A Gesture Based Interface for Human-Robot Interaction

Stefan Waldherr
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA

Roseli Romero
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, SP, Brazil

Sebastian Thrun
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA

Abstract.

Service robotics is currently a pivotal research area in robotics, with enormous societal potential. Since service robots directly interact with people, finding "natural" and easy-to-use user interfaces is of fundamental importance. While past work has predominately focussed on issues such as navigation and manipulation, relatively few robotic systems are equipped with flexible user interfaces that permit controlling the robot by "natural" means.

This paper describes a gesture interface for the control of a mobile robot equipped with a manipulator. The interface uses a camera to track a person and recognize gestures involving arm motion. A fast, adaptive tracking algorithm enables the robot to track and follow a person reliably through office environments with changing lighting conditions. Two alternative methods for gesture recognition are compared: a template based approach and a neural network approach. Both are combined with the Viterbi algorithm for the recognition of gestures defined through arm motion (in addition to static arm poses). Results are reported in the context of an interactive clean-up task, where a person guides the robot to specific locations that need to be cleaned and instructs the robot to pick up trash.

1. Introduction

The field of robotics is currently undergoing a change. While in the past, robots were predominately used in factories for purposes such as manufacturing and transportation, a new generation of "service robots" has recently begun to emerge (Schraft and Schmierer, 1998). Service robots cooperate with people, and assist them in their everyday tasks. Specific examples of commercial service robots include the Helpmate robot, which has already been deployed at numerous hospitals worldwide (King and Weiman, 1990), an autonomous cleaning robot that has

successfuly been deployed in a supermarket during opening hours (Endres *et al.*, 1998), and the Robo-Caddy (www.icady.com/homefr.htm), a robot designed to make life easier by carrying around golf clubs. Few of these robots can interact with people other than by avoiding them. In the near future, similar robots are expected to appear in various branches of entertainment, recreation, health-care, nursing, etc., and it is expected that they interact closely with people.

This upcoming generation of service robots opens up new research opportunities. While the issue of *robot navigation* has been researched quite extensively (Cox and Wilfong, 1990; Kortenkamp *et al.*, 1998; Borenstein *et al.*, 1996), considerably little attention has been paid to issues of *human-robot interaction* (see Section 5 for a discussion on related literature). However, many service robots will be operated by non-expert users, who might not even be capable of operating a computer keyboard. It is therefore essential that these robots be equipped with "natural" interfaces that facilitate the interaction between robots and people.

Nevertheless, the need for more effective human-robot interfaces has well been recognized by the research community. For example, Torrance developed in his M.S. thesis a natural language interface for teaching mobile robots names of places in an indoor environment (Torrance, 1994). Due to the lack of a speech recognition system, his interface still required the user to operate a keyboard; nevertheless, the natural language component made instructing the robot significantly easier. More recently Asoh and colleagues developed an interface that integrates a speech recognition system into a phrase-based natural language interface (Asoh *et al.*, 1997). The authors successfully instructed their "office-conversant" robot to navigate to office doors and other significant places in their environment through verbal commands. Among people, communication often involves more than spoken language. For example, it is far easier to point to an object than to verbally describe its exact location. Gestures are an easy way to give geometrical information to the robot. Hence, other researchers have proposed vision-based interfaces that allow people to instruct mobile robots via arm gestures (see Section 5). For example, both Kortenkamp and colleagues (Kortenkamp *et al.*, 1996) and Kahn (Kahn, 1996) have developed mobile robot systems instructed through arm poses. Most previous mobile robot approaches only recognize static arm poses as gestures, and they cannot recognize gestures that are defined through specific temporal patterns, such as waving. Motion gestures, which are commonly used for communication among people, provide additional freedom in the design of gestures. In addition, they reduce the chances of accidentally classifying arm poses as gestures that were not intended as such. Thus,

they appear better suited for human robot interaction than static pose gestures alone.

Mobile robot applications of gesture recognition impose several requirements on the system. First of all, the gesture recognition system needs to be small enough to "fit" on the robot, as processing power is generally limited. Since both the human and the robot may be moving while a gesture is shown, the system may not assume a static background or a fixed location of the camera or the human who performs a gesture. In fact, some tasks may involve following a person around, in which case the robot must be able to recognize gestures while it tracks a person and adapt to possibly drastic changes in lighting conditions. Additionally, the system must work at an acceptable speed. Naturally, one would want that a 'Stop' gesture would immediately halt the robot—and not five seconds later. These requirements must be taken into consideration when designing the system.

This paper presents a vision-based interface that has been designed to instruct a mobile robot through both pose and motion gestures (Waldherr *et al.*, 1998). At the lowest level, an adaptive dual-color tracking algorithm enables the robot to track and, follow a person around at speeds of up to 30 cm per second while avoiding collisions with obstacles. This tracking algorithm quickly adapts to different lighting conditions, while segmenting the image to find the person's position relative to the center of the image, and using a pan/tilt unit to keep the person centered. Gestures are recognized in two phases: In the first, individual camera images are mapped into a vector that specifies likelihood for individual poses. We compare two different approaches, one based on neural networks, and one that uses a graphical correlation-based template matcher. In the second phase, the Viterbi algorithm is employed to dynamically match the stream of image data with pre-defined temporal gesture templates.

The work reported here goes beyond the design of the gesture interface. One of the goals of this research is to investigate the usability of gesture interface in the context of a realistic service robot application. The interface was therefore integrated into our existing robot navigation and control software. The task of the robot is motivated by the "clean-up-the-office" task at the 1994 mobile robot competition (Simmons, 1995). There, a robot had to autonomously search an office for objects scattered at the floor, and to deposit them in nearby trash bins. Our task differs in that we want a human to guide the robot to the trash, instructed with gestures. The robot should than pick up the trash and carry it and dump it into a trash bin.

The remainder of the paper is organized as follows. Section 2 describes our approach to visual servoing, followed by the main gesture

recognition algorithm described in Section 3. Experimental results are discussed in Section 4. Finally, the paper is concluded by a discussion of related work (Section 5) and a general discussion (Section 6).

2. Finding and Tracking People

Finding and tracking people is the core of any vision-based gesture recognition system. After all, the robot must know where in the image the person is. Visual tracking of people has been studied extensively over the past few years (Darrel *et al.*, 1996), (Crowley, 1997), and (Wren *et al.*, 1997). However, the vast majority of existing approaches assumes that the camera is mounted at a fixed location. Such approaches typically rely on a static background, so that human motion can be detected, e.g., through image differencing. In the case of robot-based gesture recognition, one cannot assume that the background is static. While the robot tracks and follows people, background and lighting conditions often change considerably. In addition, processing power on a robot is usually limited, which imposes an additional burden. As we will see in this Section, an adaptive color based approach tracking both face and shirt color appears to be capable of tracking a person under changing lighting conditions.

Our approach tracks the person based on a combination of two colors, *face color* and *shirt color*. Face color as a feature for tracking people has been used before, leading to remarkably robust face tracking as long as the lighting conditions do not chance too much (Yang and Waibel, 1995). The advantage of color over more complicated models (e.g., head motion, facial features (Rowley *et al.*, 1998)) lies in the speed at which camera images can be processed—a crucial aspect when implemented on a low-end robotic platform. In our experiments, however, we found color alone to be insufficient for tracking people using a moving robot, hence we extend our algorithm to tracking a second color typically aligned vertically with a person's face: *shirt color*. The remainder of this section describes the basic algorithm, which run at approximately 10Hz on a low-end PC.

2.1. Color Filters

Our approach adopts a basic color model commonly used in the face detection and tracking literature. To detect face color, it is advantageous to assume that camera images are represented in the RGB color space. More specifically, each pixel in the camera image is represented by a triplet $P = (R, G, B)$. RGB (implicitly) encodes color, brightness, and

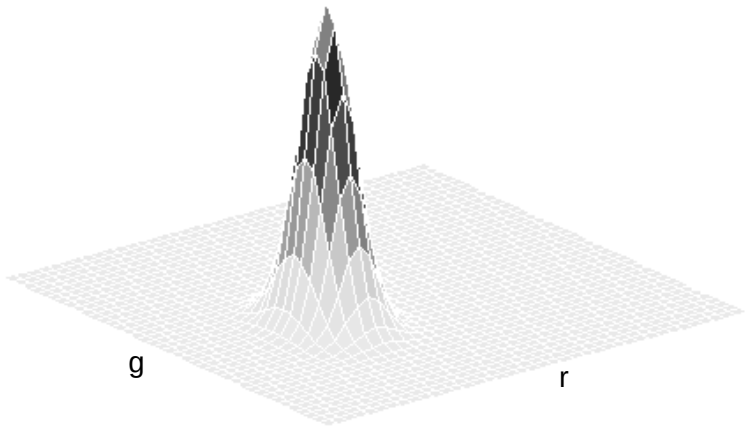


Figure 1. Face-color distribution in chromatic color space. The distribution was obtained from the person's face shown in Figure 2(a).

saturation. If two pixels in a camera image, P_1 and P_2, are proportional, that is

$$\frac{R_{P_1}}{R_{P_2}} = \frac{G_{P_1}}{G_{P_2}} = \frac{B_{P_1}}{B_{P_2}}, \tag{1}$$

they share the same color but (if $R_{P_1} \neq R_{P_2}$) differ in brightness. Since brightness may vary drastically with the lighting condition, it is common practice to represent face color in the *chromatic color space*. Chromatic colors (Wyszecki and Styles, 1982), also known as *pure colors*, are defined by a projection $\Re^3 \mapsto \Re^2$, where

$$r = \frac{R}{R+G+B} \tag{2}$$

and

$$g = \frac{G}{R+G+B}. \tag{3}$$

Following the approach by Yang et al. in (Yang and Waibel, 1995), our approach represents the chromatic face color model by a Gaussian model $M = (\mathbf{\Sigma}, \mu_r, \mu_g)$, characterized by the means μ_r and μ_g

$$\mu_r = \frac{1}{N}\sum_{i=0}^{N} r_i, \tag{4}$$

$$\mu_g = \frac{1}{N}\sum_{i=0}^{N} g_i \tag{5}$$

and the covariance matrix $\mathbf{\Sigma}$

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_r^2 & \sigma_{rg} \\ \sigma_{gr} & \sigma_g^2 \end{pmatrix}, \tag{6}$$

where

$$\sigma_r^2 = \sum_{i=0}^{N} (r_i - \mu_r)^2, \tag{7}$$

$$\sigma_g^2 = \sum_{i=0}^{N} (g_i - \mu_g)^2 \tag{8}$$

and

$$\sigma_{rg} = \sigma_{gr} = \sum_{i=0}^{N} ((r_i - \mu_r)(g_i - \mu_g)). \tag{9}$$

Typically, the color distribution of human face color is centered in a small area of the chromatic color space, i.e., only a few of all possible colors actually occur in a human face. Figure 1 shows such a face color distribution for the person shown in Figure 2(a). The two horizontal axes represents the r and g values, respectively, and the vertical axis is the response to a color model. The higher the response, the better the person's face matches the color model.

Under the Gaussian model of face color given above, the Mahalanobis distance of the i'th's pixels chromatic r and g value is given by:

$$f(r_i, g_i) \;=\; \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}}\, e^{-\frac{(r_i - \mu_r, g_i - \mu_g)^T \Sigma^{-1} (r_i - \mu_r, g_i - \mu_g)}{2}} \tag{10}$$

Here the superscript ^{T} refers to the transpose of a vector.

As noticed above, our approach uses two distinctive colors for track people: face color and shirt color. Consequently, it employs two separate color models:

$$M_{\text{face}} = (\mathbf{\Sigma}_{\text{face}}, \mu_{r_{\text{face}}}, \mu_{g_{\text{face}}}) \tag{11}$$

and

$$M_{\text{shirt}} = (\mathbf{\Sigma}_{\text{shirt}}, \mu_{r_{\text{shirt}}}, \mu_{g_{\text{shirt}}}) \tag{12}$$

for face and shirt color, respectively. To determine a person's location in the image, each pixel is mapped into chromatic color space and the responses to the face and shirt color filter are computed: $f_{\text{face}}(r_i, g_i)$ and $f_{\text{shirt}}(r_i, g_i)$ (see Equation (10)). For the raw camera image in Figure 2(a) one can see the response to the face color filter in Figure 2(b) and to the shirt color filter in Figure 2(c). The better the pixel matches

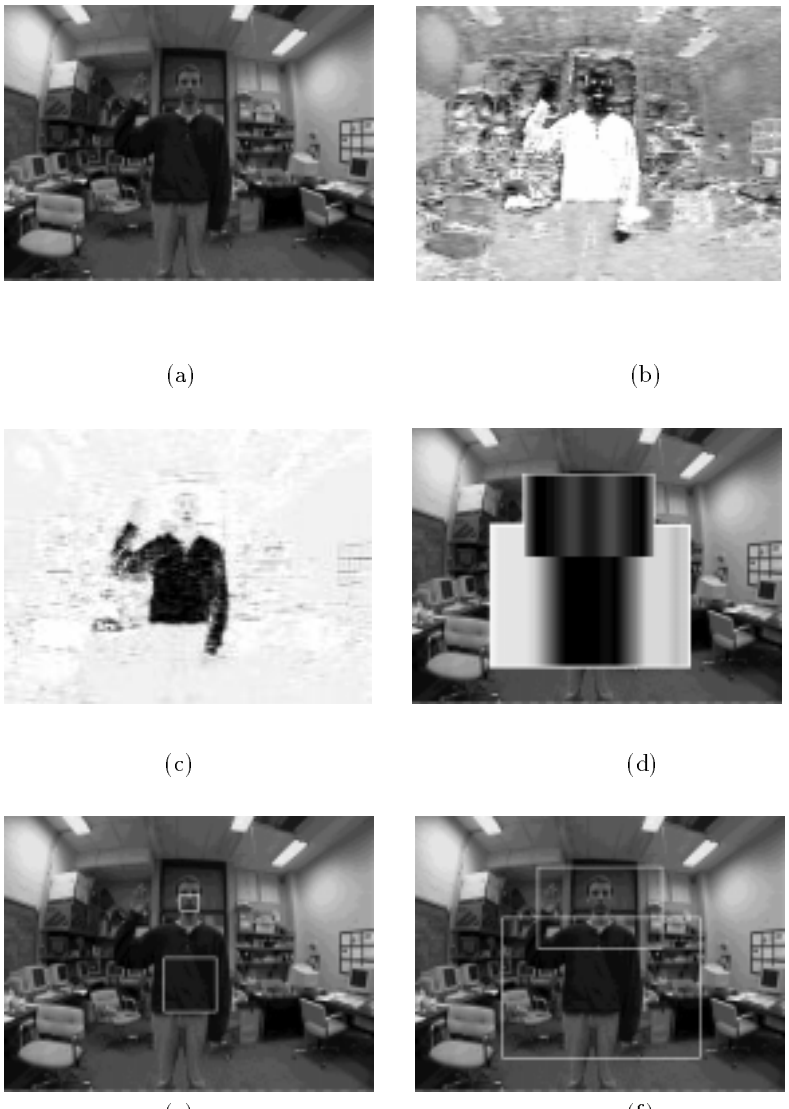


Figure 2. Tracking a person: Raw camera image (a), face-color filtered image (b) and shirt-color filtered image (c). Projection of the filtered image onto the horizontal axis (within a search window) (d). Face and shirt center, as used for tracking and adaptation of the filters (e). Search window, in which the person is expected to be found in the next image (f).

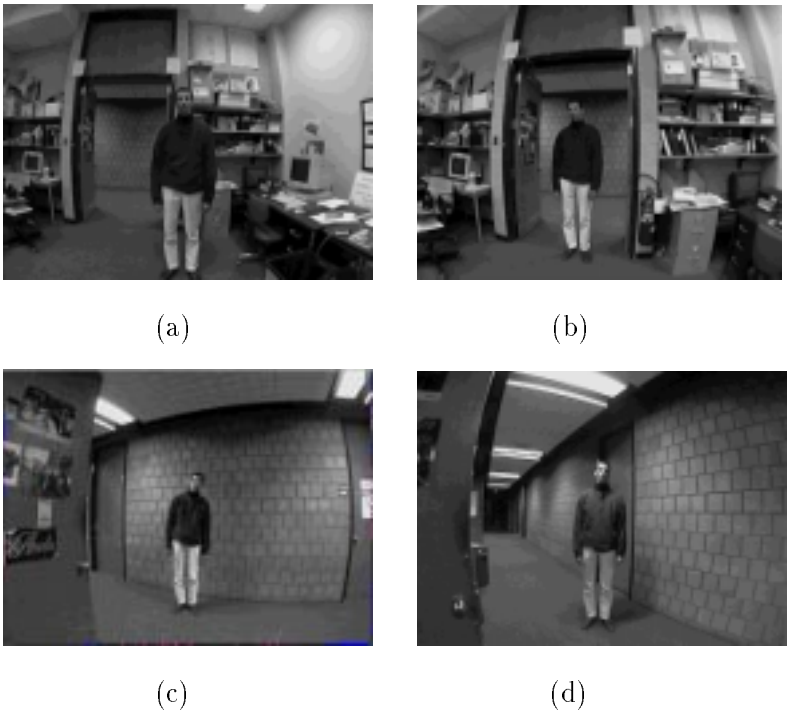


Figure 3. This sequence of camera images, recorded in a single run, illustrates the potential speed and magnitude of changes in lighting conditions, which make gesture recognition on mobile robots difficult.

our color model, the higher is the response to the color filter and hence the darker the pixel is in the image.

2.2. Adaptation

When the robot moves through its environment (e.g., following a person), the appearance with respect to color changes due to changing lighting conditions. Figures 3(a) to 3(d) show images that were recorded while the robot followed a person through a building. Note the different illumination of the person in the lab and in the hallway. Even under the same lighting conditions, changes in background colors may influence face-color appearance. Similarly, if the robot follows the person, the apparent face color change as the person's position changes relative to camera or lighting. Consequently, it is essential for the tracking system to cope with different lighting conditions.

Our approach adapts the color model on-the-fly. In particular, it adapts the face model using a leaky integrator (filter with exponential

decay):

$$\Sigma^t_{\text{face}} = \alpha\Sigma^*_{\text{face}} + (1-\alpha)\Sigma^{t-1}_{\text{face}} \quad (13)$$

$$\mu^t_{r_{\text{face}}} = \alpha\mu^*_{r_{\text{face}}} + (1-\alpha)\mu^{t-1}_{r_{\text{face}}} \quad (14)$$

$$\mu^t_{g_{\text{face}}} = \alpha\mu^*_{g_{\text{face}}} + (1-\alpha)\mu^{t-1}_{g_{\text{face}}} \quad (15)$$

The shirt color model is adapted in the same fashion:

$$\Sigma^t_{\text{shirt}} = \alpha\Sigma^*_{\text{shirt}} + (1-\alpha)\Sigma^{t-1}_{\text{shirt}} \quad (16)$$

$$\mu^t_{r_{\text{shirt}}} = \alpha\mu^*_{r_{\text{shirt}}} + (1-\alpha)\mu^{t-1}_{r_{\text{shirt}}} \quad (17)$$

$$\mu^t_{g_{\text{shirt}}} = \alpha\mu^*_{g_{\text{shirt}}} + (1-\alpha)\mu^{t-1}_{g_{\text{shirt}}} \quad (18)$$

Here, Σ^*_{face}, $\mu^*_{r_{\text{face}}}$ and $\mu^*_{g_{\text{face}}}$ denote values that are obtained from the most recent image only, whereas $\Sigma^{t-1}_{\text{face}}$, $\mu^{t-1}_{r_{\text{face}}}$ and $\mu^{t-1}_{g_{\text{face}}}$ are values of the model at a previous time-step (same for the face model). The learning rate α, which we set to 0.1 in all our experiments, determines how quickly the model adapts to changes in lighting conditions. It determines the rate at which the system "forgets" information. For example, after

$$k = \left\lfloor \frac{\log .5}{\log(1-\alpha)} \right\rfloor = -\left\lceil \frac{1}{\log_2(1-\alpha)} \right\rceil \quad (19)$$

iterations approximately 50% of the initial filter coefficients are "forgotten". Thus, α trades off the algorithm's ability to rapidly adapt to new situations (α close to 1) with the ability to memorize past color observations (α close to 0). In our experiments reported below, we found that adapting the color models is absolutely crucial for robustly tracking a person with a moving camera.

2.3. Finding People

Our approach also detects people to learn an initial color mode and initiate the tracking. This is achieved through a generic face color model M_{face}, which is a high-variance model that accommodated a range of "typical" face colors. to be tracked by the robot, a person has to present herself in the center of the robot's cameras. When the robot is not tracking a specific person, it constantly screens a fixed, central region in the image. Once a region with face color has found, which is detected by applying a fixed threshold to the face color filter response in the target area, the face's center is assumed be the centroid of the face color in the region. The initial means of the face model, the means μ_r, μ_g and covariance Σ, is then obtained from a small region surrounding the

estimated face center, according to Equations (4), (5), and (6) Similarly, the shirt color model is initialized using pixels in a small, rectangular image region at a fixed distance under the face. The location of this rectangular image region corresponds to a region 50cm below a person's face center under the assumption that the person's distance to the robot is approximately 2.5 meter. Due to the size of shirts, we found the placement of this rectangular region to be very robust to variations in people's heights and distance to the robot's camera. Thus, in practice a person has to present himself in the center of the robot's camera image. This is only necessary in the initial step; during tracking, the person's relative position in the camera's field of view is unconstrained.

Examples of both windows W_{face} and W_{shirt} are depicted in Figure 2(e). Note that no previous knowledge about the person's shirt color is necessary in order to find the person. The system acquires its initial shirt color model based on a region below the face. Thus, it can locate and track people with arbitrary shirt colors, as long as they are sufficiently coherent, and as long as they are vertically aligned. While uni-color shirts work best, the approach can also cope with multi-colored shirts, in which its adaptive color model tends to specialize on one of the colors—with some degradation in performance.

2.4. Tracking People

In the initial task of finding the person, the center of the face and then the center of the shirt are determined. However, treating each filter response independently in order to track the person is not as advantageous as combining the responses of both filters. Consequently, it is still possible to track the person even if the response of one filter is not as good as expected.

The locating of a person in the image is now performed in two steps: In the first step, the *horizontal* coordinates of the person in the image is determined. Secondly, the person's *vertical* coordinate, that is the vertical center of the face and shirt, are determined.

More specifically, in the first phase our system searches for co-occurrences of vertically aligned face and shirt color. This step rests on the assumption that the face is always located above a person's shirt. The color filter's response is summed horizontally over m neighbors, and the average over an entire column is computed. Figure 2(d) shows the resulting two vectors, which are expanded vertically (for the reader's convenience). The grey-level in the two regions S_{face} and S_{shirt} ggraphically indicates the horizontal density of face and shirt color, respectively. The darker the pixel, the better the match. Finally, both vectors are multiplied component-wise. To determine the estimated

(a) (b)

Figure 4. Example gestures: 'Stop' gesture (a) and 'Follow' gesture (b). While the 'Stop' gesture is a static gesture, the 'Follow' gesture involves motion, as indicated by the arrows.

horizontal coordinate of the person, the maximum value of this product is determined and its index is taken as the person's x coordinate.

The remaining problem of finding the vertical coordinates of face and shirt is now a search in a one-dimensional space. Here our algorithm simply returns (after local smoothing) the location of the highest response, subject to the condition that the face must be above the shirt (and not below).

To decrease the amount of computation per image, the search is restricted to a small window centered around the position where the person was previously observed. Since our approach tracks two colors, two search windows are constructed around the person's face and body: S_{face} and S_{shirt}. Examples of these windows are shown in Figure 2(f).

3. Gesture Recognition

The primary goal of this research is develop and evaluate a vision-based interface that is capable of recognizing both *pose gestures* and *motion gestures*. Pose gestures involve a static configuration of a person's arm, such as the 'Stop' gesture shown in Figure 4(a), whereas motion gestures are defined through specific motion patterns of the arm, such as the 'Follow' gesture shown in Figure 4(b).

The recognition of gestures is carried out in two phases. In the first phase, arm poses are extracted from individual camera images. We will refer to this process as *pose analysis*. In the second phase, temporal sequences of arm poses are analyzed for the occurrence of a gesture. This process will be referred to *temporal template matching*.

Figure 5. Examples of pose templates. The excitatory region is shown in black and the inhibitory in grey. White regions are not considered in the matching process.

3.1. Pose Analysis

Our approach uses two different methods for pose analysis: Neural networks and correlation-based template matching. The use of two methods, instead of one, is motivated by our desire to understand the relative benefits of either approach. Further below, in Section 4.4, we will present empirical results comparing both approaches.

Both approaches have in common that they operate on a color-filtered sub-region of the image which contains the person's right side, as determined by the tracking module. In other words, they rely on the tracking module to segment the image. Their input is a section of the image which contains the right upper side of a person's body, computed using the coordinates obtained when tracking the person. Both approaches also have in common that they output a probability distribution over arm poses. They differ in the way the segmented image is mapped to a probability distribution.

3.1.1. *Graphical Template Matcher*

The graphical template matcher (also called: *pose template matcher*) compares images to a set of pre-recorded templates of arm poses, called *pose templates*. More specifically, the color-filtered image is correlated with a set of R pre-recorded templates of arm poses. Each pose template $\mathbf{T}$ consists of two regions: an *excitatory* region, which specifies where the arm is to be expected for a specific pose, and an *inhibitory* region, where the arm should *not* be for this pose.

Figure 5 shows examples of pose templates. Here excitatory regions are shown in black, and inhibitory regions are shown in grey. The templates are constructed from labeled examples of human arm poses (one per template), where the excitatory region is extracted from the largest coherent region in the filtered image segment, and a simple geometric routine is employed to determine a nearby inhibitory region. Consequently, a pixel $p \in \mathbf{T}$ can have the following values

$$p = \begin{cases} +1 & \text{Excitatory region} \\ -1 & \text{Inhibitory region} \\ 0 & \text{Irrelevant} \end{cases} \tag{20}$$

The correlation $\varrho_{\mathbf{TI}}$ between a pose template $\mathbf{T}$ and the response $\mathbf{I}$ to shirt color model is defined by the correlation coefficient

$$\varrho_{\mathbf{TI}} = \frac{\sigma_{\mathbf{TI}}}{\sigma_{\mathbf{T}}\sigma_{\mathbf{I}}} \tag{21}$$

and is computed as follows:

$$\varrho_{\mathbf{TI}} = \frac{\sum_{p\in\mathbf{T}}(p-\mu_{\mathbf{T}})(p_{\mathbf{I}}-\mu_{\mathbf{I}})}{\sqrt{\sum_{p\in\mathbf{T}}(p-\mu_{\mathbf{T}})^2\sum_{p\in\mathbf{T}}(p_{\mathbf{I}}-\mu_{\mathbf{I}})^2}} \tag{22}$$

where $p_{\mathbf{I}}$ denotes a pixel in the filter response with the same coordinates as the pixel p in the pose template. The means $\mu_{\mathbf{T}}$ and $\mu_{\mathbf{I}}$ are only computed in the area of the pose template. In regions where we expect the response of the shirt color model to be high, the value of a pixel in the pose template is one. In inhibitory regions (value is -1), on the other hand, the response of the shirt color model to expected to be low. Note that the terms $(p-\mu_{\mathbf{T}})$ in the numerator and $\sum_{p\in\mathbf{T},p\neq 0}(p-\mu_{\mathbf{T}})^2$ in the denominator only have to be determined once, whereas all other computations have to be made for each camera frame.

The result of correlating the image segment with R pre-recorded templates is a correlation vector, or *feature vector*, of R components—one for each pose template as depicted in Figure 6. An example is shown in the center of Figure 6. Here a component of the vector is represented by a square whose size is associated with the magnitude and whose color indicates the sign (black is negative, white is positive; this example happens to be non-negative). For each new image frame, the feature vector is computed as described above. These feature vectors form the basis of the temporal template matching.

3.1.2. *The Neural Network-Based Method*

Alternatively, one might use an artificial neural network to recognize poses. The neural network-based approach predicts the angles of the two arm segments relative to the person's right side from the image segment. The input to the network is the image segment, down-sampled to a 10 by 10 matrix (Figure 8). The output corresponds to the angles of the arm segments, encoded using multi-unit Gaussian representations (Figure 7), just like in Pomerleau's ALVINN (Pomerleau, 1993). More specifically, the Gaussian output encoding uses multiple units to encode a single scalar value, by generating Gaussian-like activations over the array of output units. Like Pomerleau, we found that those representations gave the best results among several ones that we tried during the course of this research.

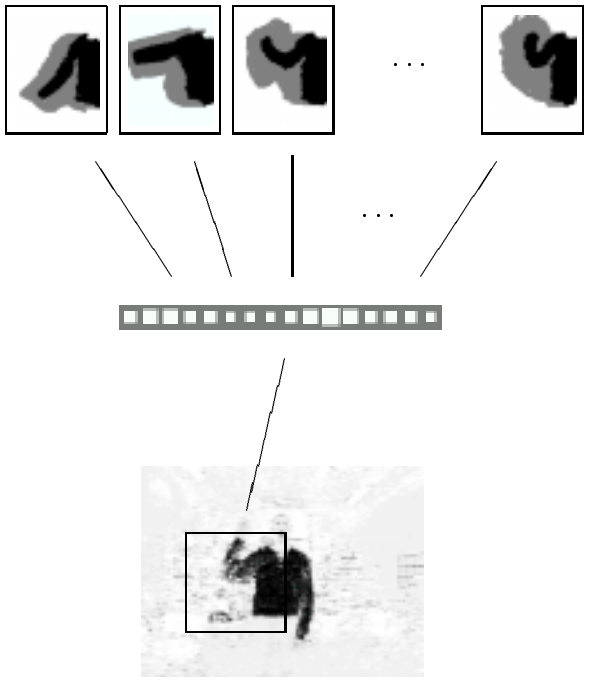


Figure 6. The pose template matching process between the response of the shirt color model and each of the arm-position templates in the database.

In our implementation, the network was trained using Backpropagation (Rumelhart *et al.*, 1986), using a database of 1708 hand-labeled training images. Training labels consisted of the two angles of the two arm segments relative to the image plane, which were specified using a graphical interface in less than an hour.

In our implementation, the neural network possesses 60 output units, 30 of which encode the angle between the arm upper segment (angle α) and the vertical axis, and 30 measure the angle between the arm lower segment (angle β) and the horizontal axis. Both values are encoded using Gaussians. As an example, Figure 7 shows two different angles: Angle1 = 124° and Angle2 = 224° encoded through the following Gaussian function:

$$y_i = e^{-c*d_i^2}$$

where c is a constant ($c = 50$ in our experiments), $d_i = \frac{angle}{360} - \frac{i}{29}$ for $i = 0, 1, \ldots, 29$ and $angle \in \Re$.

Figure 8 shows an example of the network's input, output, and target values. The input is a down-sampled, color-filtered image of size 10 by 10. The output is Gauss-encoded. The nearness of the outputs (first

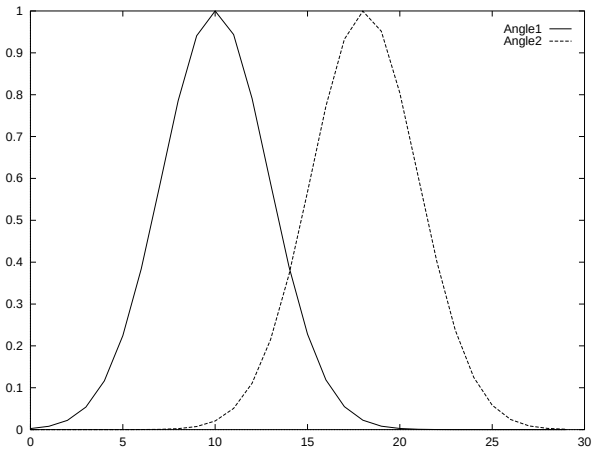


Figure 7. Gaussian output encoding for a pair of angles.

and third row) and the targets (second and forth row) suggests that in this example, the network predicts both angles with high accuracy.

Figure 9 shows a filtered example image. Superimposed here are the two angle estimates, as generated by the neural network.

To recover the angle estimates from the Gaussian encoding, the network's output is converted into two angles by fitting Gaussians to the network's output units as described in (Pomerleau, 1993). More specifically, for both sets of 30 output units our approach determines the best fitting fixed-variance Gaussian:

$$a^* \;=\; arg_a min\{\Sigma_i\ [y_i - e^{-\frac{(x_i - a)^2}{\sigma^2}}]\}$$

The network's average error for the angles α and β, measured on an independent set of 569 testing images, is shown in Table I. In these experiments, three different network topologies were tested, using vanilla Backpropagation and a learning rate of 0.025 (which was empirically determined to work best).

Both pose analysis methods, the neural network-based method and the pose template matcher, generate multi-dimensional *feature vectors*, one per image. To make sure both methods adhere to the same output format—which will be essential for the empirical comparison, the network output is transformed analytically into the output format of the template-based algorithms. Recall that our template matcher uses 16 different pose templates (c.f., Section 3.1.1) and Figure 6). To obtain the same representation for the neural network-based technique, a vector of 16 components is constructed where each component corresponds to the Euclidean distance between the pair of angles generated by

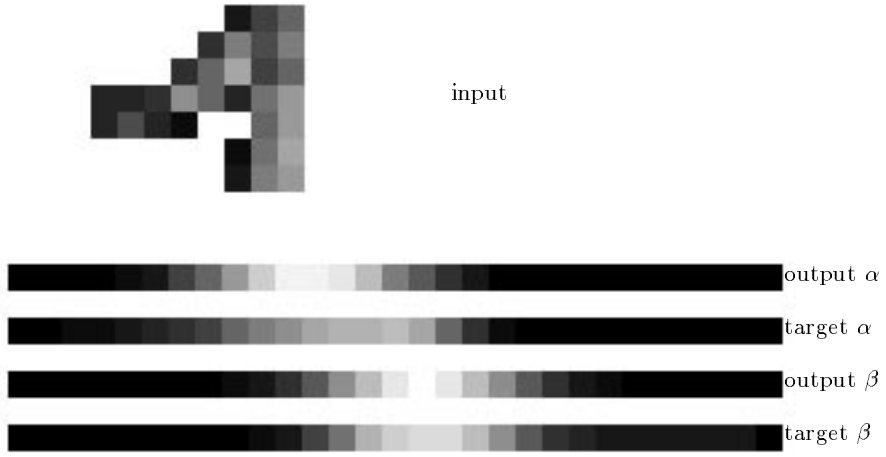


Figure 8. The input to the neural network, a down-sampled, color-filtered image of size 10 by 10, and the outputs and targets of the network for the two angles.

network's output and a pair of angles of the pose template (Figure 9). This "trick" enables us to compare the graphical template matcher and the neural network-based approach empirically, as described below.

3.2. Temporal Template Matching

The temporal template matcher integrates multiple camera images, to detect gestures defined through sequences of arm poses (static and dynamic). It compares the stream of feature vectors with a set of prototypes, or *gesture templates*, of individual gestures that have been previously recorded. Figure 10 shows examples of gesture templates, for the gestures 'Stop' (Figure 4(a)) and 'Follow' (Figure 4(b)). Each of these templates is a sequence of prototype feature vectors, where time is arranged vertically.

As can be seen in this figure, the 'Stop' gesture is a pose gesture (hence all feature vectors look alike), whereas the 'Follow' gesture involves motion. Our approach treats both types of gestures the same. Both types of gestures—pose gestures and motion gestures—are encoded through such temporal templates.

To teach the robot a new gesture, the person presents itself in front of the robot and executes a gesture several times (e.g., five times), in prespecified time intervals. Our approach then segments these examples

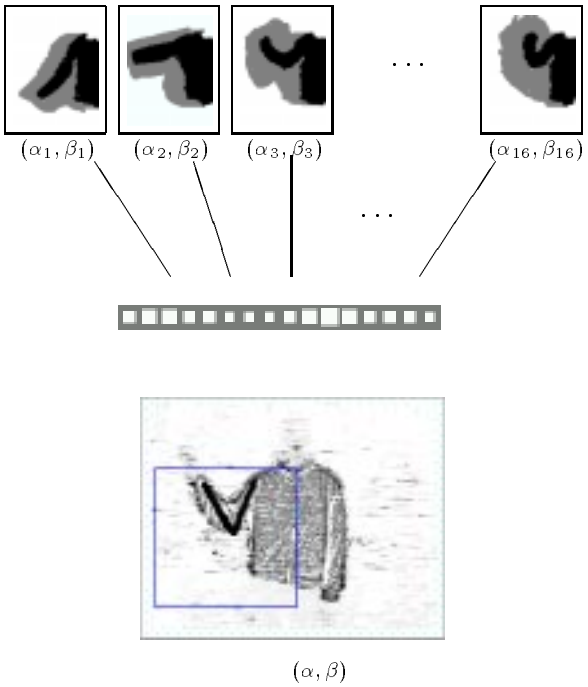


Figure 9. The angles matching process between the response of the shirt color model and each pair of angles of the arm-position templates in the database.

into pieces of equal length, and uses the average feature vector in each time segment as a prototype segment. Figure 11 shows the segmentation process—for simplicity the figure only depicts one training example. Note that this process compresses an observed gesture by a factor of four: For example, a gesture that lasted 60 frames is represented by 15 feature vectors.

To compensate differences in the exact timing when a person performs a gesture, our approach uses the Viterbi algorithm for time alignment. The Viterbi alignment, commonly used in the context of Hidden Markov Models (Rabiner and Juang, 1986), employs dynamic programming to find the best temporal alignment between the feature vector sequence and the gesture template. It has become highly popular in the speech recognition community (cf. (Waibel and Lee, 1990)), since it compensates for variations in the timing of spoken language (and gestures).

In our approach, the Viterbi algorithm continuously analyzes the stream of incoming feature vectors for the possible presence of gestures. It does this by aligning the observation stream $\mathbf{o}$ with each gesture

Table I. Average error obtained by neural network in the testing data

Topology	Upper arm segment	Lower arm segment
100-200-60	4.85^o	5.24^o
100-100-60	4.73^o	5.46^o
100- 50-60	4.96^o	5.56^o

template $\mathbf{p}$ (of size N, i.e., it contains N compressed feature vectors) using a matrix $(e)_{k,j}$.

The matrix $(e)_{k,j}$ is of size $T \times N$, and is computed from the top left to the bottom right as follows:

$$e_{k,j} = \begin{cases} ||\mathbf{o}_k - \mathbf{p}_j|| & k = 0 \wedge j = 0, \\ e_{k,j-1} & k = 0 \wedge j \neq 0, \\ e_{k-1,j} + ||\mathbf{o}_k - \mathbf{p}_j|| & k \neq 0 \wedge j = 0, \\ e_{k,j-1} & e_{k,j-1} < e_{k-1,j} + ||\mathbf{o}_k - \mathbf{p}_j||, \\ e_{k-1,j} + ||\mathbf{o}_k - \mathbf{p}_j|| & e_{k,j-1} \geq e_{k-1,j} + ||\mathbf{o}_k - \mathbf{p}_j||. \end{cases} \quad (23)$$

This dynamic programming equation computes the likelihood of a gesture under *optimal* time alignment: Consequently, the final element $e_{T,N}$ determines the sequence's likelihood conditioned on the gesture template. Our approaches uses the Euclidean distance to compute the difference $||\mathbf{o}_k - \mathbf{p}_j||$ between a feature vector of the observation and a feature vector of the gesture template.

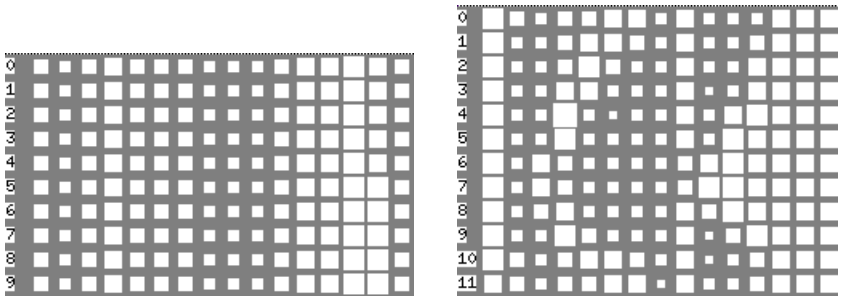
To summarize, time variations in gesture template matching are accommodated using the Viterbi algorithm. This algorithm aligns the actual observation sequence and the templates *optimally*, compensating variations in the exact timing of a gesture. The Viterbi algorithm also determines the likelihood of observing this sequence under each gesture template, which is essential for classification of gestures.

4. Experimental Results

Our approach has been implemented on a mobile robot and evaluated in a series of experiments. It also has been integrated into an autonomous robot navigation system, to test its utility in the context of a realistic service robotics task.

The central question underlying our experiments are:

- **Robustness:** How robust is the system to variations in lighting conditions, differences in people's individual motion, different shirt colors, and so on?



(a) (b)

Figure 10. Examples of gesture templates. Gesture templates are sequences of prototype feature vectors. Shown here are gesture templates for (a) a 'Stop' gesture (does not involve motion), (b) a 'Follow' gesture (involves motion, as indicated by the change over time).

– **Usability:** How usable is the interface, specifically when used to control a robot in the context of a service task?

Th experimental results presented in this section address these questions systematically. For the reader's convenience, the section is broken down into three parts:

1. **Tracking:** Results are presented for the performance of the people tracker, obtained while following a person through an office building with rooms and hallways. The environment is characterized by drastically varying lighting conditions.

2. **Gesture Recognition:** A comparative experiment evaluating both approaches to gesture recognition is presented. Additional experiments investigate the utility of dynamic motion gestures, in comparison to static pose gestures.

3. **Integration:** Results are reported obtained in the context of a clean-up task, aimed to elucidate the usefulness of our gesture interface in the context of a realistic service robotics task. The clean-up task involves human-robot interaction and mobile manipulation, and shows that our interface can recognize dynamic gestures and responds to them through corresponding actions.

The robot used in our research, AMELIA (Figure 13), is a RWI B21 robot equipped with a color camera mounted on a pan-tilt unit, 24 sonar sensors, and a 180^0 SICK laser range finder. To evaluate the utility of gesture recognition in a real world scenario, it has been integrated into the robot navigation and control software BeeSoft, developed in conjunction with the University of Bonn and RWI (Thrun *et al.*, 1998).

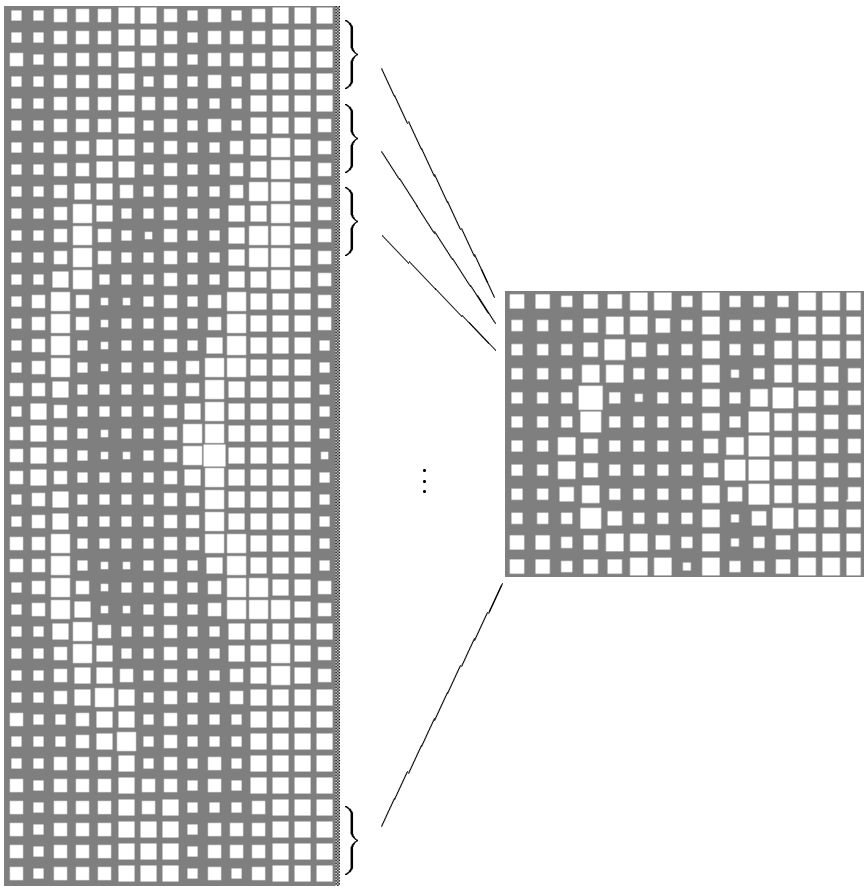


Figure 11. Creating a gesture template by example: a stream of observed feature vectors on the left is compressed into the gesture template on the right.

4.1. Implementation Details

Our approach was implemented on a low-end PC (Intel 200 MHz Pentium with 64MB main memory), grabbing images at a rate of 30 fps and a resolution of 320×240 pixels. The major computational bottleneck is the filtering of the image using the Gaussian filters, and the pose analysis (template matching, neural networks). By focusing the computation of the response to the color models to a narrow window around the person, a performance of roughly 10 frames per second was obtained. Furthermore, the Gaussian responses were only computed for every second pixel horizontally and vertically, dividing the computational effort by a factor of four. Adapting the covariances and means in the regions W_{face} and W_{shirt}, both of fixed size, does not pose a computational problem since both windows are relatively small.

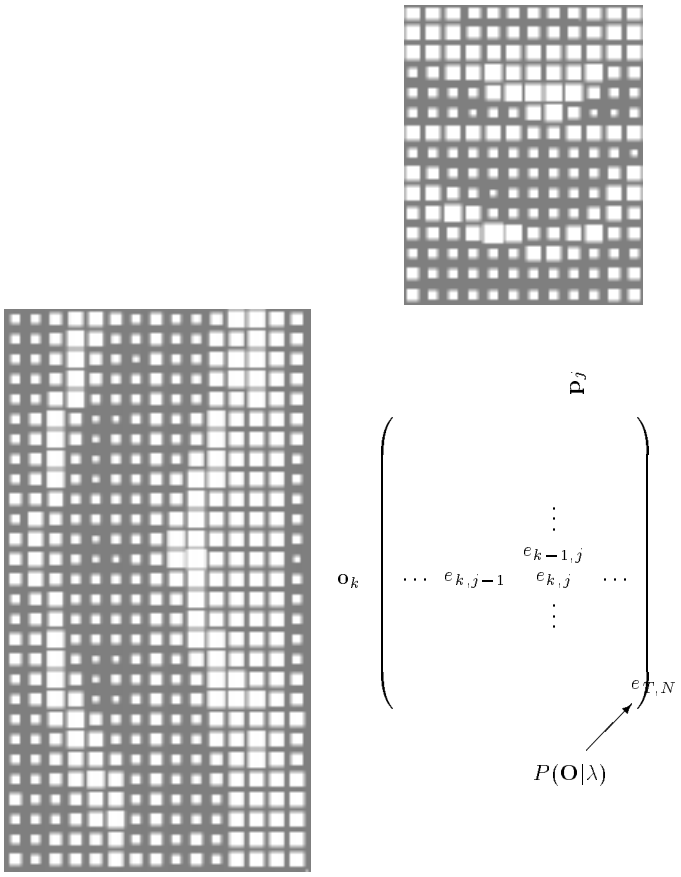


Figure 12. Aligning the last T observations with a gesture template. Shown on the left are the last T feature vectors that are temporally aligned, using the matrix $(e)_{k,j}$, with the gesture template (shown on the top right, rotated 90 degrees counterclockwise).

To cope with slightly different gestures of the same kind, our system uses multiple prototypes for each gestures (2 to 5 per gesture). The time alignment is performed at multiple time scales, each of which corresponded to a specific, assumed total duration of a gesture. The observed feature vectors are constantly analyzed and aligned to a set of gesture templates using the Viterbi algorithm, as described in Section 3.2. A gesture is detected whenever the result $e_{T,N}$ of the alignment surpasses a certain, pre-specified threshold.

We trained our system to recognize four different gestures:

Stop: The 'Stop' gesture is shown in Figure 14(b). This gesture is a static gesture: Moving the arm into the right position for about a second is sufficient to trigger the stop command.

Figure 13. Amelia, the robot used in our experiments.

Follow: The 'Follow' gesture involves a wave-like motion, moving the arm up and down as shown in Figures 14(d) and (e).

Pointing Vertical: This gesture is also a motion gesture. Starting from the initial arm position depicted in Figure 14(a), the person moves the arm up to the position shown in Figure 14(f), holds it there for a brief moment, and returns to the initial arm position.

Pointing Low: Again, starting in the initial arm position of Figure 14(a), the person points to an object on the floor, as shown in Figure 14(e), and returns to the initial arm position.

The choice of gestures was motivated by the particular service task described further below.

4.2. Tracking

We measured the performance of the people tracker by instructing the robot to follow a person through our environment. An experiment was judged to be successful if the robot managed to track (and follow) the person for 20 meters or more, during which it processed close to 1,000 camera images. The testing environment contains a brightly illuminated lab, and a corridor with low-hanging ceiling lights that have a strong effect on the brightness of a person's face.

We conducted more than 250 experiments involving four subjects with a range of different shirts (see below), many of which during

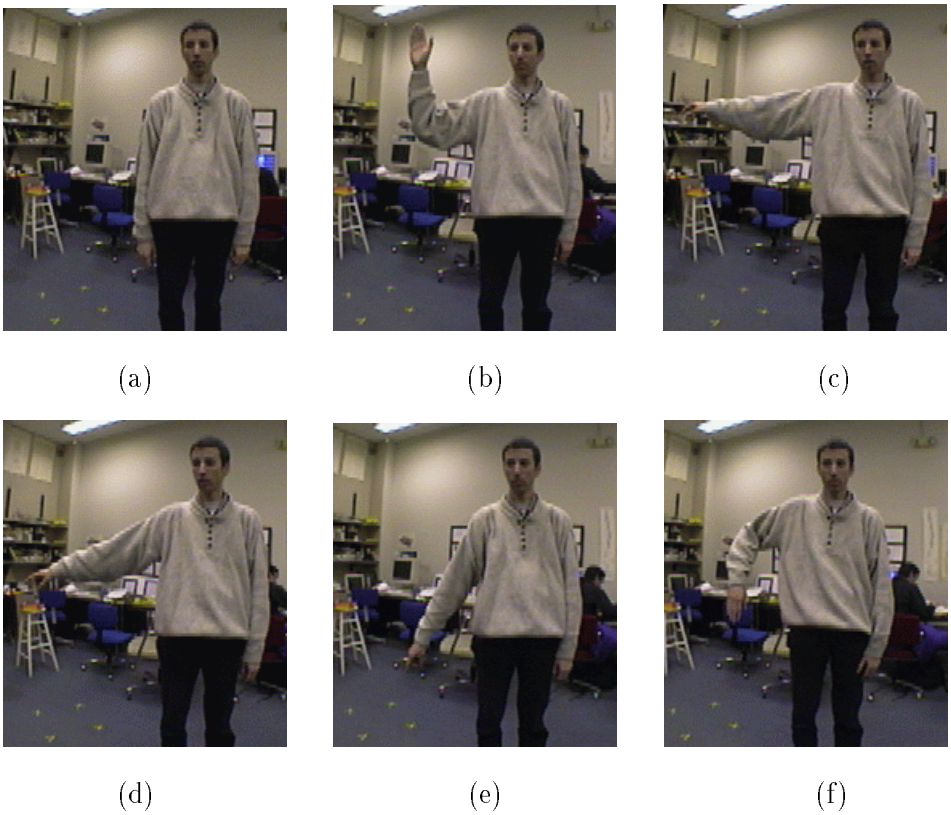


Figure 14. Images of arm positions that are contained in the five example gestures. Each image marks a specific point in time — for example the start, middle or end position of a motion gesture.

system development and demonstrations. In a subset of 82 final experiments carried out after the software development was completed using the same four subjects, we measured a success rate of 95%. An experiment was labeled a success if the robot never lost track of the person. Our results indicate an extremely low per-frame error rate, since a single recognition error typically leads to an overall failure of the tracking module. The identical software (and parameters) was used in each of the tests, regardless of the subject or his/her shirt color. In our experiments, the subjects wore shirts with different colors, including gray, red, green, light blue, white, and black. It comes at little surprise that brightly colored shirts maximize the reliability of the tracking algorithm. The majority of failures were observed for white and black shirts, which are the most difficult colors to track. Here the failure rate was approximately 12%. Due to limited range of CCD cameras, dark regions in the background (e.g., shades) are often perceived as

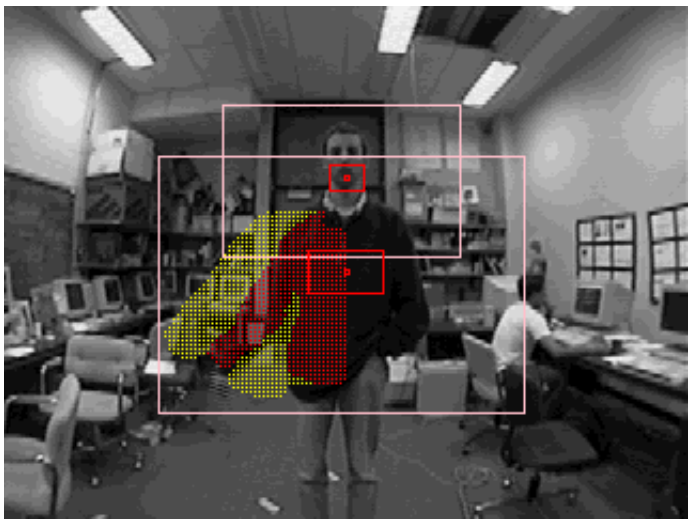


Figure 15. Tracking of the person and pose analysis. Shown here are the search and adaptation windows. Additionally, the pose template whose component in the feature vector has the highest correlation is superimposed on the person.

pitch black, whereas bright spots (ceiling lights, direct sunlight) usually appear white in the image. Nevertheless, the combination of two colors—face and shirt—and the ability to adapt quickly to new lighting conditions made failures rare even for black or white shirts.

Figure 15 shows an example, recorded when tracking a person with a dark shirt. Shown there are the estimated centers of the face and the shirt (small rectangles), along with the location of the search windows (large rectangles). The most likely template is superimposed. Here the person is in the midst of carrying out a 'Pointing low' gesture.

4.3. Single Image Gesture Recognition

We first evaluated static gestures, recognized from a single camera image. The goal of this experiment was to characterize the accuracy of gesture recognition in the extreme case, where only a single image is available—to set the base line for our subsequent analysis of gesture recognition from image streams. The single-image recognition applies our pose template matcher, but does not use Viterbi to match temporal gesture prototypes. Instead, a gesture is recognized when the probability of a specific gesture surpasses a pre-selected (optimized) threshold. Obviously, the natural variance in images affects the classification rate much more than in the multi-image case, where individual image variations have a much lesser effect (and thus yield more accurate

recognition). Figure 15 shows an example pose template superimposed on the person,s image.

In a systematic experiment involving 507 camera images, we found that the correct pose was identified in 85% of the cases; in 10% a nearby pose was recognized, and in only 5% the pose template matcher produced an answer that was wrong. While these results are encouraging, they can be improved by considering a stream of camera images over time (as shown below).

The misclassifications were exclusively due to variations in the image (e.g., when lighting changes mandated an adaptation of the color filters). In a real-world application, it could easily happen that a person moves its arm into a gesture-type position without intending to perform a gesture. Such incidents undoubtedly would further increase the misclassification rate, making it questionable if a single-image gesture recognizer is sufficiently robust for real-world applications. In the next section, we will report results for dynamic gesture recognition from multiple images, illustrating that the latter provides much better results than the static, single-image approach.

4.4. Gesture Recognition

The evaluation of gestures from sequences of camera images is more interesting. Here gestures may be static or dynamic—both are recognized form a sequence of images using the Viterbi algorithm. In this section, we report results for the temporal gesture recognition system, including a comparison of the two different methods for pose analysis: the graphical template matcher and the neural network-based approach.

We evaluated both approaches using a database of 241 image sequences. In 191 of these sequences, the person performed one of the four gestures defined above. In the remaining 50 sequences, no gesture was performed. To test the robustness of the system in the extreme, more than half of these sequences were collected while the robot was in motion. In both sets of experiments, the recognition threshold was hand-tuned beforehand using a separate data set. Since false-positives (the robot recognizes a gesture that was not shown by the instructor) are generally worse than false-negatives (the robot fails to recognize a gesture), we tuned our thresholds in such a way that the number of false-positives was small.

Tables II and III show the recognition accuracy obtained for the graphical template matcher and the neural network approach, respectively. Overall, both approaches—the neural network based approach and the pose template matcher—classified 97% of the examples correctly. Both erred in 7 of the 241 testing sequences. The template

Table II. Recognition results for the pose template matcher.

		Gestures recognized				
		Stop	Follow	Point.Vert	Point.Low	No Gesture
Gestures given	241	39	59	45	42	56
Stop	40	39	-	1	-	-
Follow	59	-	59	-	-	-
Point.Vert	50	-	-	44	-	6
Point.Low	42	-	-	-	42	-
No Gesture	50					50

matcher's most prominent error was a failure to recognize 'Pointing Vertical' gesture, which we attribute to a poor set of motion templates for this gesture. The neural network's poorest gesture was the 'Follow' gesture, which sometimes was detected accidentally when the person gave no other gesture, and sometimes was mistaken for a 'Pointing Low' gesture. The different failures of both approaches suggest that combining both—template matching and neural networks—should yield better results than either approach in isolation. However, a combined approach increases the computational complexity, reducing the frequency at which images are analyzed. Since near-frame-rate is essential for successful tracking, we decided not to integrate both approaches.

As an example, Figure 16 shows the recognition results for three different classes of gestures: 'Stop', 'Follow' and 'Pointing', for one of the testing sequences. Here the horizontal axis depicts time (frame number), while the vertical axis shows the probability estimate of each gesture template. While the 'Stop' class contains three different templates (dstop1, dstop2 and dstop3), both the 'Follow' and 'Pointing' class contain only two templates. The system recognized a 'Stop' gesture at point (a) and 'Follow' gesture at point (b).

Figure 17 shows the output of the pose template matcher over time, for a subset of the image sequence (frames $t = 400$ to $t = 508$). This figure should be compared to Figure 10, which depicts a prototype template for the 'Stop' gesture, and one for the 'Follow' gesture. The 'Follow' gesture, which is executed in this subsequence, is correctly detected.

4.5. Clean-Up Task

The final experiment evaluates the gesture recognition interface in the context of a realistic, real-world service robot task. The task we chose

Table III. Recognition results for the neural network-based algorithm.

		Gestures recognized				
		Stop	Follow	Point.Vert	Point.Low	No Gesture
Gestures given	241	39	59	50	45	48
Stop	40	39	-	-	-	1
Follow	59	-	56	-	3	-
Point.Vert	50	-	-	50	-	-
Point.Low	42	-	-	-	42	-
No Gesture	50	-	3	-	-	47

was motivated by the "clean-up an office" task, designed for the AAAI-94 mobile robot competition (Simmons, 1995), in which an autonomous robot was asked to locate and retrieve trash items scattered around an office-like environment (various subsequent competitions had similar themes). Our scenario differs in that a human interacts with the robot and initiates the clean-up task, which is then performed autonomously by the robot. The particular task at hand involved a human instructing a robot, through gestures, to follow him to the location of trash. The robot then picks the trash up and delivers it to a nearby trash bin.

In detail, the clean-up task involves the following gesture operations:

- perform a following gesture to initiate the follow behavior,
- perform direct motion commands, e.g., stopping the robot,
- guide the robot to places which it can memorize,
- point to objects on the floor, and
- initiate clean-up tasks, in which the robot searches for trash, picks it up, and delivers it to the nearest trash-bin.

If the robot is shown a 'Stop' gesture, it immediately stops. If the person points towards an object on the ground, the robot starts searching for an object within its visual field. If the robot succeeds, it moves towards the object, picks the object up, and then returns to the nearest trash-bin. The location of trash-bins is known to the robot.

To perform a task of this complexity, the gesture interface was integrated into the BeeSoft robot control architecture, previously developed at our lab in collaboration with Real World Interface Inc (now a division of IS Robotics) (Thrun *et al.*, 1998). In a nutshell, our navigation system enables robots to navigate safely while acquiring maps of unknown

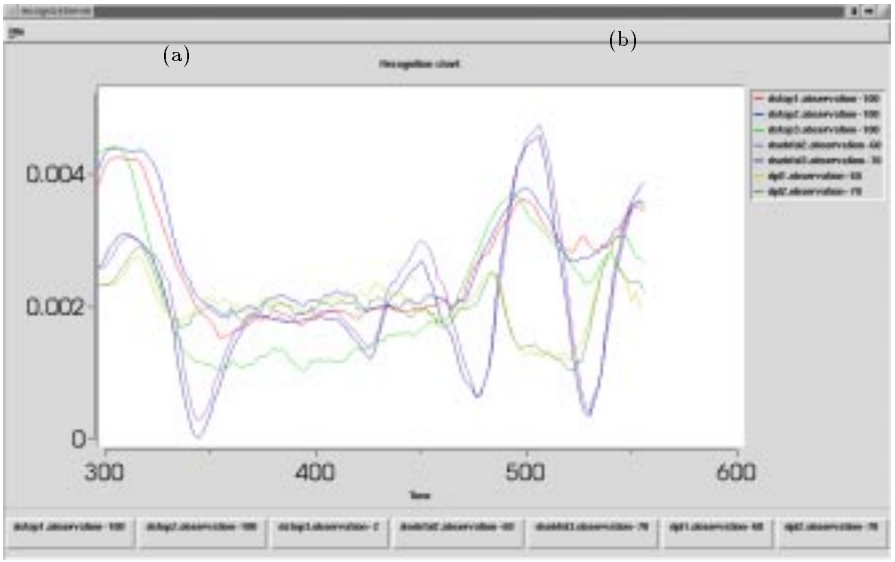


Figure 16. Results $e_{T,N}$ of the matching process over time, normalized to the length of each template. The system recognized a 'Stop' gesture at point (a) and 'Follow' gesture at point (b)

environments. A fast motion planner allows robots to move from point to point or, alternatively, to explore unknown terrain. Collisions with unexpected obstacles are avoided by a fast, reactive routine, which sets the velocity and travel direction of the robot according to periodic measurements of the robot's various proximity sensors (laser, sonar, infrared, tactile). While the robot is in motion, a probabilistic localization method continuously tracks the robot's position, by comparing sensor readings with the learned map of the environment. Permanent blockages lead to modifications of the map, and thus to new motion plans.

Our previous approach has been demonstrated to let robots navigate safely with speed of more than 150 centimeter per second, through densely populated environments. Previous versions of the software won a second, and a first, price at the 1994, and 1996, AAAI mobile robot competitions, respectively. Most recently, our software was the backbone of two "interactive robotic tour-guides" (Rhino and Minerva), which were successfully installed in the Deutsches Museum Bonn (Germany) (Burgard *et al.*, 1999) and in the Smithsonian's National Museum of American History in Washington, DC (Thrun *et al.*, 1999). The robots successfully gave tours to thousand of visitors, at an average speed of over 30 centimeter per second.

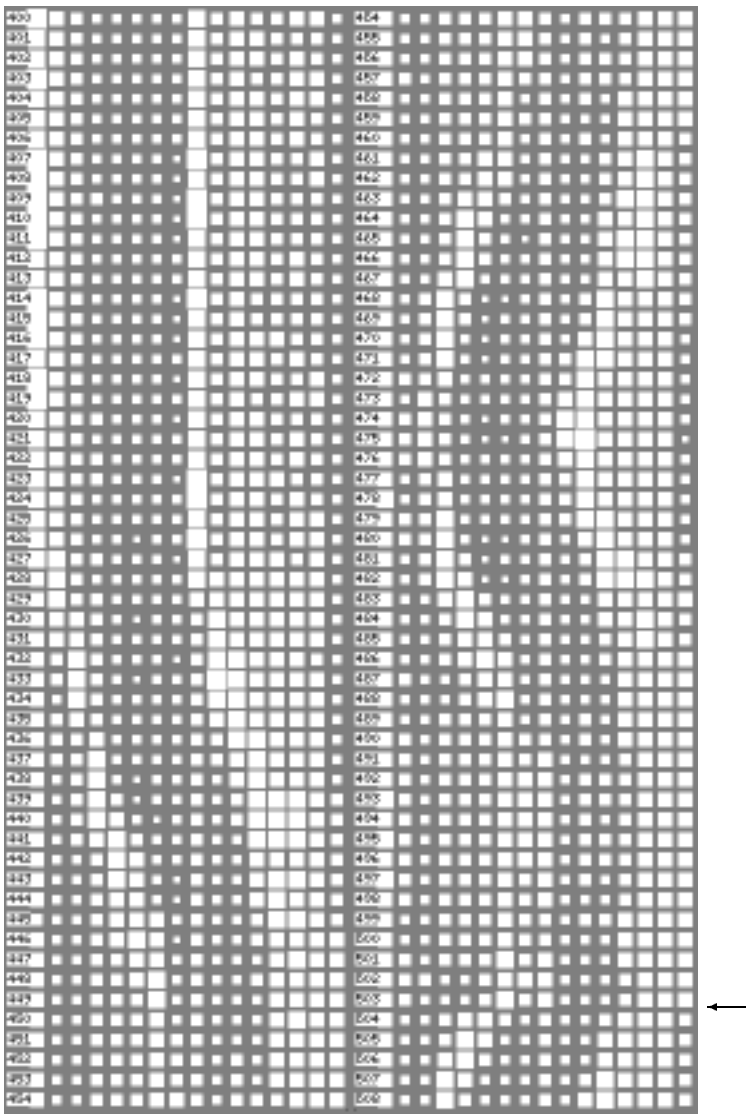


Figure 17. Example of a feature vector stream that contains a 'Follow' gesture. The arrow indicates the point in time at which 'Follow' was recognized (see Figure 16). One of the gesture templates used in the matching is depicted in Figure 10(b), the output of the temporal template matcher is shown in Figure 16.

One of the primary deficiencies of our previous system was a lack of a natural human-robot interface. To instruct the robot, we basically had to program it by hand. This is of course not really acceptable for service robots, since many of them will necessarily be instructed and operated by non-expert users. Thus, the new gesture based interface,

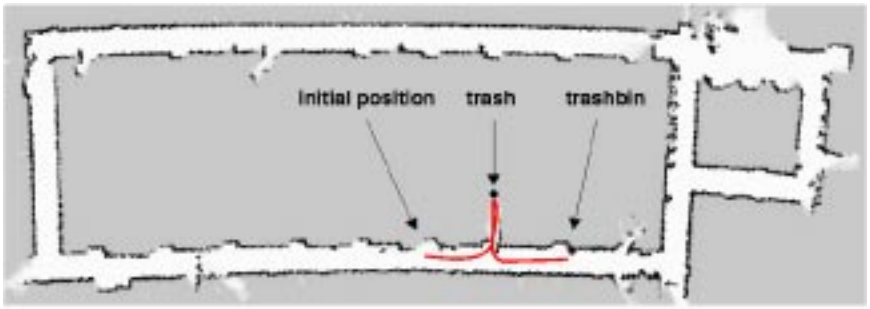


Figure 18. Map of the robot's operational range (80×25 meters) with trace of a specific example of a successful clean-up operation. The robot waited in the corridor, was then guided by a human into a lab, where it picked up a can and later deposited it into a trash-bin.

in combination with our navigation methods, provides a new level of usability to our overall system.

4.6. Integration Results

Figure 18 shows an example run, in which Amelia is instructed to pick up a soda can. Shown there is a map of the robot's environment, constructed using an occupancy grid technique (Elfes, 1989; Moravec, 1988; Thrun, 1998), along with the actual path of the robot and the (known) location of the trash-bin.

In this example, the robot initially waits in the corridor for a person. The person instructs the robot to follow him into the lab using the 'Follow' gesture. The robot follows the person at a speed of no more than 25cm/sec, simultaneously avoiding collisions with nearby obstacles. In the lab, the person stops the robot by invoking the 'Stop' gesture. Within a second, the gesture is recognized and the robot stops. If there is trash nearby, the person now points at a piece of trash (a soda can) on the floor. The robot then uses its visual routines (Margaritis and Thrun, 1998) to determine the exact location of the soda can. It then deploys its manipulator, navigates to the soda can, picks it up, and navigates back to the corridor where the trash is deposited into the trash-bin. The task was conveniently encoded using a finite state automaton.

Our gesture interface worked extremely reliably. In 9 tests of the full clean-up tasks, we did not observe a single failure. In our subjective judgment, we also found the interface easy to use. Operating the robot at low speed (25 cm per second) made it easier to instruct the robot, since it gave the person more time to perform the 'Stop' gesture while the robot was in motion. Tests at higher speeds (45 cm per second)

made it difficult to reliably position the robot close to a soda can, even though our approach can flawlessly manage this speed. Obviously, the person must know the gestures in advance. However, the gestures were similar to those used for communication between humans (e.g., in heavy-noise environment such as airports), making is easy for people to learn the gestures or, alternatively, teach the robot a new set of gestures. Thus, our overall assessment of the clean-up task is that the gesture based interface provides a flexible and easy-to-use interface for the command and control of mobile service robots.

4.7. Failure Modes

While the overall performance of the gesture interface is encouraging, the reader should notice that it also possesses limitations that, if not met, lead to the failure of the gesture interface. In particular:

- The tracking algorithm assumes that the person's face is visible at all times. This is unproblematic when a person is interacting with a non-moving robot. When being followed by a robot, however, our current interface makes it necessary that the person walks backwards. More sophisticated tracking algorithms are necessary to cope with people not facing the robot. For example, a tracking algorithm might simultaneously learn a model of a person's hair color, so that tracking can be continued when a person turns around.

- The current gesture interface is not scaling invariant. Due to the extremely limited computational power on a mobile robot, the size of the image and that of the template is fixed. As a result, the person has to stay within a constant distance to the robot (currently between 2 to 3 meters). The recognition rate drops when the distance to the robot is too small or too high. This limitation could be overcome by automatically scaling the gesture templates or by using a motorized zoom lens. A person's distance is easily recognized using the robot's proximity sensors.

- We do not expect the interface to be sufficiently robust to function in dense crowds, such as the ones Rhino and Minerva faced in their museums. This is because of the difficulty of tracking people in crowds, where many similar-looking faces might confuse the tracking routine. Recent progress of head tracking with occlusion (MacCormick and Blake, 1999) might provide an answer. However, testing the interface in such situations is beyond of the scope of the current paper.

5. Related Work

The idea of using gestures to instruct a mobile robot is not new.

In (Huber and Kortenkamp, 1995) and (Kortenkamp *et al.*, 1996) Kortenkamp et al. describe a system that uses stereo vision and optical flow to model people and recognize gestures. The system is capable of recognizing up to six distinct pose gestures such as pointing and hand signals and then interprets these gestures within the context of an intelligent agent architecture.

Gestures are recognized by modeling the person's head, shoulders, elbows, and hands as a set of proximity spaces. Each proximity space is a small region in the scene measuring stereo disparity and motion. The proximity spaces are connected with joints/links that constrain their position relative to each other. The robot recognizes pose gestures by examining the angles between links that connect the proximity spaces. Confidence in a gesture is build up logarithmically over time as the angles stay within limits for the gesture.

In comparison, the gesture recognition system of Kortenkamp et al. runs completely on-board the robot, just like ours. We are also bound to a single camera and do neither use stereo vision nor any dedicated vision hardware to process the camera images. While these issues are only differences in the physical setup of the system, our approach has a key difference that it can recognize motion gestures, whereas Kortenkamp's approach relies on static gestures that do not involve motion.

Both systems have a high level control architecture. Our approach uses the BeeSoft, Kortenkamp et al. have integrated their system into Firby's Reactive Action Package (RAP) ((Firby, 1994)). The RAP system takes into consideration the robot's task, its current state and the state of the world and then selects a set of skills that should run to accomplish the tasks. Kortenkamp et al. have also extended their work to recognize people's faces on a mobile robot architecture (Wong *et al.*, 1995).

The *Perseus system* (Kahn, 1996), (Kahn *et al.*, 1996), (Firby *et al.*, 1995), specifically addresses the task of recognizing pointing gestures. It is capable of finding objects that people point to. Perseus uses a variety of techniques, including feature maps (such as intensity feature maps, edge feature maps, motion feature maps, etc.), to reliably solve this visual problem in non-engineered worlds. Perseus provides interfaces for symbolic higher level systems like the RAP reactive execution system mentioned before.

Perseus has also been applied to an object pickup task. The first part of the task requires recognizing if and where a person is in the scene. Perseus assumes that people are the only moving objects in the scene.

Hence motion can be used to segment the person from the background. Once the image is segmented, the color of the clothing is known and may be used for tracking the person in the future. Knowledge about the task and environment is used at all stages of processing to best interpret the scene for the current situation. Next the important parts of his body, such as the hands and head, are found and tracked. By fusing the abovementioned feature maps, Perseus is able to reliably track parts of the body. A pointing gesture is recognized if the person's arm has moved to the side of the body and remained stationary for a short time. After the person points, the position of the head and hand is used to determine which area is pointed to. This area is searched for an object.

Perseus is also described in (Franklin *et al.*, 1996) in the context of building a robot 'waiter'. In this task, Perseus has been extended to recognize a 'reach' gesture (similar to a pointing gesture) of the person's arm. The person's hand may or may not contain a soda can. Depending on this information and the current context, the robot either delivers a soda can to the person or receives a soda can from the person.

Unlike our approach, Perseus is restricted to the domain of recognizing pointing gestures. These gestures are static configuration of the person's arm and do not include motion gestures. Like our system, Perseus is not only able to detect pointing gestures, but also capable of identifying the object pointed to. While our approach uses only color information for the same task, Perseus uses multiple independent types of information, among them intensity feature maps and disparity feature maps. Since Perseus does not model the person's arm as a whole, the distance between robot and person may vary.

Boehme an colleagues (Boehme *et al.*, 1998a; Boehme *et al.*, 1998b) have developed a similar system, for gesture-based control of their robot Milva. Like most approaches in this field, their approach recognizes static pose gestures only. Their approach is similar to ours in that they use a (different) neural network algorithm for gesture recognition. Most notably, their work goes beyond ours in that their tracking algorithm uses a collection of cues—not just color cues—to track a person and segment the image. In particular, is uses neural networks for face detection (see also (Rowley *et al.*, 1998)), and their approach also analyses the shape of the head-shoulder contour. As a result, we expect that their approach to track people more robustly than the one presented here; however, as the results reported in this paper suggest, tracking of color pairs with adaptive color filters yields very good results in a wide array of situations.

Finally, approaches for recognizing hand gestures with mobile robots can be found in (Cui and Weng, 1996; Triesch and von der Malsburg,

1997; Triesch and von der Malsburg, 1998). For example, Triesch and van der Marlsburg's approach tracks a hand using a stereo camera system, using color, motion, and depth cues for localization of the hand. The hand posture is recognized using an elastic graph matching algorithm. Due a relatively complex scene analysis, their approach is unable to recognize gestures in real-time. It is also not suited for mobile robots, where background and lighting conditions can vary dramatically over very short periods of time. Nevertheless, in the domain of gesture-based robot arm manipulation they achieve remarkable recognition results in scenes with cluttered background.

Gesture interfaces have long been applied to non-moving systems, where cameras are typically mounted in a static location (often on a pan/tilt unit). Here we only review a selected number of related approaches, refer the reader to the rich and decade-old literature on this topic.

Wren et al. have extended the Artificial Live IVE (ALIVE) project (Maes *et al.*, 1997) to produce a model of people for a variety of tasks (cf. (Wren *et al.*, 1997)). The ALIVE project creates a virtual world for a person in which the person can interact with virtual agents. While the person watches his image interacting with animated agents on a television monitor, ALIVE allows the person to interact with the agent by performing gestures such as waving for an agent to come to him.

Pfinder, or Person Finder, models people as connected sets of blobs: one for each arm and leg, the head, the torso, and the lower body. Each blob has a spatial and color Gaussian distribution associated with it. These distributions are used to track the various parts of the person. One interesting application of this blob representation is described by Starner and Pentland in (Starner and Pentland, 1995). The authors have associated the spatial statistics of the users hands, together with Hidden Markov Models, to interpret a 44 word subset of the American Sign Language (ASL). They were able to produce a real-time ASL interpreter with a 99% sign recognition accuracy.

The abovementioned systems have clear differences to the domain of mobile robots. Most of them assume a static background and controlled lighting conditions—assumptions which cannot hold when the robot and thus the camera is moving. Furthermore, processing power and network bandwidth in case of our mobile platform is limited. However, our approach uses similar techniques for coping with the temporal characteristics of a gesture.

In (Wilson and Bobick, 1995a) Wilson and Bobick model gestures as view-point dependent motions of the person's body. Since gestures are intended to communicate information, the authors assume that people will actively try to make them easy to understand by orienting them in

a standard way with respect to the recipient. This premise allows them to use a Hidden Markov Model to model a motion gesture by observing examples of the gesture from a single viewpoint. One major difference to our system is that the location of the person's body (hands, etc.) is known to the system. For example when recognizing gestures that are performed with the hands, the input to the system were the joint angles returned by a DataGlove.

Wilson, Bobick et al. introduce a system that uses the 3D position of the head and hands, as determined by 3D vision system, to recognize 18 different T'ai Chi gestures (Campbell *et al.*, 1996). To cope with the temporal characteristics of the gesture, the authors use a five state linear Hidden Markov Model.

In (Wilson and Bobick, 1995b) and (Wilson *et al.*, 1996) the authors focus on representing the temporal characteristics of a gesture. They present a state-based technique for the summarization and recognition of gestures. Gestures are defined to be a sequence of states in a measurement or configuration space. Again, the authors assume known gesture-related sensory data (e.g., the movement of the hand as measured by a magnetic spatial sensor).

As with Pfinder, limitations to the domain of mobile robots, such as changing lighting conditions, do not apply here. For example, Wilson et al. assume that the background does not change. Unlike our system, they also use dedicated vision hardware to process the stereo images.

6. Conclusions

This article described a vision-based gesture interface for human-robot interaction. A hybrid approach was presented, consisting of an adaptive color-filter and two alternative methods for recognizing human arm gestures: pose template matcher and neural-network based method. The Viterbi algorithm was employed to recognize variable-length motion gestures from streams of feature vectors, extracted from the image. The interface has been integrated into a mobile robot navigation architecture, where it was evaluated in the context of a mobile manipulation task (clean-up) cooperatively carried out with a human instructor.

Our interface goes beyond previous work in that it is capable of recognizing not only static pose gestures, but also dynamic motion gestures, which are commonly used for communication among people. Motion gestures are defined through specific temporal patterns of arm movements, which improve the classification accuracy and reduce the chances of accidentally classifying arm poses as gestures that were not intended as such.

Experimental results illustrated a high level of robustness and usability of the interface and its individual components. The overall error rate was 3%. In our subjective judgment, we found the interface to be well-suited to instruct a mobile robot to pick up trash and deliver it to a trash bin. While this is only an example application designed to test the utility of gestures in human-robot interaction, we conjecture that the proposed interface transcends to a much broader range of upcoming service robots.

There are some open questions that warrant further research. Some of the most prominent limitations of the current approach were already addressed in Section 4.7. For example, the tracking module is currently unable to follow people who do not face the robot. Future research could address algorithms that considering other sources of information, such as shape and texture, when tracking people. Another limitation arose from the fact that our approach works well if the person keeps a fixed distance to the robot. We did not find this to be a severe limitation, as our collision avoid routines maintain a specific distance while following a person.

As argued in the very beginning, we believe that finding "natural" ways of communication between humans and robots is of utmost importance for the field of service robotics. While this paper exclusively addresses gestures as input modality, we believe it is worthwhile to augment the interface by a speech-based interface, so that both gestures and speech can be combined when instructing a mobile robot. Gestures can help to clarify spoken commands, particularly in such situations in which noise impedes the ability to communicate vocally.

Acknowledgments

We gratefully acknowledge various fruitful discussions with the members of CMU's Robot Learning Lab.

This research is sponsored in part by DARPA via AFMSC (contract number F04701-97-C-0022), TACOM (contract number DAAE07-98-C-L032), and Rome Labs (contract number F30602-98-2-0137), FAPESP (Brazilian Foundation of Research Support) under grant number 96/4515-8, and DaimlerChrysler Research Berlin (via Frieder Lohnert). The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of DARPA, AFMSC, TACOM, Rome Labs, the United States Government, FAPESP, or DaimlerChrysler.

References

H. Asoh, S. Hayamizu, I. Hara, Y. Motomura, S. Akaho, and T. Matsui. Socially embedded learning of office-conversant robot jijo-2. In *Proceedings of IJCAI-97*. IJCAI, Inc., 1997.

H.-J. Boehme, A. Brakensiek, U.-D. Braumann, M. Krabbes, and H.-M. Gross. Neural networks for gesture-based remote control of a mobile robot. In *Proc. 1998 IEEE World Congress on Computational Intelligence WCCI'98 - IJCNN'98*, pages 372–377, Anchorage, 1998. IEEE Computer Society Press.

H.-J. Boehme, U.-D. Braumann, A. Brakensiek, A. Corradini, M. Krabbes, and H.-M. Gross. User localisation for visually-based human-machine interaction. In *Proc. 1998 IEEE Int. Conf. on Face and Gesture Recognition*, pages 486–491, Nara, Japan, 1998.

J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1996.

W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 1999. to appear.

W. Campbell, A. Becker, A. Azarbayejani, A. Bobick, and A. Pentland. Invariant features for 3-d gesture recognition. Technical Report 379, M.I.T. Media Laboratory Perceptual Computing Section, 1996.

I.J. Cox and G.T. Wilfong, editors. *Autonomous Robot Vehicles*. Springer Verlag, Berlin, 1990.

J.L. Crowley. Vision for man-machine interaction. *Robotics and Autonomous Systems*, 19:347–358, 1997.

Y. Cui and J.J. Weng. Hand sign recognition from intensity image sequences with complex backgrouns. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, Killington, Vermont, 1996.

T. Darrel, B. Moghaddam, and A.P. Pentland. Active face tracking and pose estimation in an interactive room. In *Proceedings of the IEEE Sixth International Conference on Computer Vision*, pages 67–72, 1996.

A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

H. Endres, W. Feiten, and G. Lawitzky. Field test of a navigation system: Autonomous cleaning in supermarkets. In *Proc. of the 1998 IEEE International Conference on Robotics & Automation (ICRA 98)*, 1998.

R.J. Firby, R.E. Kahn, P.N. Prokopowicz, and M.J. Swain. An architecture for active vision and action. In *Proceedings of IJCAI-95*, pages 72–79, 1995.

R. J. Firby. Task networks for controlling continous processes. In *Proceedings of the Second International Conference on AI Planning and Application*, 1994.

D. Franklin, R. Kahn, M. Swain, and J. Firby. Happy patrons make better tippers, 1996.

E. Huber and D. Kortenkamp. Using stereo vision to pursue moving agents with a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1995.

R.E. Kahn, M.J. Swain, P.N. Prokopowicz, and R.J. Firby. Gesture recognition using the perseus architecture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 734–741, San Francisco, CA, 1996.

R. Kahn. *Perseus: An Extensible Vision System For Human-Machine Interaction*. PhD thesis, University of Chicago, 1996.

S. King and C. Weiman. Helpmate autonomous mobile robot navigation system. In *Proceedings of the SPIE Conference on Mobile Robots*, pages 190–198, Boston, MA, November 1990. Volume 2352.

D. Kortenkamp, E. Huber, and P. Bonasso. Recognizing and interpreting gestures on a mobile robot. In *Proceedings of AAAI-96*, pages 915–921. AAAI Press/The MIT Press, 1996.

D. Kortenkamp, R.P. Bonassi, and R. Murphy, editors. *AI-based Mobile Robots: Case studies of successful robot systems*, Cambridge, MA, 1998. MIT Press. to appear.

J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proceedings of the International Conference on Computer Vision*, Kerkyra, Korfu, 1999.

P. Maes, B. Blumberg, T. Darrel, and A. Pentland. The ALIVE system: Full body interaction with animated autonomous agents. ACM Multimedia Systems, 5 1997.

D. Margaritis and S. Thrun. Learning to locate an object in 3d space from a sequence of camera images. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1998.

H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

D. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Boston, MA, 1993.

L.R. Rabiner and B.H. Juang. An Introduction to Hidden Markov Models. In *IEEE ASSP Magazine*, 1986.

H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–28, 1998.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Vol. I + II*. MIT Press, 1986.

R.D. Schraft and G. Schmierer. *Serviceroboter*. Springer verlag, 1998. In German.

R. Simmons. The 1994 AAAI robot competition and exhibition. *AI Magazine*, 16(1), Spring 1995.

T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of International Symposium on Computer Vision*. IEEE Computer Society Press, 1995.

S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.

S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

S. Thrun. Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, 1998. to appear.

M. C. Torrance. Natural communication with robots. Master's thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1994.

J. Triesch and C. von der Malsburg. Robotic gesture recognition. In *Proceedings of the Bielefeld Gesture Workshop (GW'97)*, Bielefeld, Germany, 1997. Springer, Lecture Notes in Artificial Intelligence 1371.

J. Triesch and C. von der Malsburg. A gesture interface for human-robot-interaction. In *Proc. 1998 IEEE Int. Conf. on Face and Gesture Recognition*, Nara, Japan, 1998.

A. Waibel and K.-F. Lee, editors. *Readings in Speech Recognition.* Morgan Kaufmann Publishers, San Mateo, CA, 1990.

S. Waldherr, S. Thrun, D. Margaritis, and R. Romero. Template-based recognition of pose and motion gestures on a mobile robot. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

A. Wilson and A. Bobick. Learning visual behavior for gesture analysis. Technical Report 337, M.I.T. Media Laboratory Perceptual Computing Section, 1995.

A. Wilson and A. Bobick. Using configuration states for the representation and recognition of gestures. Technical Report 308, M.I.T. Media Laboratory Perceptual Computing Section, 1995.

A. Wilson, A. Bobick, and J. Cassell. Recovering the temporal structure of natural gesture. Technical Report 388, M.I.T. Media Laboratory Perceptual Computing Section, 1996.

C. Wong, D. Kortenkamp, and M. Speich. A mobile robot that recognizes people. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, 1995.

C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Learning*, 19(7):780–785, 1997.

G. Wyszecki and W. Styles. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* John Wiley & Sons, 1982.

J. Yang and A. Waibel. Tracking human faces in real-time. Technical Report CMU-CS-95-210, School of Computer Science, Carnegie Mellon University, 1995.