater efficiency than a single robot. They can be useful for practical purposes like urban search and rescue. It might be also very attractive if small multiple robots work together to help daily chores at home or to entertain people in amusement parks. However, most of the existing user interfaces for mobile robots focused on controlling one robot and are not designed to allow users to control multiple robots simultaneously. With this kind of interfaces, the users must command robots for many times and are often required to switch among views of each robot.

Our core challenge is to design a user interface which allows real-time simultaneous navigation of multiple mobile robots without switching operation modes. In this paper, we present an intuitive interface using a multi-touch display as shown in Figure 1. The display shows a top-down view from a ceiling-mounted camera in real time. The view is overlaid with a 2-dimensional vector field, and all the robots move in accordance with the directions of the vectors around them. Users can manipulate the vector field by touching and moving their hands on the display. We also combined the vector field with a traditional collision avoidance method which is described in detail later.

2.Background and Related Work

2.1.User Interfaces for Robots

Previous studies have tried to utilize robots by giving them intelligence. Wang et al. conducted experiments on and compared the performance in accomplishing tasks by robot teams with different degrees of autonomy [1]. Their results show that a completely autonomous approach is currently not yet feasible. Driewer et al. describe what the user interface in human-robot teams should be like [2]. They pointed out that in teams consisting of people, robots, and their supervisor, the use of graphical user interfaces (GUI) greatly affects the performance of their tasks. On the whole, user interfaces play a considerable role in the cooperation between people and multiple robots. Thus, more advanced user interfaces are needed for effective cooperation between people and robots.

Single robots are normally controlled with joysticks, keyboards, and other pointing devices. However, with advances in robotics, a wider variety of user interfaces for these purposes have been developed. For example, multimodal interfaces, such as a combination of hand gestures and speech for an assistant robot [3] and a portable interface using a personal digital assistant for mobile robots [4], have been proposed. These systems enable users to navigate a robot with waypoints projected on the screen. Recently, Kemp et al. proposed an intuitive interface [5] in which motor-impaired users use laser pointers and a touch-screen to control a robot in grabbing things.

There are fewer studies about user interfaces for managing a team of robots. Skubic et al. proposed a way to directly command a team of robots by drawing a sketch of the environment from a downward viewpoint [6]. While they conducted a user test with three robots, they did not consider a greater number of robots. McLurkin et al. designed hardware infrastructure and software for a swarm (hundreds) of robots that can operate largely without physical interaction [7]. Their study is mainly focused on user output, and their GUI inspired by real-time strategy video games is experimental. They say much future work is required for human-swarm interaction.

2.2.Multi-touch User Interfaces

Concept of multi-touch technology is dating back to the late 1990s, e.g., the HoloWall by Matsushita and Rekimoto [8], but it has become popular through commercialization and the drop in the cost of the technology. The iPod Touch by Apple is arguably the most affordable product with a multi-touch display, and Windows 7 by Microsoft officially declares support for multi-touch display devices. Tabletop systems with multi-touch capability are now provided by many companies. As for researchers, Han proposed a way to make a low-cost tabletop multi-touch display [10]. Wilson proposed a multi-touch table system and an interaction technique using the optical flow of hand motions, in which users can pan, rotate and zoom a map by moving their hands on the table [11]. Wilson et al. combined physical simulation to help users to manipulate virtual objects using a multi-touch display [12]. Characteristics of a multi-touch interface is often described by its capability to enable gestures with multiple fingers, but its possibility as a user interface tool comes from its capability of detecting shapes of contact surfaces.

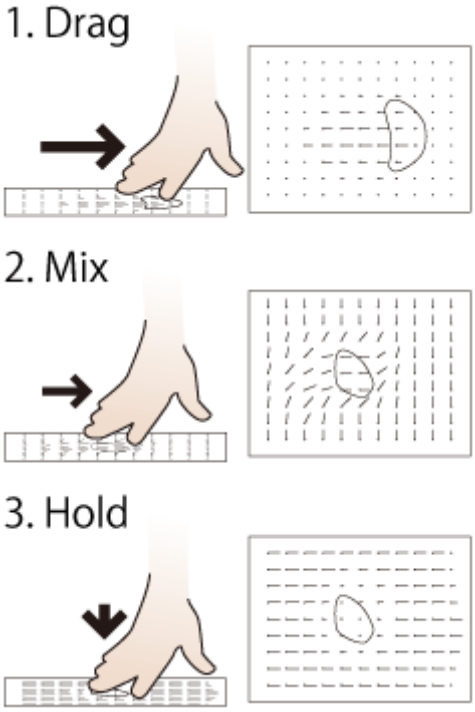

Figure.2: Available operations

3.Uniqueness of the Approach

3.1.Our user interface

We used a multi-touch table for the robot control. The table shows the video image of robots moving on a floor captured from a ceiling-mounted camera. In this way, users can see the overall situation at a glance. Icons for detected robots are overlaid on real images of them with a brief description (their names and statuses, e.g., "Stopped", "Rotating", or "Moving forward"). We overlaid a virtual flow field on the top-down view. This flow field is used to command the robot associated with a specific location in the field. The user manipulates the flow field, and the robot moves in accordance with the flow. The flow field is stored as vectors specified at 2-dimensional grid points. When the user touches his hand to the display and moves that hand, the vector values stored at nearby grid points are updated. The direction in which each robot will go is determined by the sum of vectors at nearby grid points.

In our system, all the vectors gradually decrease over time. In other words, all streams made by the user's hands gradually disappear. Operations available in the current

implementation are as follows (Figure 2). Please note that the users do not need explicit operation to switch between these three modes.

Drag
: When users touch and drag their hands along the panel, a virtual stream appears on the vector field and robots move in accordance with the stream.

Mix
: When users drag their fingers on existing streams, the vector data near the streams are blended in proportion to their distances from the streams. Nearer streams affect the vectors more strongly.

Hold
: Touching the panel without further motion will reset the vectors under that hand. Thus, we can stop a robot by touching and holding an area in front of the robot.

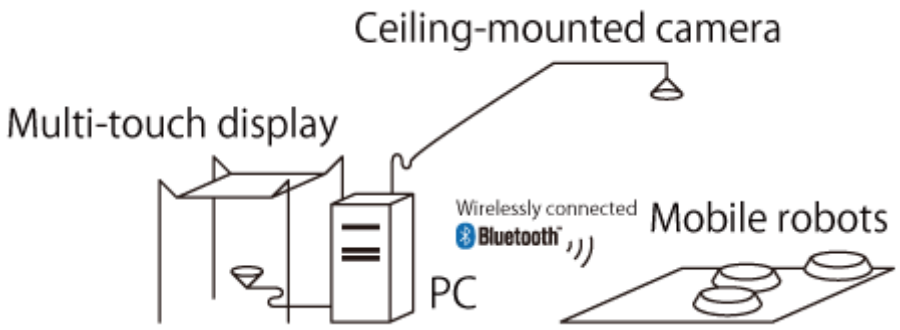

Figure.3: System overview

4.2.Implementation

4.2.1.Hardware and Back-end System

The system overview is shown in Figure 3. The multi-touch interface is based on the low-cost method proposed by Han [9], where the frustrated total internal reflection of infrared light in an acrylic panel is used. The shapes of the surfaces being touched are observed as luminous blobs by an infrared camera set under the panel.

We used Roomba robots by the iRobot Corporation and proprietary small mobile robots developed in our project. A Roomba is 34 cm in diameter, while the small robot is 10 cm in diameter. Both robots have two wheels and basic location capabilities. They can move

forward and backward and also spin. Both robots are remotely controlled by a host computer via a Bluetooth connection.

A camera is mounted on the ceiling. It is approximately 190 cm from the floor when using Roombas and about 120 cm in the case of our small robots. It provides a top-down view of the real environment. The images are captured at 30 fps. The positions of the robots are calculated through detecting fiducial markers attached to their top surface in the captured images using the Java implementation of ARToolKit [13]. The markers are 11.5-cm squares when using Roombas and 5-cm squares in the case of our small robots.

The back-end system was built using the Java platform, which works on both the Mac OSX and Windows. Bluetooth links were established with the JSR-82 implementation. Every time the positions of robots are updated, the system calculates how the robots should move, and if needed, commands are wirelessly sent to the robots.

Uniqueness of our system setup is that the captured images work as both an intuitive interface for users and a global sensor for the robots.

4.2.2.Vector Field Operation

The screen with 800 × 600 pixel resolution is divided into a predefined set of grids, and each grid holds 2D vector data. In the tested environment, the grid interval was 46 pixels. Every time the luminous blob information is updated, the motion of the blobs is tracked using optical flow. When luminous blobs are detected, the system finds the nearest luminous blob in the previous captured image. If a blob nearer than a defined threshold value is found, the new and old blobs are recognized as the same and are viewed as a continuously moving blob. Otherwise, the new blob is neglected at that time, but its information will be used the next time as an old blob. Tracked motion affects the existing vector field: grids directly under the touched surface are completely overwritten with the motion vector, and those near the surface are blended with the vector in proportion to the distance from the center of the surface. Grids farther than a predefined distance (92 pixels) are not affected. All vector data are shortened to a predefined rate (98%) every time the information from the camera is updated. Thus, after being neglected for a while, the vector field will gradually return to the initial state.

The direction of the robot movement is determined by summing the vector values at nearby grid points. We use the same weighting scheme as that for updating the vector field in accordance with the movement of the luminous blobs. That is, nearer grid points strongly

affect but farther ones weakly affect the movement of the robot in proportion to the distance from the robot. Grid points farther than a certain threshold are ignored. The robot stops moving and starts spinning when the target orientation defined by the vector field and the current orientation is larger than a defined threshold (10 degrees). Otherwise, the robot keeps moving at a speed proportional to the length of the vector.

"Drag" and "Mix" operations are naturally achieved using the explained algorithm. In addition, when users simply touch the surface and do not move their hands, the corresponding motionless luminous blobs are detected continuously in the touched areas. These blobs move very little, so the grids near those areas hold almost zero vectors. As a result, robots in those areas stop. This is how the "Hold" operation works. Three vector field operations are implemented using a single simple algorithm.

4.2.3.Potential Field Integration

The algorithm explained in the previous subsection can be easily combined with a collision avoidance method based on the virtual force field (VFF) concept [14]. The system represents the potential field around a robot using the vector field radiating from the robot. The radiated vectors from a robot affect nearby robots. These vectors are summed with the original flow vector field specified by the user. As a result, robots move in accordance with the users' intention while avoiding collision with other robots.

This combination of autonomous algorithm and user input is achieved by not only visualizing raw data for movements of robots but also allowing its manipulation by using rich input information from a multi-touch display. Even when the mobile robots are basically commanded by the autonomous algorithm, our interface can be helpful when the robots are in trouble like being stuck with obstacles or other robots and the situation cannot be solved by the algorithms.

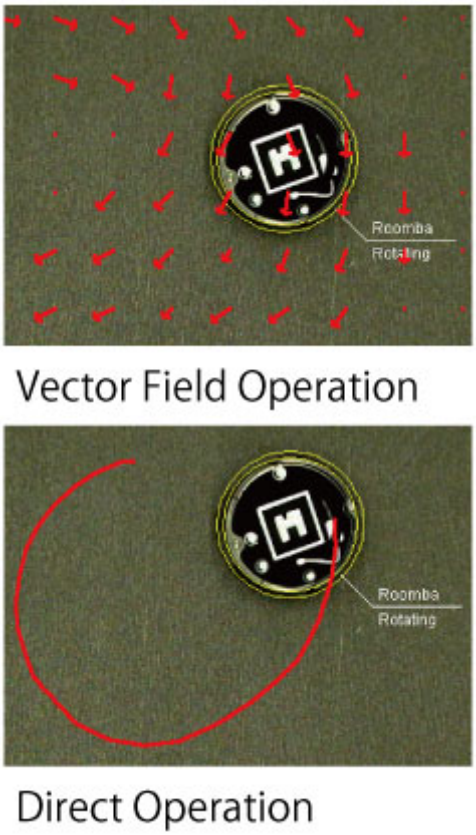

Figure.4: Screenshots of the two operating modes

5.Results

5.1.User Test

5.1.1.Design and Method

We ran an informal user test with our interface and proprietary small robots. The aim of this test was to verify the usefulness of our interface, to clarify its characteristics compared with traditional interface, and to get insights for our future work. The test compared our vector field operation (V) with direct operation (D), in which the user touches a robot's icon on the display and draws a path with his or her finger to specify the desired path for the robot (Figure 4). The collision avoidance based on VFF was turned off during the user test.

Four users, all new to the multi-touch display, participated in the study. They were first given time to practice and become familiar with the multi-touch display. After this, they

were asked to try to accomplish three tasks with two operating modes. Through these tasks, each user was requested to control four robots simultaneously in real time. Each task was attempted twice by each user, once with V and once with D. Two users tried D first then V, while the other two tried V first then D. They were not allowed to use both modes in one trial. The first task was to gather robots that were placed randomly as the initial state. All robots were required to go to an area at the corner of the field. The second was to move robots from one side to the opposite side of a field that had an obstacle at the center. The robots were lined up on the left side as the initial state. In this task, users were required to make the robots avoid the obstacle and go to the other side. The third and final task was to make all the robots push a box together. A box was set almost at the center of the field, and the robots were lined up as in the second task. The box was heavy enough to require at least three robots to move it.

5.1.2.Results of the Test

For the first task, two of the four users preferred D and the other two V. At first glance, the users seemed to have different opinions. However, after reading the comments about why they chose their answers, all of their impressions were found to be consistent. Those who said D was good explained that when all the robots neared the goal area, the formation of the robots had to be designed to avoid robots colliding with each other and that moving the robots into a formation required precise operation of each robot, which could only be accomplished with D. Those who answered V said that making streams that poured to the destination area was very intuitive and useful. They did not care about the formation of the final state. These results clarify the characteristics of the two operating modes: V is suitable for rough instructions that allow for a large margin of error while D can be used to conduct delicate maneuvers.

In the case of the second task, all users preferred V. They stated that with V, only one stream had to be made around the obstacle, while with D, similar waypoints had to be specified a number of times to control all the robots.

The task to make the robots push a box together was apparently the most difficult among all the tasks, and three of the users said that D was better. Their preference for D is easily understood by the result from the first question. D can move each robot precisely, so users can try to make robots push one side of the box equally. As a result, they could push the center of gravity of the box and were able to complete the task. With V, they failed to push the center of gravity and thus inadvertently rotated the box many times. Despite this, one

user answered that V was better. He indeed succeeded in pushing the box from one side of the field to the opposite side by drawing strokes on the display. His operation made a successful stream on the field, and the robots followed the stream, pushing the box simultaneously. He analyzed his success and said that he might have succeeded because he carefully planned where and how to draw a stream to foresee the movement of the robots.

The users said that while they were using V in the test, they often wanted one specific robot to move in a certain way. In addition, some users were observed to give redundant commands to the robots. They repeated their actions in order to ensure that their directions were accepted by the system.

5.2.Discussion and Future Work

The user test revealed important characteristics of the two operating methods: V is good for roughly controlling multiple robots, while D is good for giving precise instructions to one robot. When all the robots are expected to accomplish a unique task as a swarm and each robot is not required to do a specific task, V exceeds D in performance. Since the amount of operation required by D increases in proportion to the size of the swarm, this difference becomes clearer when the size increases.

According to the demand by the users, the system can be mainly based on V, with an optional function of D activated only when the user starts dragging the robot icons. Besides integration with D, there are many possible interfaces which can be integrated with our interface. For example, the users can draw or clear virtual walls on the field. Virtual dog icons which can be dragged by the users to make many robots as sheep run away from the icons would be convenient. Binding relative positions of robots as if they were connected with physical ropes would reduce mental load of the users in some cases.

In the aspect of visual feedbacks to the users, there are much suggested future work. The test users pointed out the importance of foreseeing movements of robots in the near future. Simulation and visualization of paths of the robots are expected. In the current implementation, users can know whether or not their commands are accepted by looking at the status text beside the robot icons. However, test users often failed to recognize this and tended to repeat commands for the robots. More noticeable and comprehensible visual feedback will help users to operate the robots with more confidence.

6.Conclusion

In this paper, we have presented an intuitive multi-touch user interface for navigating multiple mobile robots simultaneously without any explicit mode switch. Users get a top-down view of the environment which is virtually overlaid with a 2-dimensional vector field. All the robots move along the vector field. Our interface allows direct user manipulation of raw data which decide movements of robots. As a result, it can be combined with autonomous navigation algorithms whose calculation results are stored as a 2-dimensional vector field including collision avoidance method. The results of the informal user study showed that the test users were able to use our interface successfully for the given tasks. It also suggested that the combination of our interface and other user interfaces for navigating robots is more promising, and more sophisticated visualization is desired.

7.References

1. J.Wang and M. Lewis, "Human control for cooperating robot teams," *Proceedings of SIGHRI 2007*, pp. 9–16.
2. F. Driewer, M. Sauer, and K. Schilling, "Discussion of Challenges for User Interfaces in Human-Robot Teams," in *Proceedings of ECMR 2007*.
3. O. Rogalla, M. Ehrenmann, R. Zollner, R. Becher, and R. Dillmann, "Using Gesture and Speech Control for Command a Robot Assistant," in *IEEE RO-MAN'02*, pp. 25–27.
4. T. Fong, C. Thorpe, and B. Glass, "PdaDriver: A Handheld System for Remote Driving," in *IEEE International Conference on Advanced Robotics*, IEEE, 2003.
5. Y. Choi, C. Anderson, J. Glass, and C. Kemp, "Laser pointers and a touch screen: intuitive interfaces for autonomous mobile manipulation for the motor impaired," in *Proceedings of ASSETS 2008*, pp. 225–232.
6. M. Skubic, D. Anderson, S. Blisard, D. Perzanowski, and A. Schultz, "Using a hand-drawn sketch to control a team of robots," *Autonomous Robots*, vol. 22, no. 4, pp. 399–410, 2007.
7. J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt, "Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots," in *AAAI Spring Symposium*, 2006.
8. N. Matsushita and J. Rekimoto, "HoloWall: designing a finger, hand, body, and object sensitive wall," in *Proceedings of UIST 1997*, pp. 209–210.
9. J. Han, "Low-cost multi-touch sensing through frustrated total internal reflection," in *Proceedings of UIST 2005*, pp. 115–118.
10. K. Fukuchi and J. Rekimoto, "Marble Market: Bimanual Interactive Game with a Body Shape Sensor," *Lecture notes in computer science*, vol. 4740, p. 374, 2007.

11. A. Wilson, "PlayAnywhere: a compact interactive tabletop projection-vision system," in *Proceedings of UIST 2005*.
12. A. Wilson, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. Kirk, "Bringing physics to the surface," in *Proceedings of UIST 2008*.
13. H. Kato, "ARToolKit : Library for Vision-based Augmented Reality," *Technical report of IEICE. PRMU*, vol. 101, no. 652, pp. 79–86, 2002.
14. J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.