A Social Robot that Stands in Line

Yasushi Nakauchi and Reid Simmons*

Department of Computer Science
National Defense Academy
1-10-20 Hashirimizu
Yokosuka 239-8686 Japan

* School of Computer Science
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh 15213 USA

Abstract

Recent research in mobile robot navigation make it feasible to utilize autonomous robots in service fields. But, such applications require more than just navigation. To operate in a peopled environment, robots should recognize and act according to human social behavior. In this paper, we present the design and implementation of one such social behavior: a robot that stands in line much as people do. The system employs stereo vision to recognize lines of people, and uses the concept of personal space for modeling the social behavior. Personal space is used both to detect the end of a line and to determine how much space to leave between the robot and the person in front of it. Our model of personal space is based on measurements from people forming lines. We demonstrate our ideas with a mobile robot navigation system that can purchase a cup of coffee, even if people are waiting in line for service.

1 Introduction

Recent research in mobile robot navigation make it feasible to utilize autonomous robots in service fields [14]. In general, the environments where service robots operate are shared with humans. Thus, the robots have to interact with humans, whether they like it, or not.

Similarly, humans interact with each other. Sometimes, the interactions lead to resource conflicts. To

maintain order, humans use social rules. For example, at bank counters or grocery stores, humans stand in line and wait their turn. If a person does not understand or obey the social rules, he/she will generally not be able to get the services.

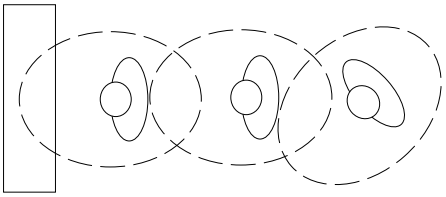
This is also true for service robots. If a service robot is asked to purchase merchandise and it does not know the social rules of humans, it may treat the humans who are standing in line as obstacles and will not be able to achieve its task. Therefore, service robots should be designed to understand the social behaviors of humans, and to obey social rules. To date, meal deliver robots in hospitals [3], tour guide robots in museums [15] and secretary robots in offices [8] have been developed and are actually used in real environments. These robots, however, generally do not interact with people according to social rules. This makes them much more difficult to interact with for the average person, who is used to interacting according to social conventions.

There are many aspects to the social rules and behaviors in human society. Among them, standing in line is one of the most highly socialized and crucial skills required for robots which operate in peopled environments. Therefore, as a first step towards creating a social robot, we have developed an autonomous robot system that can stand in line with other people [9]. The robot uses stereo vision to detect people, model their orientation, and model how (or if) they are forming into a line. The concept of *personal space* is used to model lines of people and robots: Personal space is used both to detect the end of a line and to determine how much space to leave between the robot and the person in front of it. Using this model, the robot moves until it detects the end of the line, then gets in place. It maintains its place in line, moving up (or back) according to how the person in front of it moves. Our experiments demonstrate the feasibility of this approach in a natural, indoor setting.

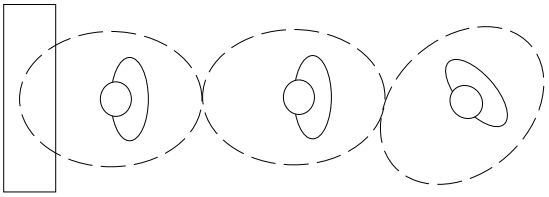
In the next section, we discuss and derive a model of how the people form a line as the result of social behaviors. In section 3, we present our algorithms for detecting lines of people using stereo vision and for having the robot enter and wait in line. Implementation issues and experimental results are described in sections 4 and 5, respectively.

2 Modeling a Line of People

To realize the social behavior of standing in line, a robot should be able to recognize a line of people, enter the end of the line, and wait in line as people do. To do this, we must first understand the social activities of humans and how they form lines. Then, we can use this understanding to create a robot system that recognizes the social behavior of humans and performs actions as people do.



(a) Closest case.



(b) Farthest case.

Figure 1: A line of people modeled by a chain of personal spaces.

The notion of "personal space" or "human territoriality" has been studied in the field of cognitive science [7, 13]. A person feels uncomfortable when other people are in his/her personal space, which is considered as the person's own territory. Cognitive science research has shown that personal space can be modeled as oval in shape, wider towards the front of a person. We can use the notion of personal space to model how people form, and move, in lines.

When people form a line, in general, they keep a certain distance from other people. They also usually stand so that they face towards the person in front. These phenomena can be described based on the concept of personal space. The idea is that the person who is standing in line keeps enough distance to the person in front so that he/she does not feel uncomfortable. On the other hand, he/she stands close enough to the person in front to avoid other people cutting in the line. We use these ideas to model a

line of people as a chain of personal spaces, as shown in Figure 1.

In general, there is some distance where we feel that we actually are standing in line. We assumed that the closest distance is observed when we stand as close as possible so that other person feels uncomfortable. In this case, the personal spaces of two people overlap, but a person may not be in the other person's personal space, as shown in Figure 1a. On the other hand, we assumed the farthest distance is observed when we stand as far as possible so that other person may not cut in a line. In this case, personal spaces of two persons are connected but do not overlap, as shown in Figure 1b.

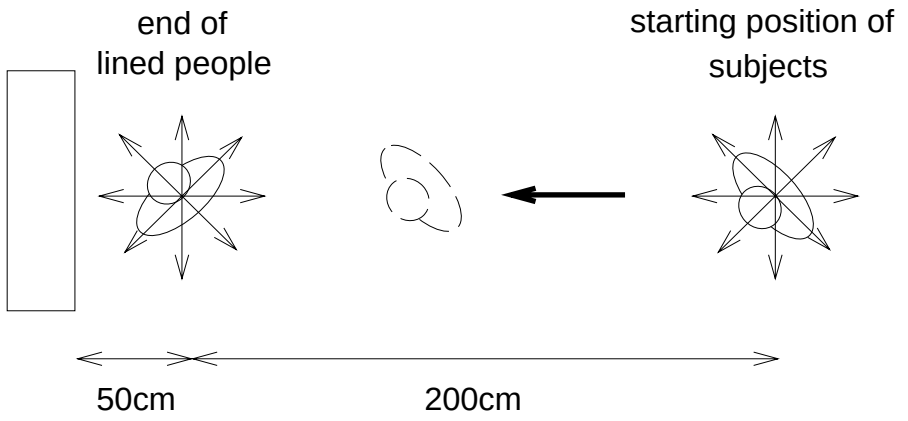


Figure 2: Experiment of personal space.

It is reported that the actual size of personal space varies depending cultural norms and on the task being performed (e.g., standing in line, sitting on a bench, chatting together) [7, 13]. Since we know of no work that explicitly addresses the size of personal space for the task of standing in line, we performed some experiments to estimate the actual size and shape. In particular, we asked subjects to start $200cm$ away from the person at the end of a line and to move forward towards the end of the line (see Figure 2). We asked the subjects to move in two ways. One is "to move forward until you feel uncomfortable," to estimate the closest range. The other is "to move forward until you feel that another person would not cut in a line," for the farthest case. We asked the subjects not to make eye contact or talk to the person in front, to eliminate the influence of affinity. Since personal space varies by body orientation, we collected data for eight different orientations of both the subject and the person at the end of the line.

We performed these experiments with 10 subjects.[1] When calculating the size of the personal space, we assumed the size of personal space of the person in front is identical to the personal space of the subject. This is because a subject has no way of knowing the size of the other person's personal space. Therefore, when the same body direction of two persons are facing, the personal space towards that direction is the half of the distance between two persons. The average size of personal space, as estimated from these experiments, is illustrated in Figure 3. As shown in the figure, the experimentally derived personal space is roughly oval and larger towards the front, as has been reported in the cognitive science literature.

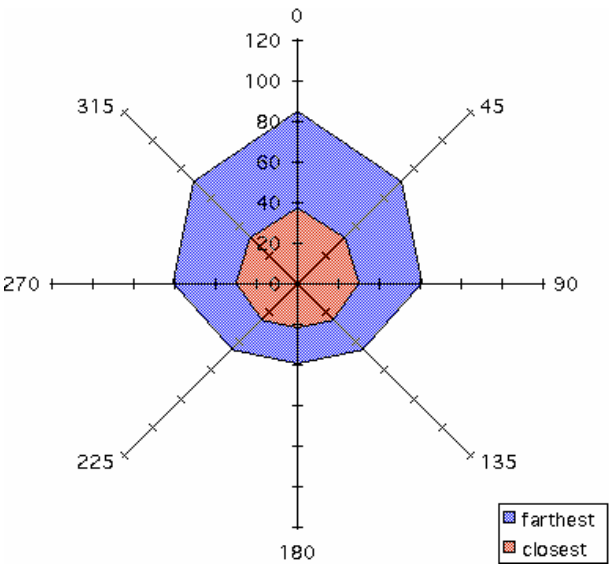


Figure 3: Actual size of personal space when people stand in line.

3 A Robot That Stands in Line

This section describes how we detect people using stereo vision, how we use that information to estimate which people are in line, and how the robot decides how to move to enter the line.

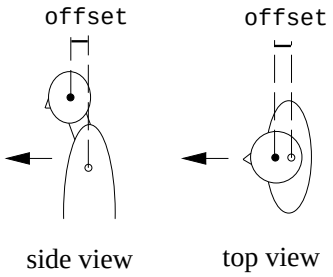


Figure 4: Our simple model of the human body.

3.1 Recognizing People in Line

To recognize a line of people, the robot detects each person's position and body orientation. One possible approach is to detect faces using vision [1, 12]. Unfortunately, most current face detection methods work only for frontal views. Since the robot typically observes people either from the side or the back, these face detection methods cannot be used for our purpose. In another approach, Oren et al. have developed a system to detect human bodies in an image [10]. This system only detects segments of people, however, and it cannot determine the positions and orientations of the people. Thus, we needed to develop a novel method for detecting people that is geared towards our application.

In general, three dimensional information is required to determine the position and orientation of people. Typically, stereo vision systems and laser range finders are used for that purpose. In this research, we use stereo vision system to detect people for two main reasons. First, stereo is cheap in cost compared with laser range finders. Second, there are readily available packages that can do real-time stereo calculations on images of modest size [6].

We model the shape of a human body as shown in Figure 4. We assume that a person has an oval shape around the torso, is round around the head, and that the neck is the narrowest part of the upper body. We also assume that the head juts over the body in front. Based on these assumptions, we have developed a person-detection algorithm using stereo vision (see Table 1).

The first two steps of the algorithm collect sensor data and perform the stereo calculations (see

[1] Throughout the research, the subjects used were graduate students and staff of the Robot Learning Laboratory, CMU.

Table 1: Algorithm to detect people using stereo vision.

1. Capture left and right camera images using two CCD cameras.
2. Calculate disparity image from the two images using triangulation method.
3. Convert disparity image to distance map.
4. Apply smoothing filter to distance map.
5. Cluster data points to separate out multiple objects (people) in the distance map.
6. Find the nose and the chest height position data for each person (see Figure 4, "side view").
7. Find the ellipse that best fits the stereo data at the chest height, and find the circle that best fits the stereo data at the nose height (see Figure 4, "top view").
8. Determine the body's orientation based on the center of the ellipse and the circle.

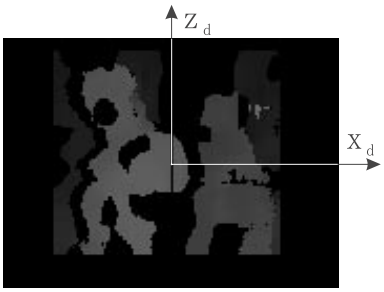
Figure 5). Next, we convert the disparity image to the three dimensional distance map based on the following equations.

$$\begin{cases} x_i &= (y_i + f)x_d/f \\ y_i &= b \cdot f/d_{x_d,z_d} \\ z_i &= (y_i + f)z_d/f \end{cases} \tag{1}$$

x_d and z_d are the coordinates of a data point in the disparity image and d_{x_d,z_d} is the disparity at that point (Figure 5-b shows the frame of reference). f is the focal length of the camera lenses and b is the baseline (the distance between the centers of the lenses). x_i, y_i, and z_i are the coordinates of a data point in the distance map (Figure 7-a shows the frame of reference). The distance map is represented as a three dimensional grid, discretized at $1cm$ resolution (see Figure 6). Since we calculate the distance map from disparities, which represent the surface of objects, there is the restriction that only one data point (y_i) exists in each x_i-z_i grid.

We then smooth the distance map by using the average of 8 neighboring data points (see Figure 6). To preserve edges in the distance map, we do not use neighboring data that is too far (further than $4m$)

(a-1) Left raw image. (a-2) Right raw image.



(b) Disparity image.

Figure 5: Raw images and the derived disparity image.

from a camera or where two data points in each direction have a gap of more than $10cm$ (this preserves the edges of people whose figures are overlapping). This process helps by filling in missing data points (by using the average of neighbors) and by smoothing curved surfaces, such as around the chest.

Next, we separate out the people. The problem is difficult because there may be multiple figures in the image, as well as other objects in the distance (such as furniture or walls). We use a clustering method to collect disparity data for each object in the image. Data points that are within $5cm$ of each other are connected, and all connected data points are categorized as a single cluster. We then discard clusters that have fewer than a threshold number of data points, since these typically represent noise or objects other than humans.

We then determine the position and orientation of each person from the clustered disparities. For each cluster, we find the nose and the chest height position data by using the characteristic body shape. First, we find the highest data point in a cluster and assume it is the top of the head. This is reliable since there are no data above the head, in general. Then, we find the narrowest point in width below the head, and assume it corresponds to the neck. Finally, we estimate the nose height as the middle point

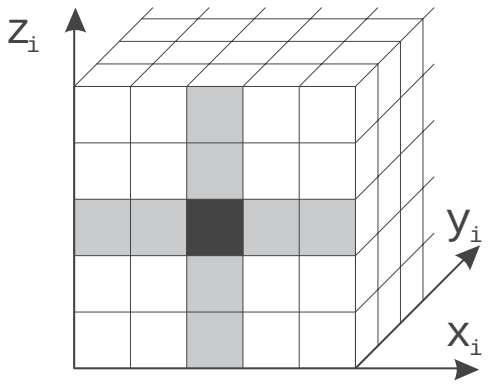


Figure 6: Eight neighboring data points in the distance map used for smoothing. Only one data point is available in the y_i (depth) direction. The y_i's of the gray shaded points are averaged to find the new y_i (the black shaded point).

between the head top and the neck, and the chest height as $20cm$ below the neck. At the same time, to eliminate objects other than people, we discard clusters whose highest point is lower than a certain threshold.

We then find the ellipse that best fits the stereo data at the chest height, and find the circle that best fits the stereo data at nose height. The mathematical expression of an ellipse that has arbitrary size, location and orientation is:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a\cos\theta_e\cos\theta + b\sin\theta_e\sin\theta + x_e \\ -a\sin\theta_e\cos\theta + b\cos\theta_e\sin\theta + y_e \end{bmatrix} \tag{2}$$

where a and b denote the length of the major and minor axes, respectively, x_e and y_e denote the offset of the center from the origin, and θ_e denotes the inclination of the ellipse. To simplify things, we use constant values for a and b for all people, assuming that humans are approximately the same size. Thus, the problem becomes finding the variables x_e, y_e and θ_e that best fits the observed data points.

Least-square fitting algorithms are efficient methods for calculating the best fitting ellipse from data points [2, 11]. These methods do not restrict the size of ellipses they consider, however, so they can potentially fit the data to very large ellipses if the data points are mainly linear. Even if we use a modified algorithm that restricts the ellipse size, the best fitting model, in the least-squares sense, may be one that is not physically possible. For instance, due to noisy data, a least-squares algorithm might

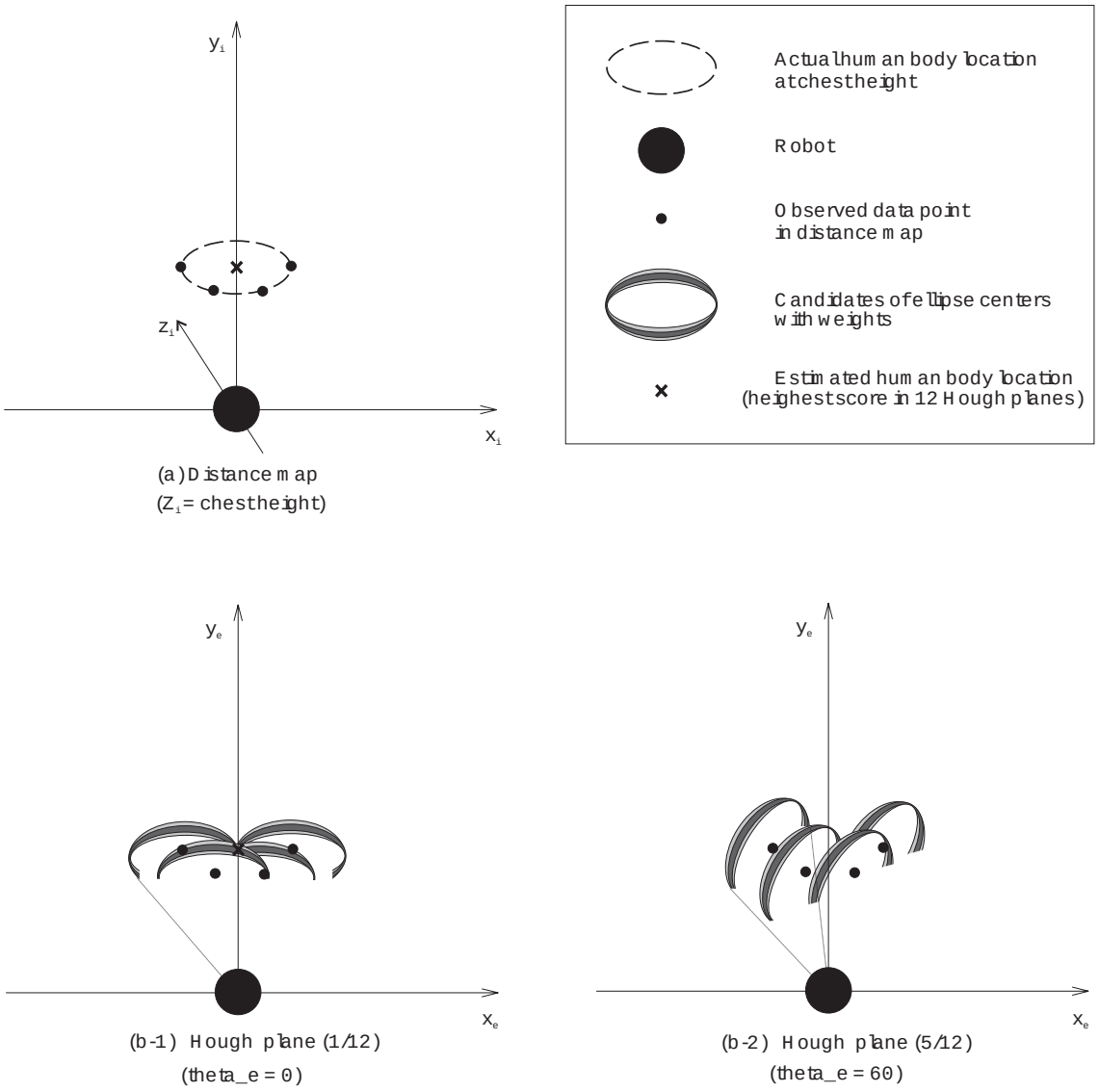


Figure 7: Hough transform used for ellipse fitting.

decide that the best fitting ellipse lies in *front* of the data points, which would imply that the cameras had seen through the body. Thus, we need the algorithm to constrain the ellipse (body) to fall "behind" the data points, relative to the gaze of the camera.

The Hough transform [4] is an efficient and intuitive method to calculate geometric features from data points. Since it has fewer limitations than the least-squares method, we use the Hough transform with some modifications to make it applicable for people detection.

Since there are three variables to estimate, we need to have a three dimensional Hough transform space. To do this, we discretize orientation (θ_e) in 15 degree increments and use 12 separate x_e-y_e Hough planes, each corresponding to a different θ_e. The x_e-y_e planes are represented as grids, discretized at $1cm$ resolution. The basic idea behind our Hough-transform based algorithm is to draw ellipses centered around each data point. The ellipses are drawn at different orientations for each x_e-y_e plane. For example, Hough images for $\theta_e = 0$ and $\theta_e = 60$ are the x_e-y_e planes as shown in Figure 7-b-1 and 7-b-2, respectively. In each plane, the grid point that has the most number of ellipse intersections is considered to be the most likely center of the ellipse. The plane that has the highest ranking center point determines the most likely orientation.

In the example shown in Figure 7, the data points come from viewing a person whose orientation is $\theta_e = 0$ (for clarity, we are using data points with no noise in this example). Thus, in the projected x_e-y_e plane where $\theta_e = 0$ (Figure 7-b-1), the ellipses all intersect at the position where the person is actually located (note that each curve in Figure 7-b represents potential *center points* of the body ellipse, given a particular data point). But in the x_e-y_e plane where $\theta_e = 60$ (Figure 7-b-2), the ellipses do not intersect very much. Thus, $\theta_e = 0$ is determined to be the best orientation.

To further take account of noisy data, we have made additional modifications to the Hough transform. One problem is that, if the ellipses are drawn with a thin line, the number of intersections may be small or, in some cases, the wrong position may become dominant. We reduce that possibility by widening the rims of the ellipses (Figure 7-b). At the same time, we weight the rims so that the center of the rim has a high weight and the weights decrease towards the edges of the rims.

To ensure that the candidate ellipses are actually behind the data points, we use only the portion of the ellipse that can observed by the camera. In particular, the ellipse is cut at the tangent lines from

the origin (camera position), denoted by dotted lines in Figure 7-b-2. Note that the portions are concave because the curves are potential center points, not the actual candidate body ellipses.

The algorithm to find a circle that corresponds to the head position is the same as for ellipse fitting. However, since the circle does not have orientation, we need only one x_e-y_e plane.

Finally, we determine the body orientation. We can use the direction of the minor axis of the best fitting ellipse for the chest height data to determine body orientation. Using only this measure, however, it is ambiguous whether the person is facing forward or backward. So, based on the assumption that the head juts out in front of the body, we resolve the ambiguity by calculating the offset of the head circle from the body ellipse. An example of recognized humans with their locations and orientations are as shown in Figure 8. Section 5 presents experimental results on the accuracy and reliability of this procedure.

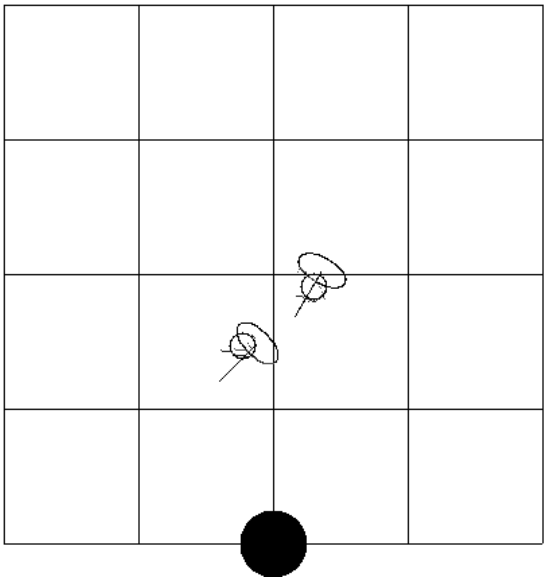


Figure 8: Two people detected by the stereo vision system. (black circle: robot; ellipse: chest; circle: head; arrow: body orientation; grid size: $1m$)

3.2 Entering and Waiting in Line

To begin the process of standing in line, the robot is told the location of the head of the line (the *point of service*) and the (approximate) direction in which the line forms. The robot navigates (see Section 4) to a

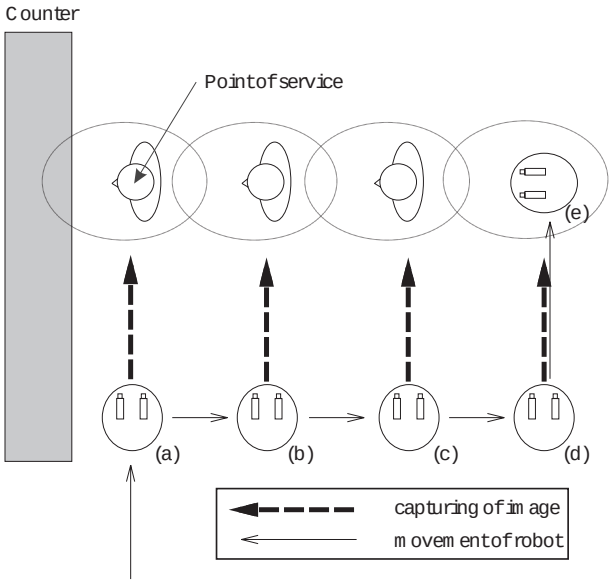


Figure 9: The movement of the robot to enter a line.

point 150cm from the point of service and faces perpendicular to the expected line direction (Figure 9a). If it detects no body at the point of service, it moves forward, rotates 90 degrees, and assumes that it is at the head of the line.

If a person is detected, on the other hand, the system turns and moves down the line. The distance moved is based on the position of the last person detected, their orientation, and the estimated size of their personal space. For this calculation, we use the average of the closest and the farthest personal space, as shown in Figure 2. Thus, the robot will travel further if the person is standing backwards than if he/she is facing forwards (towards the point of service).

After moving, the robot tries again to detect people. If there is no one detected, or if the distance between the last detected person and the current detected person is greater than the farthest personal distance, then the robot moves into line (a better approach for this latter case would be to ask the person whether he/she is indeed in line, but this is a subject for future work). Otherwise, the robot moves further from the point of service until it eventually finds the place to enter the line. This process is illustrated in Figure 9, as the robot moves from positions (b) to (e).

Once the robot enters the line, it uses a combination of sonar and laser sensors to track the position of the person in front of it. The robot moves up in line to keep its personal space connected to the presumed

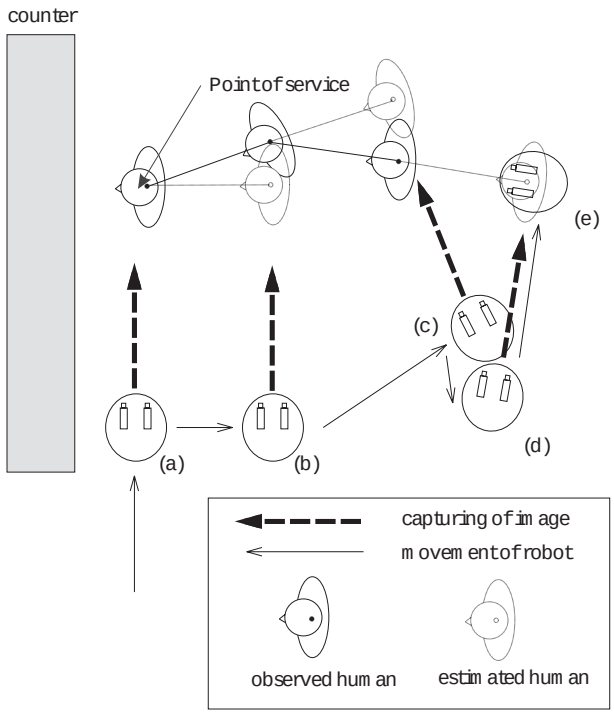


Figure 10: The movement of the robot to enter a curved line.

personal space of the person in front. It will also back up if that person backs up enough to impinge on the robot's own personal space. The robot also recognizes when it has reached the head of the line. For this, we assume that the point of service is a counter and we use data from a laser range finder, set at the height of human legs. If the observed object surface is smooth enough, it is recognized as the counter. If not, it is recognized as a person.

We have extended the above procedure to handle curved lines, as well. The basic idea is to keep an ordered list of all the detected people. Each time the robot detects a new person, it adds that person to the list and computes a vector between the positions of the last two people in the list. This vector is the robot's estimate of the local direction of the line (Figure 10). The robot then turns so that it is heading parallel to that vector and moves a sufficient distance so that it will be across from the expected position of the next person in line (under the assumption that the line continues in the given direction). Once the robot is in line, it moves in the direction of the next person on the list, so that it follows the curve of the line as it moves forward. This algorithm has been show to work reliably (Section 5), as long as the line does not bend too sharply.

4 Implementation

Figure 11: Mobile robot Xavier.

We have implemented these algorithm on the autonomous mobile robot Xavier [14]. Xavier is built on top of a 24 inch diameter Real World Interface base (see Figure 11). The base is a four-wheeled synchro-drive mechanism that allows for independent control of the translational and rotational velocities. The sensors of Xavier include bump panels, wheel encoders, a 24 element sonar ring, a Nomadics front-pointing laser light striper with a 30 degree field of view, and two Sony monochrome cameras on a Directed Perception pan-tilt head. Xavier also has a speaker and a text-to-speech card. Control, perception, and planning are carried out on two 200 MHz Pentium computers, running Linux.

The software architecture of Xavier is based on the Task Control Architecture (TCA) [14]. TCA facilitates synchronization amongst a collection of processes and provides facilities for interprocess communication between the processes. The following modules were previously developed and were used in our research without modification (see Figure 12).

Base Control Module This module controls the base wheels, providing translation and rotation of the robot, and does obstacle avoidance. It also manages the laser and sonar sensors, and reports the

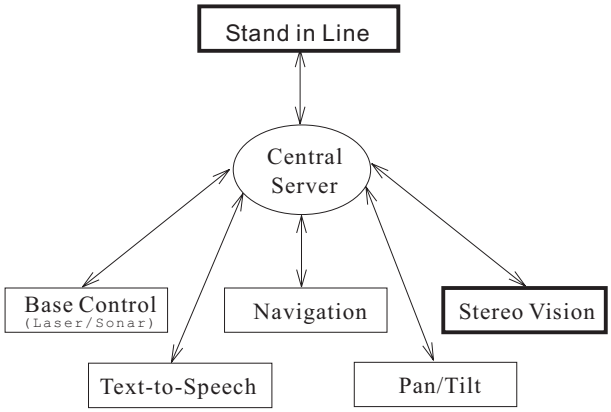


Figure 12: The software architecture of Xavier.

current readings to other modules.

Navigation Module This module maintains the environmental map. It uses decision-theoretic path planning and a probabilistic navigation scheme to reliably navigate the robot to a specified goal position [5].

Text-to-Speech Module This module synthesizes and outputs arbitrary natural language text from the speaker. It uses a commercial text-to-speech board.

Pan/Tilt Module This module pans and tilts the cameras.

For this project, we developed two new modules (the thickly lined boxes in Figure 12): one that performs stereo vision, and another that controls the overall "stand in line" task. The stereo vision module utilizes two monochrome CCD cameras and recognizes humans based on the algorithm described in section 3.1. For the calculation of disparity image, we used the SVS[2] stereo engine [6]. The cycle time for people detection was about $800msec$. The "stand in line" module controls the overall task – getting to near the point of service, finding and entering the line, and moving up to the head of the line.

As a practical application that requires the robot to stand in line, we developed a program that enable the robot to purchase a cup of coffee from a kiosk in the foyer of our building. The approach, summarized in Table 2, is described here in more detail. The robot first gets an order from a user, asking it to purchase a cup of coffee. Currently, this is done via keyboard input, but we are working on having voice-activated

[2] The SVS (Small Vision System) is a trade mark of SRI International.

Table 2: The procedure to purchase a cup of coffee.

1. Get an order from a user.
2. Navigate to the coffee counter.
3. Recognize people standing in line.
4. If people are standing in line, find the place to stand and join the line.
5. Move up in line by keeping its personal space from a person in front.
6. Place an order.
7. Determine when a cup has been received.
8. Navigate back to the place where it got the order and announce arrival.

commands. The robot then navigates to the coffee kiosk located in the foyer of our building. Here, we assume that the robot has a map of the building so that it can navigate to the kiosk and precisely localize in front of the counter. The robot then finds the end of the line, enters the back of the line, moves up until it detects that it is at the counter (i.e., at the head of the line).

At the counter, the robot uses synthesized speech to place an order and waits for the attendant to put a coffee cup into a holder attached to the front of the robot. To determine when it has received the cup, the robot tilts its camera so that it can view the cup holder and waits until the camera image changes significantly. After politely thanking the attendant (and asking for the coffee to be put on its tab), the robot navigates back to the place where it got the order and announces that it has arrived with the (hopefully still hot) cup of coffee.

5 Experimental Results and Discussion

To measure the performance and reliability of the system, we performed a number of experiments with the robot in various situations. First, we conducted experiments to determine the quality of the algorithm that detects the position and orientation of people. The errors in depth, horizontal direction, and orientation

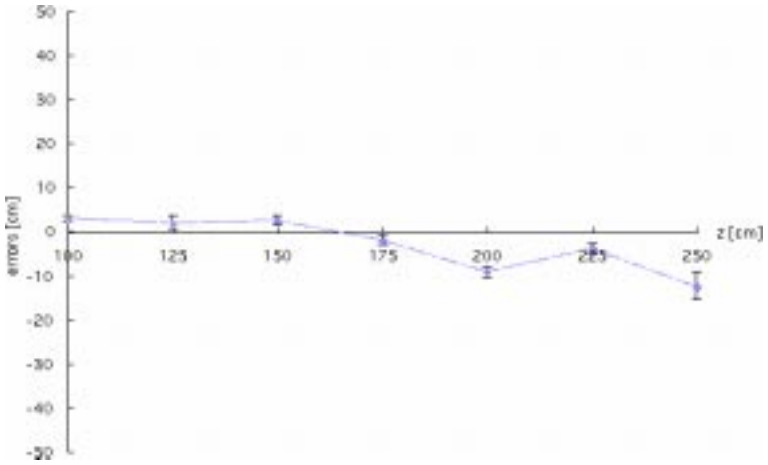


Figure 13: The depth error for the people detection algorithm

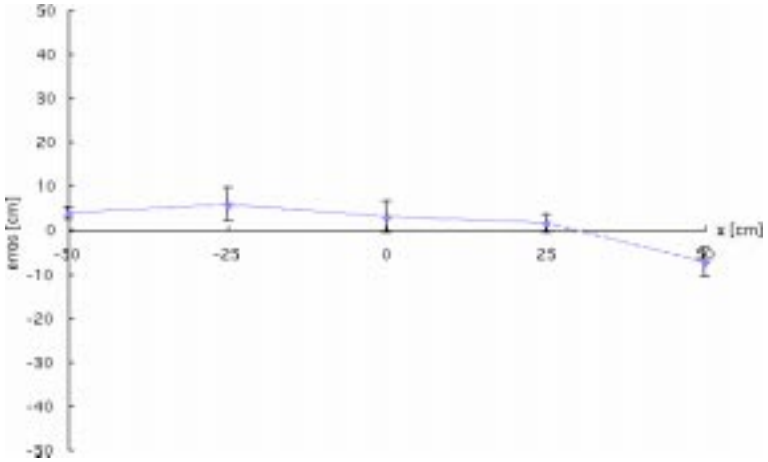


Figure 14: The horizontal error for the people detection algorithm

(with error bars) are shown in Figures 13, 14, and 15, respectively. For these experiments, we used 5 subjects and, for each subject, took 5 measurements for each position and orientation.

It is observed that the distance error increases as a person stands far from the camera and as they stand off center. Fortunately, our "stand in line" behavioral algorithm is relatively immune to errors of this type, since the robot tries to move to a spot where it predicts it will get a good view of the next person in line. As for the orientation error, it increases when the robot observe the backs of people. This is because there is typically less texture on the back of head, so it becomes more difficult to obtain good stereo data. When the robot observes a line of people, however, it usually sees people from the side. So,

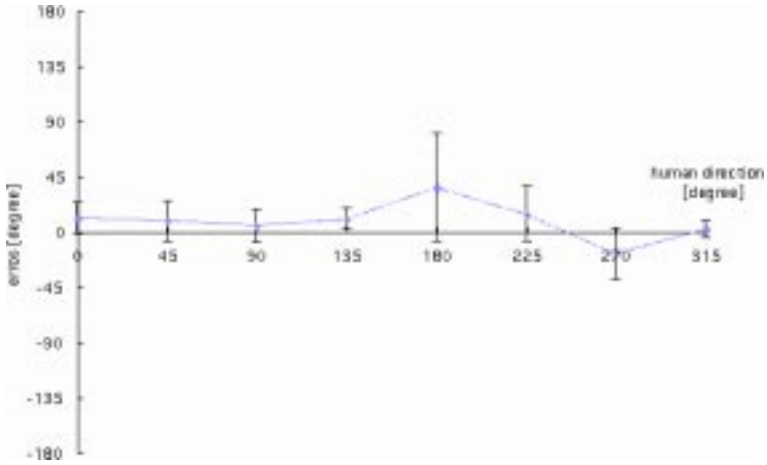


Figure 15: The orientation error for the people detection algorithm

the average orientation error is much less than the worst case.

To simplify the problem, we have assumed that people are all the same size. In particular, we use constant values for the major and minor axes of the body ellipses. The question remains: How sensitive is the algorithm to differences in peoples' shapes? In particular, is the algorithm sensitive to girth (fat or thin), height (tall or short), and appearance of the head (with hats, bald, or up-swept hairdos)?

With respect to fat or thin people, our algorithm finds the best fit ellipse, which is weighted (fattened) along the major axis (see Figure 7). Thus, it tends to weight more heavily data points along the chest (or back) rather than the side or shoulder, which is where one sees the main differences in chest size. So, the algorithm is robust to different girths. With respect to tall or short people, our algorithm finds the highest data point as the top of the head, then finds narrowest part as the neck. Thus, it can accommodate people of various heights. One indication of the robustness of the people detection algorithm is the small sizes of (most of) the error bars in Figures 13-15, since those experiments were performed with subjects of varied body types.

As for a person with hats, bald head, or up-swept hairdos, the recognition results seem to be affected by the texture and shape of the head. While sometimes hats or bald heads can provide rich texture for stereo matching, typically however, they have an adverse effect on stereo. For instance, the brim of a hat may shade the face, leading to a uniformly dark image that has insufficient texture for stereo matching. To some extent, the Hough transform can work with scattered disparity data points. But, we have to do

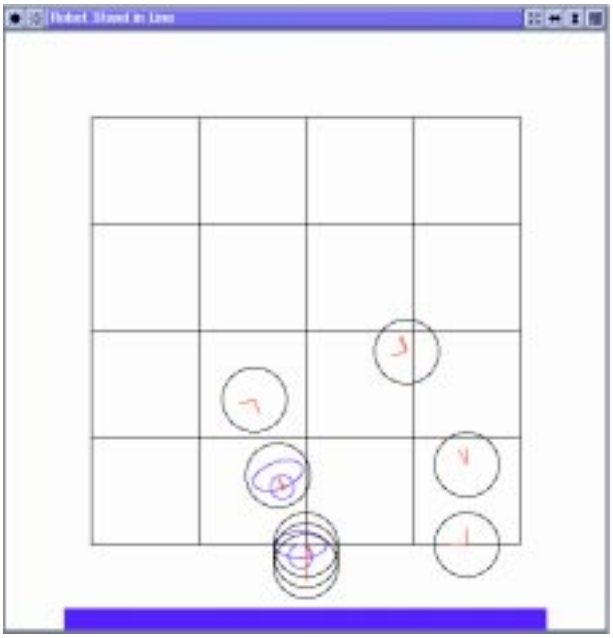


Figure 16: A typical result where Xavier stands in line.

investigate further to determine exactly when such effects occur, and how to handle them.

On the other hand, we have found that an up-swept hairdo may lead to inaccurate determination of orientation. This is because such hairdos tend to cause the circle-fitting algorithm to place the head position further back than it really is. One remedy may be to use other body features, such as arms and hands, that are more indicative of body posture.

To determine how well the robot performs in the overall task, we ordered the robot to purchase coffee at a real kiosk located in the foyer on the same floor of the building as our laboratory. The robot was able to detect the line correctly, enter it, and move up to the head of the line in 14 times out of 20 trials (70% success rate). A trace of one of the typical results where Xavier successfully stands in line is shown in Figure 16.

The majority of failures were caused either by failure of the people detection algorithm or the fact that the robot moves rather slowly, compared to people. The detection algorithm can fail because harsh lighting conditions, such as reflections or shadows, can lead to poor stereo results. One way to deal with this problem is to augment vision with other information sources, such as 3D laser range finders.

The slow speed of the robot (for safety, the maximum speed is set to $20cm/sec.$) sometimes causes a

problem because it cannot enter the line fast enough (this is exacerbated by the fact that the robot has to view the line from 150cm away, because the stereo vision is not reliable closer in). Sometimes, another person would enter the line after the robot had determined where the end of the line should be. Although sometimes the robot would still get in place correctly, other times this new situation would confuse it. One solution is to have it move faster; Another is to look out for people as it moves into place. Finally, sometimes the robot makes a mistake and gets behind a person who is near the line, but not actually in it. Here, a solution is to have the robot ask the person if he/she is really in line, if there is no movement after a reasonable period.

The algorithm we developed assumes certain things about the environment. For one, it assumes that there is enough room for the robot to follow along beside the line from the point of service to the end of line. Also, it assumes that the lines do not bend too sharply and there is enough room in the environment for the robot to enter behind the last person, in the general direction of the line. Also, we presume that there is one line for each point of service, unlike places such as banks and airport counters, where one line divides to several points of service. In cases where these assumptions do not hold, the algorithm would have to be extended. For instance, in the former case, the system could use afforded clues such as the positions and directions of people observed around the point of service. Then, it could estimate if they are forming a line towards the point of service by using the model of lines of people denoted by chain of personal spaces (see Figure 1). For instance, in the latter case, the system could use afforded clues such as a painted line on the floor or poles connected by tape, that often indicate the presence of a line of people. Of course, these issues are for future research.

Many of the experimental trials were done during normal operating hours of the coffee shop, with the experimenters unseen. While many of the customers had seen Xavier previously, they had only encountered it navigating through the corridors. Nevertheless, several customers got in line behind Xavier and waited patiently in line together with the robot. This, at least, provides some anecdotal evidence that Xavier is considered, in some sense, to be a social member of the community.

6 Conclusion

This paper presented a social robot that can stand in line as people do. It presents a model of lines of people using the notion of personal space, and describes how we experimentally obtained an estimate of the size and shape of personal space for the task of standing in line. The paper then describes an algorithm that uses this model, together with a person detection algorithm based on stereo vision, to enter and wait in line. Finally, it discusses an implementation of the algorithms on the mobile robot Xavier, and experimentally demonstrates its performance.

We believe the notion of personal space can be used in various situations. For example, when we join a conversation between people who are standing in a corridor, we select the direction to approach and keep a certain distance from the other people in accordance with our notions of personal space. In future work, we intend to develop other social behaviors, specifically by adding conversation and face tracking abilities to the robot.

Acknowledgements

We would like to thank Kurt Konolige, SRI International, Takeo Kanade and Illah Nourbakhsh, Robotics Institute, CMU for useful discussions. Greg Armstrong helped in carrying out the experiments with Xavier.

References

[1] Brunelli, R. and Poggio, T. 1995. Template Matching: Matched Spatial Filters and Beyond. M.I.T., Cambridge, MA, A.I. Memo, 1536.

[2] Fitzgibbon, A.W., Pilu, M. and Fisher, R.B. 1996. Direct Least Square Fitting of Ellipses. Dept. of Artif. Intell., The University of Edinburgh, Scotland, DAI Research Paper. 794.

[3] HelpMate Robotics Inc., http://users.ntplx.net/~helpmate.

[4] Illingworth, J. and Kittler, J. 1988. A survey of the Hough transform. Computer Vision, Graphics, and Image Processing. Vol. 44, pp.87–116.

[5] Koenig, S., Goodwin, R. and Simmons, R., 1996. Robot Navigation with Markov Models: A Framework for Path Planning and Learning with Limited Computational Resources. In *Reasoning with Uncertainty in Robotics*, Lecture Notes in Artificial Intelligence, 1093, Dorst, van Lambalgen and Voorbraak (eds.), pp. 322–337, Springer.

[6] Konolige, K. 1997. Small Vision Systems: Hardware and Implementation. In Proc. Eighth International Symposium on Robotics Research.

[7] Malmberg, M. 1980. *Human Territoriality: Survey of Behavioural Territories in Man with Preliminary Analysis and Discussion of Meaning*, Mouton Publishers.

[8] Matsui, T. et al. 1997. An Office Conversation Mobile Robot That Learns by Navigation and Conversation. In Proc. of Real World Computing Symposium, pp.59–62.

[9] Nakauchi, Y. and Simmons, R. 1999. Social Behavioral Robot that Stands in Line, In Proc. of IEEE SMC, pp.II-993–998.

[10] Oren, M. Papageorgiou, C. Shinha, P. Osuna, E. and Poggio, T. 1997. A Trainable System for People Detection. In Proc. of Image Understanding Workshop, pp.207–214.

[11] Rosin, P.L. 1993. A Note on the Least Square Fitting of Ellipses. Pattern Recognition Letters (14): 799–808.

[12] Rowley, H. Baluja, S. and Kanade, T. 1998. Neural Network-based Face Detection. In Proc. of IEEE PAMI.

[13] Sack, R. 1986. *Human Territoriality*, Cambridge University Press.

[14] Simmons, R. Goodwin, R. Zita Haigh, K. Koenig, S. and O'Sullivan, J. 1997. A Layered Architecture for Office Delivery Robots. In Proc. of Autonomous Agents, pp.245–252.

[15] Thrun, S. et al., 1999. MINERVA: A Second Generation Mobile Tour-guide Robot. In Proc. of IEEE ICRA'99.