A Sketch-Based Interface for Multi-Robot Formations

Marjorie Skubic, Derek Anderson, Mohammed Khalilia and Srikanth Kavirayani

Computational Intelligence Research Laboratory
University of Missouri-Columbia, Columbia, MO 65211
skubicm@missouri.edu

Abstract

In this paper, we describe a sketch-based interface to control a team of mobile robots. The sketch interface uses a PDA platform and incorporates editing commands and an HMM recognizer for classifying sketched symbols. In the paper, we describe the recognizer and report classification results on twelve sketched symbols. To control the robot team, a user sketches a configuration that represents a multi-robot formation, such as follow the leader or march side by side. A team of small mobile robots has been built to test the PDA interface. The robots achieve the formations through simple behaviors that use local color tracking. Each robot is fitted with brightly colored side panels to facilitate the formations. In the paper, we report observers' comments from the AAAI Conference and Robotics Competition.

Introduction

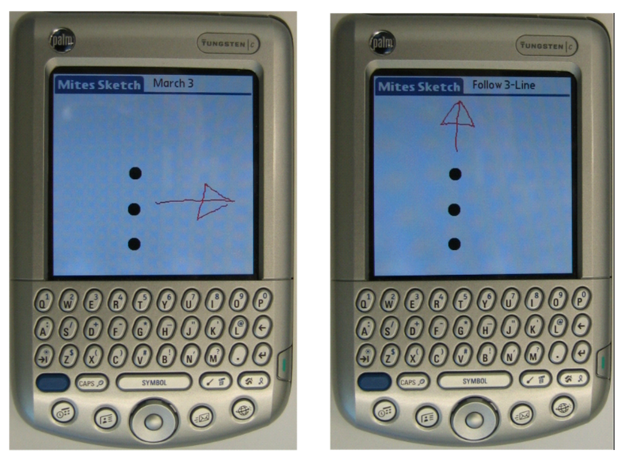
We have been investigating sketch-based interfaces in an effort to achieve intuitive interaction for controlling and communicating with one or more robots. Sketching is a natural medium for spatially-oriented applications, and directing mobile robot navigation is inherently spatially oriented. A handheld Personal Digital Assistant (PDA) is a convenient platform for in-the-field collaboration with mobile robots and especially when the user must change locations and thus, cannot be confined to a desk. Here, we have combined a sketch-based interface with a handheld PDA. We report our experience with a prototype implementation of a sketch-based interface to control a team of mobile robots.

PDA interfaces have been proposed previously for controlling mobile robots. Fong has used a PDA as an intelligent teleoperation interface (Fong, 2001). Basic controls are included for commanding the robot velocity. In addition, the user can select a live image of the scene and can designate via points on the scene to specify a robot path. The interface has been applied to a team of 2 robots, although each robot is controlled individually (Fong, Thorpe & Baur, 2003).

A PDA can also be used with a map of the environment. Perzanowski et al. (2001) and Lundberg et al. (2003) have developed such interfaces in which a user can sketch moves on top of a map, which has been acquired using the robot's sensors. Kawamura et al. (2002) have developed a PDA interface in which the user sketches a map of the environment and places artificial landmarks on the sketch to represent artificial landmarks in the physical environment. The user then sketches a route with respect to the artificial landmarks.

In our previous PDA work, the user sketches a route map that is an approximate representation of the environment (qualitatively accurate but not quantitatively accurate) and then sketches a route through the sketched map. Landmarks are automatically extracted from the sketched environment for key turning points, and a topological route representation is sent to the robot for navigation (Skubic, Bailey & Chronis, 2003; Chronis & Skubic, 2004).

In contrast, the work reported here uses a PDA sketch-based interface to send formation commands to a team of mobile robots. Example PDA sketches are shown in Figure 1. The sketch in Figure 1(a) is translated into a march side by side formation, as depicted by the arrows drawn perpendicular to the row of robot icons (shown as solid black circles). The sketch in Figure 1(b) is translated into a follow the leader formation, as the arrow is parallel to the robot row. In each case, the sketch is translated into a command that is transmitted to the robot team.



(a) (b)

Figure 1. Example sketches (a) Three robots in a march side by side formation (b) Robots in a follow the leader formation.

In this paper, we discuss the sketch understanding interface and the robot design used to achieve these formations. In Section 2, we discuss methods used to process the PDA sketch. Section 3 includes a discussion of the robot design, as well as the method for achieving formations using local sensory information. In Section 4, we discuss the performance of the system and audience response at the AAAI 2004 Conference and Robotics Competition. Concluding remarks are given in Section 5.

Sketch Understanding Methods

The sketch interface described here is based partially on our previous sketch interface (Skubic, Bailey & Chronis, 2003; Bailey, 2003) but also incorporates a new symbol recognizer based on Hidden Markov Models (HMM). The symbol recognizer is described briefly below; see also (Anderson, Bailey & Skubic, 2004) for more detail.

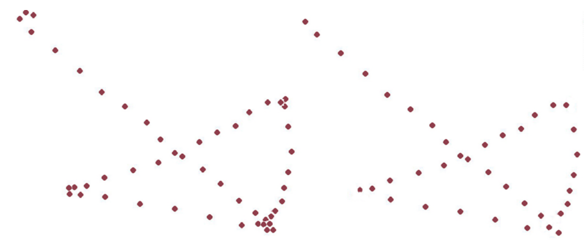
HMM Symbol Recognizer

As a symbol is sketched, the sequence of pixel coordinates, which correspond to the two dimensional locations of the pen over time, are recorded. This temporal unfolding of the symbol in terms of pixel coordinates can be viewed as a form of gesture recognition. The goal of the system is to learn a set of model parameters that encapsulate discriminate temporal features. These trained models are finite and can be transferred to a PDA to generate the online likelihood that some future observation, a new sequence of pixels, was generated from some particular known model.

The steps of the sketch-based symbol recognition include (1) the capture of pixel coordinates, (2) pre-processing of pixel data, (3) the extraction of features from the temporal pixel observations, (4) the application of HMMs to classify the symbol, (5) post-processing to reduce false alarms, and (6) the final classification of a symbol into one of a known finite set. These steps are described below.

In order to reduce processing time, we perform minimal pre-processing. The first step involves the removal of consecutive identical points. The next step is a method of minimum distance pixel sampling (to down-sample), which can be done quickly as the points are being captured, illustrated in Figure 2.

Figure 2 illustrates one additional artifact that can reduce the recognition rate. As a user sketches symbols on a PDA screen, he can generate a “hook” at the beginning of the symbol[1]. The majority of our hooks were removed simply through the sequential removal of pixels within a fixed distance. The remainder of hooks were recognized and removed by identifying regions of large angular change at the beginning of a sketch.



(a) (b)

Figure 2. Fixed distance pre-processing (a) Sketched symbol before fixed distance point removal (b) Same symbol after fixed distance point removal.

Figure 3 illustrates our feature extraction procedure. If there are N observations in the symbol, then the algorithm will compute (N-6) features. For every observation after the 3rd step and before the (N-3) step, the angle between a forward and a back vector is computed, where the forward and back vectors are computed averages as shown in Figure 3. The averaging effectively smoothes the curve and minimizes sketching differences due to distortion, sketching style, and pixelization. The pixel that the feature is being computed for is not actually used in determining the angle, which has helped to maintain important information such as sharp turns.

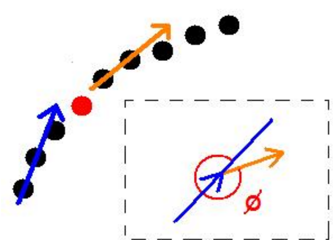


Figure 3. Seven-step, sliding window feature extraction procedure. The red pixel represents the observation step for which we are computing the present feature. The blue arrow represents the average back vector and gold arrow represents the average forward vector. The feature computed is the angle between the back and forward vectors.

Many symbols can be drawn in a variety of gesture specific sequences (e.g., clockwise vs. counterclockwise directions). Thus, each symbol may require multiple HMM models in order to capture the different gesture specific ways in which the symbol can be sketched. In the case of the arrow, a clockwise or counterclockwise sketched symbol is represented as one HMM model by computing the absolute value of the angle and restricting the feature to be within the range of –180 to 180 degrees. For the ellipse and the rectangle, we use two models each to determine a clockwise or counterclockwise gesture.

[1] A hook may also be generated at the end of the symbol; however, we did not observe this in our samples.

We have applied the traditional Baum-Welch algorithm to iteratively re-estimate the parameters for a discrete observation HMM. Model verification can be done using the conventional forward and backward procedure with appropriate scaling (Rabiner, 1989). A discrete model has limitations and can induce problems into the learning process. Discretization of the features always results in some form of information loss, but it appears to be acceptable in this application. We discuss methods to compensate for these problems by a careful selection of HMM parameters.

The angular features that we generate are mapped into 20 different discrete symbols. Note that the first symbol does not start at 0 degrees, but rather the symbol ranges from 0 - (360/N)/2 to 0 + (360/N)/2 degrees, where N=20, the number of discrete symbols as depicted in Figure 4. This strategy was used so that an approximately straight line segment would not span multiple discrete symbols.

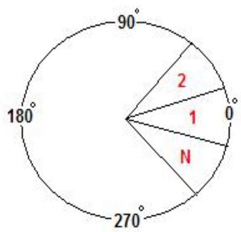


Figure 4. Discrete mapping of the angular features computed from the domain [0°, 360°] into (1 … N) discrete symbols. The numbers (3... N-1) have pictorially been left out.

We compensate for variation in the sketched symbols through selection of enough discrete observation symbols, an adequate number of training data, and an over specification in the number of HMM hidden states,. The over specification in the number of hidden states has proven to handle slight inaccuracies that can arise in the local feature extraction procedure and also help to capture reoccurring symbol variation inherent in some symbols, such as not straight but slightly curved.

For example, with little symbol variation and perfect features, there may be only two distributions (i.e. two hidden states) for an arrow symbol. One distribution represents the straight line feature, while the other corresponds to the sharp turns. We tried a two-state model initially, but the trained HMM did not yield adequate classification rates. However, we found that we could learn a model to represent this symbol if we over specified the number of hidden states. Figure 5 demonstrates our trained HMM for the arrow. We over specified the model by an additional two states. In Figure 5, the state probability density functions are depicted graphically for each state. State 1 appears to learn the straight line feature, while state 3 appears to capture the sharp angle feature. States 2 and 4 appear to learn features that correspond to subtle changes in a straight line direction, which might relate to slightly curved regions.

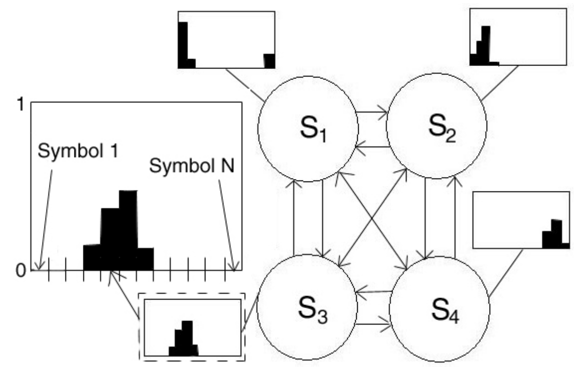
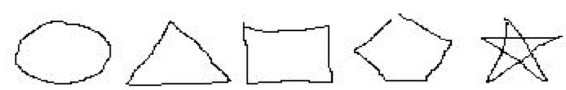
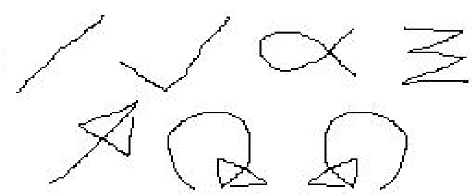


Figure 5. The trained HMM for the arrow gesture. Rectangular regions represent the state pdf's of the discrete symbols.

Finally, for post-processing, we distinguish between open and closed gestures based on the relative location of the starting and ending points. (See Figure 6 for examples of closed and open symbols.) Partially sketching a symbol can generate a likelihood value that results in a false alarm (e.g., a "c" vs. a circle). We address this problem by using the closed or open gesture criterion as a global feature.



(a)



(b)

Figure 6. (a) Closed symbols: Ellipse, Triangle, Rectangle, Pentagon and Star; (b) Open symbols: Line, Check, Cross, Delete, Arrow and Round Arrows.

The symbol recognition system has been implemented in PalmOS using C++. The HMM is trained offline using Matlab and then integrated into the PDA sketch interface.

The symbol recognizer was tested on the 12 symbols shown in Figure 6. Ten training samples of each symbol were collected from each of three users. One user was very familiar with a PDA, the other had used a PDA before and the last user had never used a PDA. These samples varied in the size, relative starting point for the gesture, orientation, and style. The recognizer was then tested using twenty separate examples from each user. Therefore, classification results are generated from 60 testing samples for each symbol. The classification results are shown in Table 1. Each classification result in Table 1 reflects the average of all models that are used to represent that one symbol. For example, the ellipse classification results are the combination of both the clockwise and counterclockwise models.

Table 1. Classification Results

Symbols	Results
Ellipse	92%
Triangle	95%
Rectangle	93%
Pentagon	93%
Star	97%
Line	98%
Check	93%
Cross	92%
Delete	89%
Arrow	98%
Round Arrow	96%

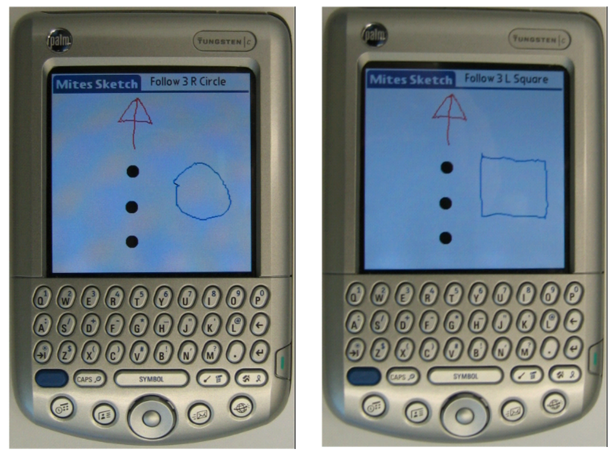
Interpreting Formations

The HMM symbol recognizer is used to identify a fixed set of control symbols which are sketched to create multi-robot formations (Figure 1). The four symbols used in the system were the ellipse, rectangle, line and arrow. A user can also sketch "blobs", which represent robots. When a robot blob is recognized, the robot icon is displayed on the PDA screen as a solid black circle.

Based on our previous work, we have also supported a set of editing capabilities to move elements (by dragging) and delete elements (by sketching an "X"). A single HMM was not used to recognize the delete command because it is the combination of two separate strokes. For the moment, our techniques can only handle the recognition of single stroke symbols. We perform the check for a delete command through the search for two consecutive lines being sketched, each one independently recognized by an HMM, and then perform a check for an intersection point. The blob, also based on previous work, is sketched as a circular "squiggle" (Skubic, Bailey, & Chronis, 2003).

The user can move and align the robot icons into configurations such as "follow the leader" and "march side by side" formations. While there are robot icons on the screen, a user can sketch an arrow; the orientation of the arrow with respect to the sketched robot icons is used to identify the robot formation. Follow the leader is identified when the robot icons are aligned in a linear formation parallel to the arrow. The march side by side formation is identified when the robot formation is perpendicular to the sketched arrow direction. Note that the row of robot icons need not be placed vertically or horizontally on the PDA screen. The sketch understanding system supports any orientation of the row and arrow and examines their relative placement to determine a formation command.

After the command to march or follow the leader has been given, there is an option to sketch additional symbols such as ellipses, rectangles, or lines to change the geometric path of the robots. For example, a sketched clockwise ellipse after a follow the leader formation (Figure 7(a)) will result in the robot leader moving in a clockwise circular path with the remaining robot team following behind. If the user then sketches a line, the robots are sent a command to follow the leader in a straight line. A sketched rectangle (Figure 7(b)) generates the square formation.



(a) (b)

Figure 7. (a) Follow the leader in a circular formation (b) The square formation.

A Prototype for Sketch-Driven Formations

Robot Design

To illustrate the proof of concept in sketch-driven formations, we designed and built a team of small, mobile robots using components from Botball kits[2]. Components included lego pieces, sensors, motors, the handyboard[3] as the micro-controller, and the CMUcam[4] for color tracking. In all, seven identical robots were built, to make two teams of three robots with one spare.

The robots have a differential drive design, with two active wheels, each controlled by a DC motor, and a third passive wheel (an omnidirectional wheel which functions

[2] www.kipr.org/products
[3] handyboard.com
[4] www-2.cs.cmu.edu/~cmucam/

as a caster wheel). Wheel encoders are included for each active wheel. With this design, the robot can be programmed to move straight ahead, straight back or turn in place by a specified number of degrees.

With a compact design, each robot measures approximately 19 cm long by 14 cm wide by 15 cm tall. As shown in Figure 8, each robot was fitted with brightly colored side panels. Each robot has an orange panel in the rear, a green panel on the right, and a blue panel on the left.

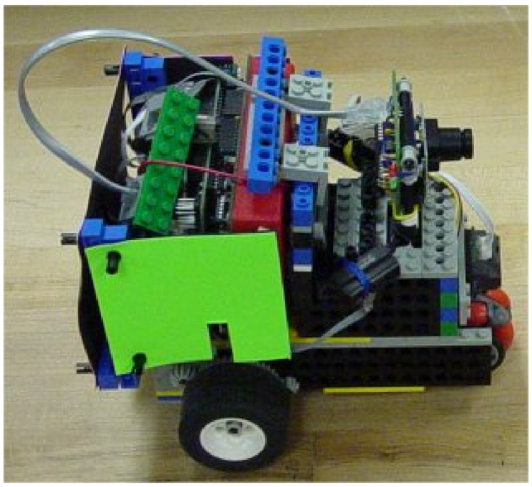


Figure 8. The robot design. Note that the handyboard is oriented vertically to make the IR receiver accessible.

Each robot is equipped with a set of sensors. There are three front infrared (IR) sensors (left-front, right-front, and directly in front) and one rear IR sensor. A CMUcam is mounted on a servo motor such that it can be rotated 180 degrees in the front for color tracking.

We also use the analog knob on each handyboard to define a default identifier (ID) number. For each team of three robots, one robot is defined with ID 1, another with ID 2, and the third with ID 3. Commands are sent to the robot team to designate the motion of one, two, or three robots, depending on the number of robot icons sketched on the PDA. A robot's response to the commands is based on its ID number. For example, if two robot icons are sketched, then robots with ID's 1 and 2 will respond; the robot with ID 3 will not move.

PDA to Robot Communication

Communication between the PDA and the robots is achieved using IR transmission. The OmniRemote™ library[5] is used on the PDA to broadcast IR commands which are received by the handyboard controlling each robot. The handyboard has an IR receiver and can be programmed to receive the equivalent IR signal from a remote control device. With the OmniRemote library, the PDA is programmed to output IR signals, as if from a remote control device. To establish this communication link, we defined a set of unique codes, one for each of the commands listed in Table 2 and trained each command on the PDA using a standard remote control device.

Commands are broadcast to all robots as a result of three processing mechanisms. First, any of the commands can be sent via a menu selection on the PDA. Second, each command can be generated using a short-cut graffiti command (or the keypad on the Tungsten PDA, shown in Figure 1). In addition, a subset of the commands can be generated whenever a sketch is interpreted; these include all of the Formation commands, as well as all of the Control commands except Stop.

Table 2. PDA to Robot Commands

Command type	Command
Control	Forward
	Reverse
	Right turn
	Left turn
	Turn around
	Stop
Initialization	Set ID from sensors
	Reset ID
Formations	Follow 2 robots
	Follow 3 robots
	March 2 robots
	March 3 robots
	Left circle
	Right circle
	Left square
	Right square

To test the sketching interface and the effective IR range for communication, the PDA software was installed on three different PDA platforms: the Handspring Visor, the Palm m505, and the Palm Tungsten. Although the Tungsten was the best to use for the sketching component due to processing speed and screen resolution, the Palm m505 had the best IR range with more than adequate processing speed.

The use of IR transmission as a communication mechanism between the PDA and the robots was undoubtedly the weakest part of the system, as direct line of sight was required and the range was limited. However, this was done for the prototype only and was not intended as a long-term solution.

Robot Actions

The robots execute specific actions in response to the commands listed in Table 2. In all cases, obstacle avoidance is incorporated as a low-level behavior, based on the four IR sensor readings. A generally forward

[5] www.pacificneotek.com/omnisw.htm

moving robot will avoid an obstacle by moving around it. A robot backing up will avoid hitting an obstacle by stopping.

The robots are programmed to respond uniquely to each of the commands listed in Table 2, based on their ID numbers. Upon receiving the Control commands, robots of all ID numbers will respond appropriately to the Right turn (turn right 90 degrees), Left turn (turn left 90 degrees), Turn around (180 degrees), and Stop. However, the forward and reverse commands are reserved for the robot with ID 1, which is considered to be the leader in a formation.

Although the analog knob on the handyboard defines a default ID, the ID numbers can also be inferred from the IR sensor readings. The Set ID command can be used to override the default ID settings when the robots are positioned in a follow the leader formation. Robot of ID l is defined as the leader; therefore, if the front IR sensors show nothing in front and the rear IR sensor shows something in the rear, the ID is set to number 1. If both the front and rear sensors indicate presence, the ID is set to 2. If the front sensors indicate presence and the rear sensor does not, then the robot is at the end of the formation, and the ID is set to 3. In addition, the Reset ID command can be used to revert back to the default ID settings.

Formations were achieved using local sensory information from the color CMUcam and the IR sensors. For example, the Follow 2 and Follow 3 commands result in the follow the leader formation shown in Figure 9. This formation was achieved by programming the follower robots (ID 2 and 3) to simply track and follow the color orange. The leader (ID 1) was programmed to move straight ahead. The front sensors on the follower robots were also used to adjust the distance between robots in the follow the leader formation, either to wait for the robot in front or to speed up.

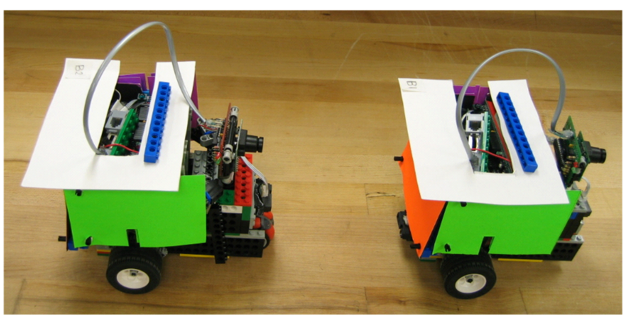


Figure 9. Two robots in a follow the leader formation.

The March 2 and March 3 commands result in the march side by side formation, shown in Figure 10. This was achieved by turning the CMUcam servo to the left on the follower robots and tracking the color green on the neighbor's side panel. The leader robot (the left-most robot in the formation) is programmed to make short "steps" forward in a march-like fashion. If a follower robot sees green, it also makes a short step forward, pausing after each step. Thus, the robot formation appears to march forward.

The follow the leader formation can be modified by the addition of a sketched ellipse symbol (Figure 7(a)), which sends the Right circle or Left circle command, depending on whether the ellipse was sketched in a clockwise or counterclockwise fashion. The leader robot will begin a right circle or left circle pattern as specified by the command. The follower robots continue to track and follow the color orange and thus follow the leader in the circular path.

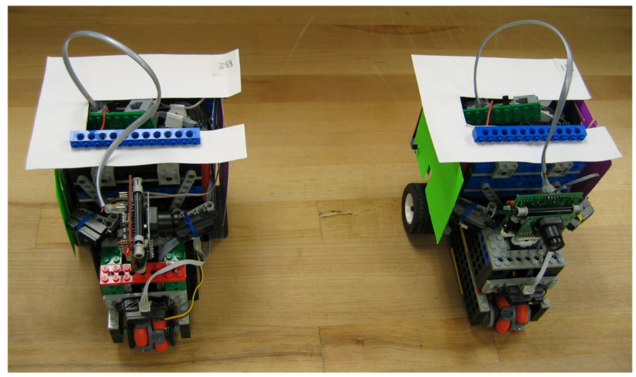


Figure 10. Two robots in a march side by side formation.

The Left square and Right square commands are generated when the rectangle symbol is sketched after either the follow the leader formation or the march side by side formation (Figure 7(b)). In either case, the robots move in a square pattern of varying sizes, based on the robot ID numbers.

A formation can be modified by sending the Turn around command. This has the effect of reversing the robot ID numbers. That is, the leader robot will become the last robot, and the last robot will become the leader. Subsequent follow or march sketches will result in the appropriate formation with the new leader. It is also possible to switch between formations, i.e., from the march formation, the robots can change to the follow the leader formation. In fact, it is possible to form a follow the leader formation from arbitrary positions, as the follower robots will search for the orange target, although the follower robots must be able to see the orange panel.

Exhibition and Interaction Results

The PDA sketch interface and robot team were demonstrated at the 2004 AAAI Conference and Robotics Competition as part of the Exhibition and Interaction events. Figure 11 shows one of the authors demonstrating the follow the leader formation. The comments and suggestions reported below are from AAAI conference attendees, as well as from Botball participants (mostly students in grades 6-12), their coaches, and their family members. The National Botball Competition was co-located with the AAAI Conference; many of the participants were especially interested in our robots, as they were built with Botball kits.

Figure 11. One of the authors demonstrates the follow the leader formation with two robots. This illustrates the typical range available for IR transmission between the PDA and the robots.

The sketch interface and robot team performed well during live demonstrations. Set up on-site was minimal; however, the local color-based behaviors required re-initialization of the color parameters at the conference site. Initially, we had made the color target range for green and orange quite broad to account for a variety of lighting conditions, but found that environment structures interfered, such as the color of the table drapes, the wall, and the neighboring yellow tent structure used by another robot team. These temporary problems were fixed by adjusting the color parameters for the orange and green targets.

We asked attendees to try the sketch interface, in an effort to (1) test the robustness of the recognizer with different users, and (2) get the general opinion of using a sketch interface, especially as a control mechanism for a team of mobile robots. Many other observers commented on the sketch-based robot system or offered suggestions.

About 18 users tried the sketch interface. Generally, the users liked the idea of sketching and wanted to do more. The recognizer robustly classified the symbols drawn. Several users wanted to draw more symbols and more complex symbols. Users also noted they would like the ability to draw symbols in a more flexible way. One user asked whether a triangle pattern could be drawn to generate a triangular follow the leader path. Another user asked whether the sketch interface would recognize an elephant and then make the robots move in an elephant-shaped path. Another user wanted to know whether the sketch interface would recognize false patterns. The issue of false alarms is important. During the exhibition, the false alarm rate was quite low. If the recognizer does not identify a stroke mark (i.e., none of the models fit), the stroke will be discarded.

Many questions and comments focused on application issues. The most asked questions were the following: "What are you going to use this for?" What do you see as applications?" "How is it useful to society?" Observers noted the potential for military applications, such as in specifying military search patterns. An observer noted that the sketch platform was a more natural method for military leaders to receive and provide war information. Observers suggested strategic applications such as planning and coordination between two or more teams. One observer suggested improving the follow the leader formation for use in a military application. Observers also noted that the sketch-based interface could be used for (non-military) search and rescue coordination. Finally, several observers suggested using the sketch interface for video games, e.g., as an interface for strategic multi-player games.

Observers also offered suggestions for improving the system. Some noted that higher performance could be achieved by using better hardware, e.g., bigger and more capable robots, and a Tablet PC instead of a PDA. The weakness of the IR communication was noted as well. The IR transmission did represent a limitation in demonstrating the system, as a result of the range and line of sight requirements. With a better communication system, more formations and more complex formations could have been included. In particular, the IR transmission scheme limited the number of robots that could reliably receive commands from the sketch interface.

Observers also commented on extending the communication capabilities, such as designating a robot to receive a command (i.e., sending commands to specified robots individually), or circling a group of robots as a way to designate the robot receivers of a command. It was also suggested that robot-to-robot communication be added, such as communication between the leader and a follower.

Observers had suggestions for extending the functionality in other ways, such as sketching a map and then incorporating the formation control with respect to the map. Another observer asked whether we had plans for more tightly coupling information which the robots receive over time, such as updating the sketch.

Finally, the most unexpected comment was from an observer that spent several minutes watching a team of two robots in a circular follow the leader formation. After observing for some time, he noted that the robots looked as if they were alive, especially in the way the follower robot searched and tracked the leader. Figure 12 shows two robots in this formation.

Figure 12. Observers watching two robots move in a circular follow the leader formation.

Concluding Remarks

In this paper, we have described a prototype implementation of a sketch-based interface for generating multi-robot formations. The interface is built on a PDA platform and allows the user to sketch desired formation configurations such as follow the leader or march side by side. The user sketches the position of robot icons and can use editing strokes to move or delete symbols. An HMM symbol recognizer is used to classify strokes. In the paper, we have reported classification results on twelve symbols using the HMM recognizer.

The sketch-based interface was demonstrated at the AAAI Conference and Robotics Competition as part of the Exhibition and Interaction events. In the paper, we report the performance, as well as comments and suggestions offered by observers.

In the future, we plan to extend the system by porting the sketch interface to a Tablet PC and using larger robots such as the Pioneer from ActivMedia. The wireless communication capabilities of these platforms will facilitate a larger robot team and more complex formations. We also plan to extend the interface to support more symbols, more complex symbols (including multi-stroke symbols), and more complex robot coordination and control. We agree with observers that the system is suitable for military and non-military applications involving search and coordination activities and will work towards these applications in future work.

Acknowledgements

This work is supported by in part by the Naval Research Lab under grant N00173-04-1-G005. We would also like to thank Ben Shelton for his help in building and testing the robot team.

References

Anderson, D., Bailey, C., and Skubic, M. 2004. Hidden Markov Model Symbol Recognition for Sketch-Based Interfaces. AAAI 2004 Fall Symposium, Workshop on Making Pen-Based Interaction Intelligent and Natural, Washington, DC, Oct.

Chronis, G. and Skubic, M. 2004. Robot Navigation Using Qualitative Landmark States from Sketched Route Maps. In *Proc. IEEE. Conf. Robotics and Automation*, New Orleans, LA, May.

Fong, T. 2001. Collaborative control: A robot-centric model for vehicle teleoperation, Ph.D. dissertation Pittsburgh, PA: Robotics Inst., Carnegie Mellon Univ.

Fong, T., Thorpe, C. and Baur, C. 2003. Multi-Robot Remote Driving with Collaborative Control. IEEE Trans. Industrial Electronics. 50(4):699-704.

Kawamura, K., Koku, A.B., Wilkes, D.M., Peters II, R.A., and Sekmen, A. 2002. Toward Egocentric Navigation, *International Journal of Robotics and Automation*, 17(4): 135-145.

Lundberg, C., Barck-Holst, C., Folkeson, J., and Christensen, H.I. 2003. PDA Interface for Field Robot. In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems*, Las Vegas, NV, pp. 2882-2888.

Perzanowski, D., Schultz, A., Adams, W., Marsh, E. and Bugajska, M. 2001. Building a Multimodal Human-Robot Interface. *IEEE Intelligent Systems*. 16(1): 16-20.

Rabiner, L. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proc. IEEE.* 77(2).

Skubic, M., Bailey, C. and Chronis, G. 2003. A Sketch Interface for Mobile Robots, In *Proc. IEEE Conf. SMC* , Washington, D.C., Oct.