Proc. of the Joint Conf. on Robotics: 37th International Symposium on Robotics (ISR 2006) and 4th German Conference on Robotics (Robotik 2006), Munich, Germany, paper nr. 83, 20 pages, VDI Verlag, 2006

PROGRESS IN THE DEVELOPMENT OF AN INTERACTIVE SHOPPING-ASSISTANT[*]

H.-J. Böhme, A. Scheidig, T. Wilhelm, C. Schröter, C. Martin, A. König, S. Müller, and H.-M. Gross
Ilmenau Technical University, Dept. of Neuroinformatics and Cognitive Robotics
P.O.B. 100565, 98684 Ilmenau, Germany

Speaker: Hans-Joachim Böhme, hans-joachim.boehme@tu-ilmenau.de
Topic: Mobile robots
Keywords: multi-modal MCL, human-robot interaction, control architecture, service-robot application

1 Introduction

Our long-term application scenario is aimed at the development of an intelligent interactive shopping-assistant [16], working as a mobile information kiosk in a home store. This scenario requires a variety of highly sophisticated methods for mobile robots. The paper focuses on recent progress concerning three methodological aspects: (i) self-localization of the robot by means of multi-modal approach within a Monte Carlo Localization (MCL) framework, (ii) estimating of identity and gender of users, and (iii) development of a well-suited control architecture for interactive service robots. In our application, exact position knowledge is not only important for secure and efficient coordination of path planning and motion control. Since one of the robot's tasks is to guide customers to the goods they are looking for, it should know its relative position to the goods shelves as accurate as possible to give precise information where to find the desired product. In human-robot interaction, beside robust people detection and tracking, the estimation of identity is of great importance. Recognizing a customer that already used the robot, the shopping-assistant can adopt its dialog regime, because this customer has some experience in interacting with the system. Knowing the gender of the current user allows us to adopt the dialog regime, too, because men and woman often behave completely different within a shopping tour. Furthermore, estimating the emotional state of the human user can be used to adopt the dialog regime too, and first results concerning the estimation of facial expressions are presented. Finally, a well-defined control architecture is needed that should fulfil needs like transparency, extensibility, hardware-abstraction and that should support the efficient implementation of different applications. For our experiments, a B21-robot is utilized.

The remainder of the paper is structured as follows: section 2 describes our proposed method for combining vision and distance measures into a unified MCL framework. In section 3 we give a more detailed insight into the developed approaches for getting more information about the current user of the robot via face analysis. Section 4 introduces a highly sophisticated control architecture where all modules running on the robot are integrated. Each of these sections starts with a short introduction and a state-of-the-art related to the topic at hand, and ends with a corresponding summary. Section 5 summarizes the whole paper, gives an outlook to still open problems and an overview to some work in progress.

2 Multi-modal Monte-Carlo Localization

The ability of self-localization is one of the key requirements for a mobile robot. In order to perform sensible goal-directed autonomous navigation, the robot must have at least a rough estimate of its current position.

Monte-Carlo-Localization is a method for estimating the position of the robot by approximating the probability function for the robot position using a particle filter. The particle distribution is initialized either as a uniform distribution over the entire environment or according to some prior knowledge about the position. With each robot motion, the particles are moved accordingly, introducing some randomness by using a probabilistic motion model. Periodically, observations from external sensors are used to evaluate the position hypotheses represented by the particles. This is done by generating a weight for each particle which is coding the probability of the current observation at the particle's location and resampling the particles from the distribution that is given by the weighted particles.

[*]supported by a grant of the Thuringian Ministry of Science, Research and Art (TMWFK) grant B509-03007 to Horst-Micheal Gross

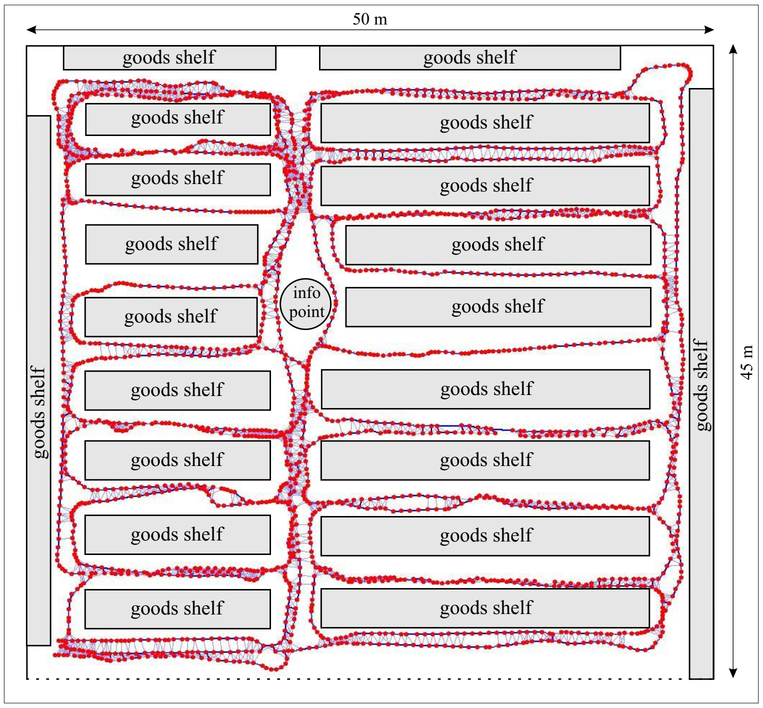


Figure 1. Learned graph of the operation area, a part of the home store. The size of the area is 50 by 45m^2, the graph consists of about 2000 reference points (red dots) labeled with reference observations and odometric data about the pose of the robot at the moment of node insertion. The total distance traveled to learn this graph was about 1.000 m.

MCL and similar approaches also appear in the literature as particle filters, Condensation algorithms, sampling-importance resampling (SIR) filters, etc. A wide range of sensors have been used for localization with MCL, most commonly laser range sensors and vision sensors (cameras). Dellaert et al. presented results for sonar and laser data in [8], [12]. The same authors used a camera observing the hall ceilings in [7]. Gross et al. presented an omnidirectional camera-based approach for large-scale maze-like environments [13], [14]. Further vision-based approaches are presented in [23], [20] and [33]. [26], [2] examined sensor fusion in localization, fusing laser and vision data and using Kalman filters for state estimation.

2.1 Visual MCL

The typical operation area of a home store has a size of about 120m by 100m and is characterized by many similar long hallways (Fig. 1). In contrast to typical indoor environments, very few flat surfaces exist, instead store-shelves with various goods present very finely structured obstacles and walls.

In our application, exact position knowledge is not only important for secure and efficient coordination of path planning and motion control of the robot. Since one of the robot's tasks is to guide customers to the goods they are looking for, it should know its relative position to the goods shelves as accurate as possible to give precise information where to find the desired product.

In [13], [14], we presented a visual localization approach. There, we described a method of splitting an omni-camera image into several slice segments and extracting the red, green and blue mean values of the RGB color space for each segment. Those RGB features are then used as a description of the environment appearance at the respective position and stored in a map. The map is a graph of nodes, covering the two-dimensional environment, where each node contains the features extracted from the omni-camera image at the respective position. The nodes are drawn as points in Fig. 1, connected along the path the robot took during map building. In the localization phase, the same RGB features are determined for the current observation, and for each particle the similarities with nearby nodes are interpolated to calculate the particle weight. A more detailed description of the method and experiments are given in [14].

We argued that because of the specific topology of our operation area with its maze-like structure, visual features should be superior to range-based features, as they produce far less ambiguities. We therefore discarded sonar sensors for MCL and were able to demonstrate that localization is possible using omni-vision only with good results in accuracy and robustness. However, we also experienced that the precision of self-localization strongly depends on the resolution of the underlying graph. In the environment discussed above, this limitation expresses itself particularly in the position component that lies perpendicular to the

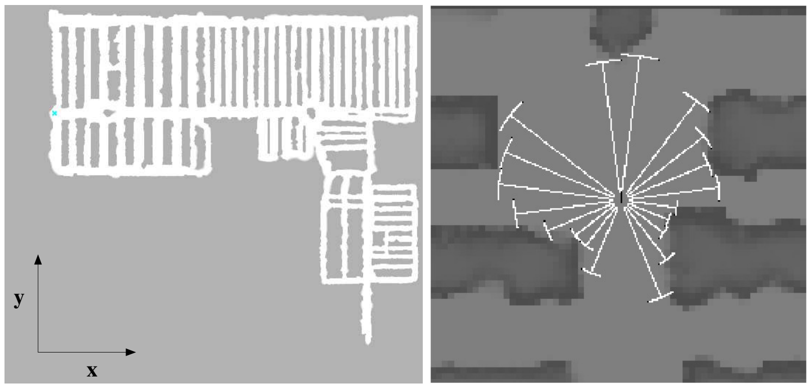


Figure 2. (Left:) A grid map of the entire home store area (light: free space, dark: obstacles). (Right:) Applying a ray casting algorithm, we can calculate a scan expectation for each position in the map (enlarged view).

corridor direction: When the robot moves down a corridor, the changes in the visual observation are far stronger than when just moving closer or further away from a wall. Therefore, to achieve equal localization resolution in both directions, we would need a higher number of reference nodes, particularly many ”lanes” in each corridor (see Fig 1: So far, in most hallways the model only consists of one or two ”lanes”).

2.2 Sonar MCL

In order to improve the localization, we implemented sonar-based MCL as extension of the vision-based approach. This section will describe MCL using sonar sensors as stand-alone solution while in the next sections, we will compare the results of both approaches and address the combination of both modalities.

As explained above, the base of MCL is the incorporation of sensor observations into the position belief by calculating the probability of an observation provided by the robot sensors, assuming the robot is located at a specific position that is given by a particle. In order to estimate that probability, a map of the environment is required. Furthermore, a sensor model must exist which provides the respective probability distribution.

In our application, the need for an environment map also results from the intention to perform autonomous navigation, which in turn requires the ability of path-planning. To that purpose, the entire area is modeled as a grid map (Fig. 2 left). During a teaching phase, the grid map is built from odometry and sonar range data, using a probabilistic update scheme for each single cell [25]. We employ various odometry correction techniques [29] at that stage in order to ensure the accuracy of the map. The basic consideration here is, that we can put high effort into the one-time process of generating environment and robot models, while the models themselves can be used in low complexity algorithms in the actual application later.

In the MCL context, we can use the grid map as a reference for expected range scans: By applying a ray casting algorithm from a certain position within the map, the expected range measurements in any direction are determined, which correspond to the ranges an ideal sensor would measure (Fig. 2 right). Although the resolution is not very high, as the grid cells in our map have a size of 0.2 by 0.2m, we will see that this is sufficient for pose estimation with MCL.

Now, in order to calculate the probability of the real sensor measurements at that position, we need a sensor model (also called observation model) that describes, for a given expected range, the distribution of real range measurements. While the B21 has 24 sonar sensors placed around the enclosure, our model only describes the distribution of one range for one expectation and is applied to each of the sensors, implicitly assuming that the behavior of all sensors is identical.

The sensor model is generated from real sensor data in the following way: During the map building as well as other experiments, we usually store all sensor data in a log file. By analyzing the log file, we can assign to each scan the respective robot position it was taken from. With the known grid map, the expected range for each sensor is calculated using the ray casting algorithm mentioned above. The expected/real scan pairs are stored in a table. To this purpose, the range interval 0 - 5 meters (beyond 5 meters scans are too unreliable, hence not used for localization at all) is discretized in segments of 0.05 meters width. Each table row refers to a reference scan range, each column to a real scan range. For each reference/scan pair, the corresponding cell is increased by 1. That way, each table row represents the absolute frequency of real scan ranges for a specific range expectation (Fig. 3). A probability table is generated afterwards by normalizing the rows, dividing by the absolute row sum. Since the model was generated from a large number of scans all over the application area, it is a good approximation of the sonar sensor behavior in that specific home store environment.

An alternative here would be the approximation of the sensor model by a parametric model, e.g. a mixture of gaussians or a related function. However, that would raise the question what function suits the model best and what are the respective parameters, with the risk of discarding relevant information due to a too simple model. Instead, the table model contains all relevant information and is reasonably small in terms of memory requirement.

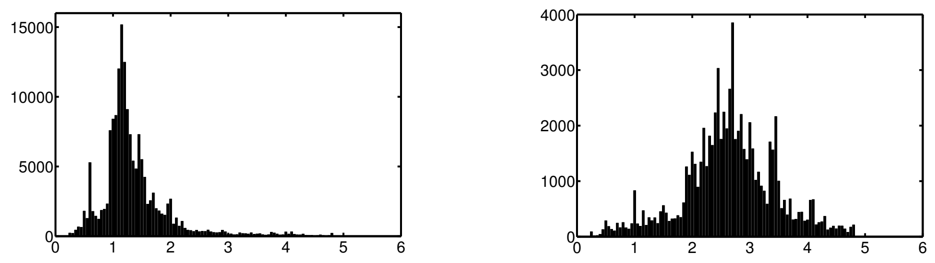


Figure 3. (Left:) Measurement distribution for an expected distance of 1.0m. (Right:) Measurement distribution for an expected distance of 2.5m. It is obviously, the variance of these sonar sensors is higher for long ranges.

We can now use the sensor model to determine a weight for each particle in the observation update step. To this purpose, an expected observation is calculated at the particle's position. Then, for each sonar sensor the respective probability is found in the normalized model table by reading the value in the row corresponding to the expectation and the column corresponding to the actual range measurement. This provides 24 individual probabilities. In order to assign the particle a weight, those 24 probabilities must be merged into one value. One could argue that the individual range readings are mutually independent and, therefore, the overall probability is the product of all values. However, we found that this overemphasizes outliers and can lead to numerical problems with small particle numbers, as the range of particle weights often becomes very large (many orders of magnitude). Instead, averaging over the individual probabilities to calculate the final particle weight leads to much better robustness in the localization. We are aware that this is a deviation from theory, but the improved results legitimate this procedure in our opinion.

Because the calculation of expected reference scans by ray casting in the grid map is quite complex in terms of computation time and could possibly prevent real-time operation with high particle numbers, one more slight modification is needed: Instead of computing the expected observation anew for each particle in each step, the range measurements are pre-calculated for each valid position in the map. Of course this is only possible at a discrete resolution. We found a resolution of 0.1 meters manageable in terms of memory while not significantly affecting the localization results. Storing all expectations for the area presented here results in 200 MB memory allocation. However, for the more relevant task of position tracking (section 2.3) it is suitable to only keep scans in a local vicinity of the current estimation, that way a few scans must be calculated in each observation update, but the majority of particles can still operate on the cache.

To prove the correctness of the sensor model and the weight calculation, an empirical global localization experiment was conducted. Figure 4 shows the progress of localization based on 20,000 samples. Initially, the samples are uniformly distributed over the entire area. As the robot moves and gathers observations, the samples concentrate in certain hallways and eventually converge to the true robot position.

It should be noted that with sonar sensors, global localization needs a denser initial particle distribution than with visual features. This results from the fact that for range sensors, the observation changes faster with small position changes, while in the camera image features are still visible from a far distance, so the actual change in the observation is smaller from one position to the next one. On one hand, this leads to a higher theoretical position resolution with sonar, on the other hand it also means particles must not be too wide-spread to avoid missing the globally best matching position and converging to a sub-optimum.

2.3 Comparison of Visual and Sonar MCL

In this section, we will compare vision-based and sonar-based MCL in terms of localization accuracy. To obtain comparable results, both algorithms were investigated on the same pre-recorded data. To this purpose, we recorded a run with a length of approximately 200 meters. By manually marking the robot position at specific reference points and interpolating between these for odometry correction, the absolute robot position (ground truth) is known very well along the entire path. For the visual localization a map was built from additional data that were also recorded at the same day. The sonar MCL used a grid map that was built from different data recorded several months earlier. Since the coarse structure of the environment remains the same except for some small shelves and movable objects, the changes in the grid map usually are much less significant than the changes in the camera view over time.

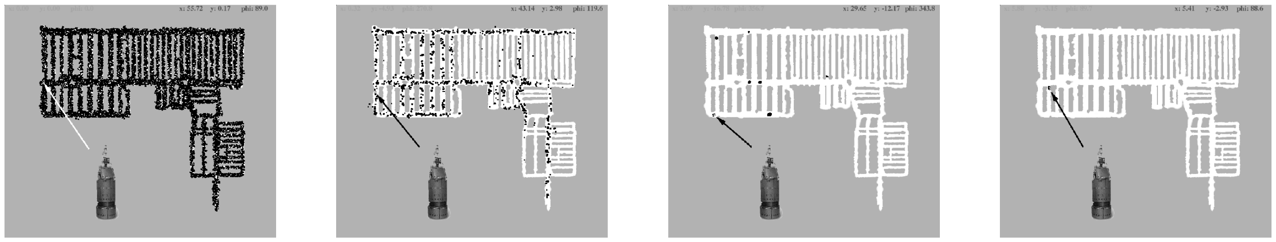


Figure 4. From left to right: 1. Global localization with 20,000 samples: the MCL samples are drawn as black points. Initially, samples are distributed uniformly over the entire map area (top left). 2. After the robot has obtained a few observations, samples only remain in the wider hallways (top right). 3. When the robot turns around the corner, only few positions remain as valid hypotheses (bottom left). 4. After a drive distance of about 30m, the robot has resolved all ambiguities and found the correct position as the only possible estimation (bottom right).

In contrast to the global localization experiment from the previous section, here we only tested position tracking. This is a simpler but in practice more relevant task, as the robot usually won't just be deployed anywhere, but will have at least some approximate knowledge about its startup position, like a "home zone". Therefore, the MCL particles were initialized as a Gaussian distribution around the known starting point. 1,000 particles were used in the the experiments described in this section. The interval of observation updates was the same for sonar-based and visual MCL.

	Sonar			Visual		
	μ	σ	max	μ	σ	max
absolute error	0.30	0.18	1.06	0.39	0.25	1.18
dx error	0.17	0.15	1.01	0.32	0.25	1.15
dy error	0.22	0.18	0.87	0.16	0.14	0.83

Table 1. Results of sonar-based and visual MCL on the same recorded data. The table shows the difference between the estimated position and the reference position. The first column for both "Sonar" and "Visual" contains the average error over all reference points, the second column is the standard deviation and the third column is the maximum error over the entire test path.

Along the test path the estimated position was compared to the known reference positions at intervals of about 0.7 meters, resulting in 290 individual position estimations. Because of the partially random nature of Monte Carlo Localization, we executed 10 runs over the same data to receive significant results. The absolute position error μ for sonar-based MCL and vision-based MCL is shown in table 1. Obviously, both methods perform well and the difference in accuracy is quite small. However, we also examined the position error in x and y direction individually: As explained above, the home store environment mainly consists of many parallel hallways. These hallways are all aligned in a certain direction, which is defined as y direction in our coordinate system (Fig. 2), particularly in the top left part where we conducted the MCL experiments described here. From these explanations follows that a position difference in y direction will mostly mean a longitudinal motion along a hallway, while a position difference in x direction corresponds to a lateral motion between the confining walls or shelves, across the corridor.

By splitting up the absolute position error into an x (lateral) and y (longitudinal) component, we are able to better evaluate the two localization methods: for the sonar-based approach, the error in y direction is slightly higher than the error in x direction. In contrast to that, for the vision-based method the error in x direction is two times higher than the error in y direction. A more detailed illustration of the localization errors is presented in Fig. 5. These results allow interpretation that with visual localization the robot is able to determine its position along the corridor very well, but it has difficulties finding out if it is close to a wall or in the middle of the corridor. With sonars, on the other hand, of course it is far easier to estimate the distance to the lateral walls, but the position along the corridor is less well defined, as the ranges that are measured by the sensors do not change much over the length of a hallway, providing little information about the true position.

2.4 Combination of Visual and Sonar MCL

Now that we know that sonar localization achieves higher accuracies in lateral direction while visual localization leads to higher accuracy in longitudinal direction, an obvious next step is to combine both sensors in order to benefit from the advantages of

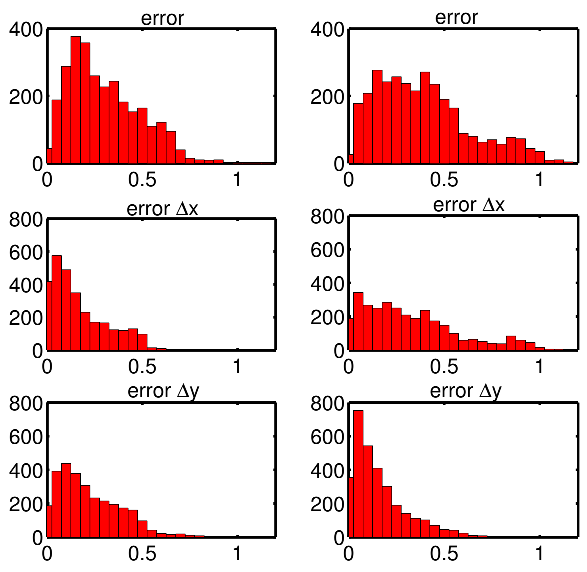


Figure 5. Error histograms for sonar-based MCL (left) and visual MCL (right). The plots show the histograms for absolute position error (top), position error in x direction (middle) and position error in y direction (bottom) over the entire robot test path (all test points on the path). The histograms give a more precise impression of the error distribution than the values in table 1. In each plot, the total number of points is the same. A stronger concentration on the left side means more small errors, therefore a smaller average error. In contrast, plots stretching out to the right contain more high errors, which corresponds to a higher average (expected) position error. Comparing the left and right plots, it is clearly visible that sonar localization is more accurate in x (corridor lateral) direction, while visual localization behaves better in y (longitudinal) direction.

both, resulting in high position accuracy in both directions.

Therefore, we must find a way to integrate the position distributions estimated by the sonar-based and vision-based MCL into one common distribution. Luckily, this is very easy with particle filters: Each of the methods calculates a fit value for each particle in the observation update step, which is then used as weight in the re-sampling process. Instead of using just one fit value, we can now multiply the values from sonar and vision and use the product as re-sampling weight. Since each fit value represents the (relative) probability of the observation at the respective position, and sonar and vision observations are mutually independent, the product of the fit values effectively is the (relative) probability of the combination of both observations. Furthermore, with multiplication (in contrast to e.g. averaging) there is no need to pay attention to the relative magnitude of the individual fit values.

Overall, the changes from MCL with one sensor to multi-sensor MCL are marginal: instead of one map there are now two maps (which share a common reference coordinate system, see Fig 6). The motion model and the motion update step remain unchanged, as well as the particle initialization. In the observation update, the fit values for both observations are calculated and multiplied. The resulting weight is used in the resampling step in the usual way. Actually, it is possible to integrate an arbitrary number of sensors, as long as a map (environment model) exists for each sensor with a common reference frame.

We repeated the experiment from section 2.3 with the Multi-Sensor MCL, again with 1000 particles. The results are shown in table 2 and in Fig. 7. It is clearly visible that not only the absolute error is 50% smaller than with only one sensor, but also in x as well as in y direction the error is smaller than the best single sensor method for each direction. This confirms our expectation that with the combination of the two sensor modalities, the resulting localization method should benefit from the strengths of both and suppress the individual weaknesses.

2.5 Summarizing multi-modal MCL

We have presented two different MCL approaches, one using omni-camera images for environment observations, the other one using sonar range sensors. We tested and compared both methods in a localization task taken from our home store service scenario and found that both perform reasonably well on their own, with each one having its specific strength and weakness. We then developed a simple and very intuitive approach for combination of omni-camera observations and range measurements. The results show that the overall accuracy can be significantly improved and the sensors complement each other very well, each one

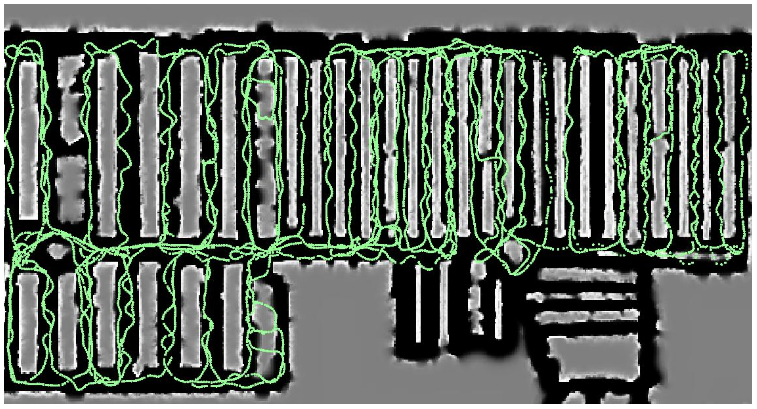


Figure 6. An example of a grid map overlayed with a visual map. Only a part of the entire area is displayed here.

	Vision + Sonar		
	μ	σ	max
absolute error	0.24	0.17	0.83
dx error	0.16	0.14	0.60
dy error	0.15	0.13	0.77

Table 2. Results of the combination of sonar-based and vision-based MCL. The average error in x as well as in y direction is smaller than the minimum of sonar and visual MCL alone (see Table 1).

contributing its own strengths.

We plan to further expand the scope of our approach by closer examination of situations where the sensors strongly diverge in the localization or even one sensor fails, like high occlusion due to crowded environments or radical lighting changes. Our aim is to identify such situations and temporarily ignore the failing sensor. The inclusion of other low-cost sensors is also considered in order to improve robustness and accuracy of the localization.

3 Analyzing Users Face - Estimation of identity, gender, facial expression, and age

A growing number of applications rely on the ability to extract information about people from images. Examples are person identification for surveillance or access control, the estimation of gender and age for building user models or facial expressions recognition, which can give valuable information for the evaluation of man-machine interfaces. As the mentioned recognition tasks have been addressed in isolation in the past, there often exists a variety of methods for each. The presented work was done in the context of building a man-machine interface for a mobile service robot [16], where all the above mentioned information is of great interest. Furthermore, our hope was to identify one method that could be used universally.

A commonly used method for face image analysis is the subspace projection of the image data, where the subspace can be spanned by principal components, independent components of the training data. This method was used for a vast amount of approaches for person identification and automatic facial expression analysis [3]. Another widespread method for person identification and for gender estimation is the Elastic Graph Matching technique [34] [19]. Elastic Graph Matching describes faces in terms of spatial frequencies in local image areas, where the relation between these areas is defined by a graph structure. These models are adaptive and can adjust themselves to some degree to variations in the image data. Active Appearance Models have been used for person identification and facial expression analysis [5] [9]. They describe the statistical variations of shape and grey values in the training data and adapt themselves in an iterative process to a given face image. Up to now, very little work was reported on age estimation from image data.

3.1 Data set coding age, gender, identity, and facial expressions

There exists a variety of face databases designed for single classification tasks, e.g. the Cohn Kanade database for facial expressions [21]. However, we wanted to test the performance of our methods for classification tasks as diverse as gender, age, facial expressions, and identity. To eliminate the influence of varying quality of the image data, we recorded our own database

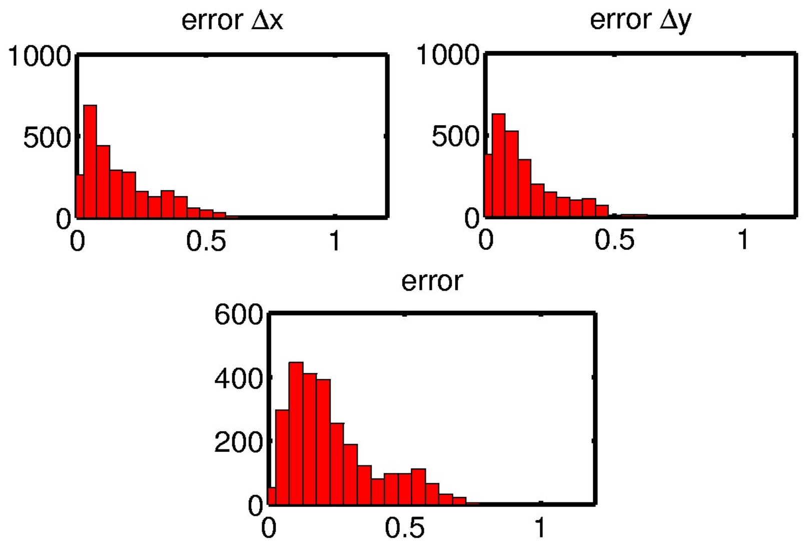


Figure 7. Error histograms for the combination of sonar-based and vision-based MCL. Clearly, the histogram values are concentrated stronger on the left side than in the individual plots in Fig. 5 indicating smaller localization errors.

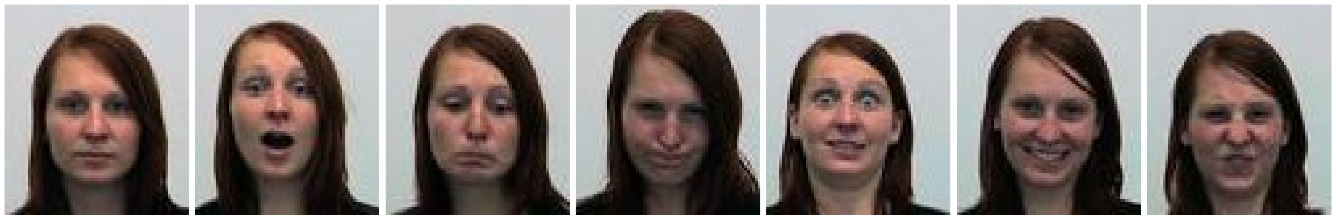


Figure 8. Examples of the used data set for facial expressions. From left to right: neutral, surprise, sadness, anger, fear, happiness, disgust.

according to our requirements. This database consists of two parts. The first part used for the classification of age, gender, and identity contains 70 people with 7 images each, with neutral facial expression, different illuminations, and small deviations in head orientation. Identities are equally distributed in the age range between 10 and 60 and equally distributed over genders. The second part consists of 30 identities with 7 images each, which represent the basic emotions happiness, sadness, surprise, fear, anger, disgust and neutral as identified by Ekman in [10], see Fig. 8. Since evoking facial expressions with movies was shown to produce mixtures of basic emotions [17], we decided to ask probands to pose facial expressions according to the seven basic emotions. As people's ability to pose facial expressions varies significantly, all the recorded images were manually classified by 10 people and only a subset of all images was used where at least 7 people agreed on the facial expression. This problem could be avoided, if the image data was labeled with FACS codes, describing the activity of facial muscles instead of basic emotions. However, since FACS coding is very time consuming and has to be done by trained personnel, it was not yet possible to obtain FACS codes for our data.

3.2 Feature Extraction

To provide the feature extraction methods with normalized data, we manually labeled the position of facial landmarks. We analyzed Independent Component Analysis (ICA) and Active Appearance Models (AAM) for feature extraction.

3.2.1 Independent Component Analysis

For ICA, only the centers of the eyes are used as facial landmarks. The ICA model depends on highly accurate aligned image data as the model does not adapt to a given face image. Thus, we applied affine transformations such that the center of the eyes are on the same position in every image. The size of these normalized images is 60×70 pixels. For building the ICA model, an observation matrix was built by using the vectorized images as rows. On this matrix we applied ICA and obtained independent base images for the data, which represent local facial features, see Fig. 9(a). Any given normalized face image is represented as linear combination of independent base images, and the fit values constitute the data to be classified. For a more detailed description see [32]. The data flow when using the ICA model is depicted in Fig. 9(b).

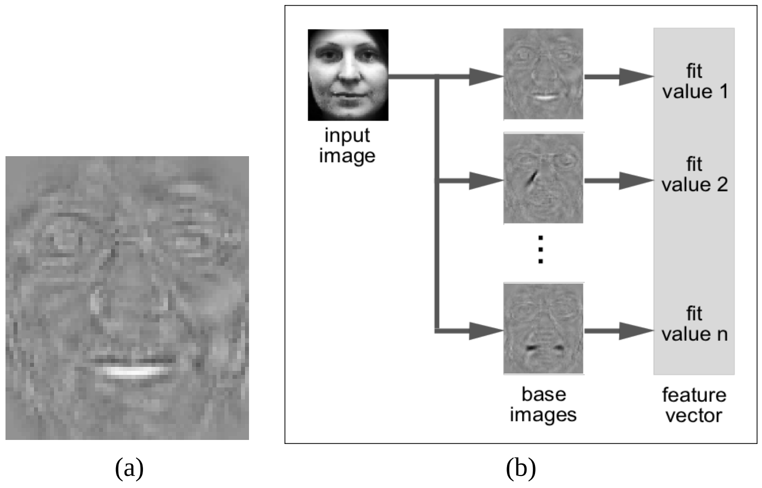


Figure 9. (a) Example of an independent base image. (b) When using the ICA model, a normalized input image is projected on the independent base images. The fit values for the base images form the feature vector to be classified.

3.2.2 Active Appearance Models

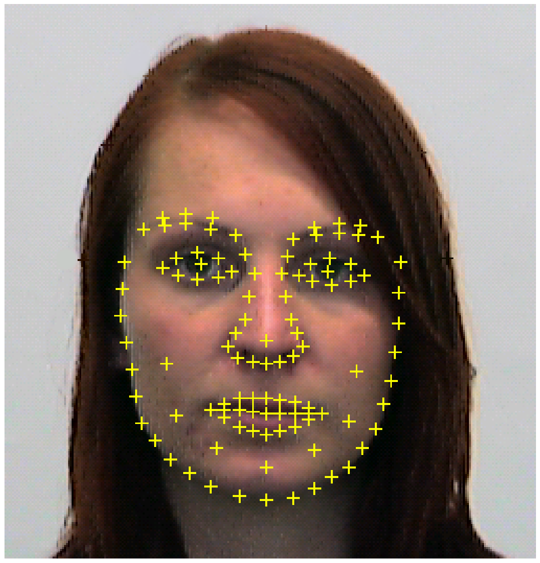
The facial landmarks for the Active Appearance Models (AAM) consist of 116 points along dominant outlines in the face, see Fig. 10(a). To construct an AAM, the mean shape of the training data is computed and the shape variation is determined by principal component analysis. In the next step, the training images are warped to the mean shape. In the same way as with the shape model, the mean grey value face is computed and the grey value variation is determined by principal component analysis. Finally, a predictor matrix is estimated by varying single appearance parameters and averaging their effects on the difference image. For details on Active Appearance Models see [5]. The data flow when using the AAM is depicted in Fig. 10(b). After adaptation of the model to a given face image, the resulting appearance parameters describe the shape and the grey value distribution of the given face and are used as feature vector for classification.

3.3 Classification

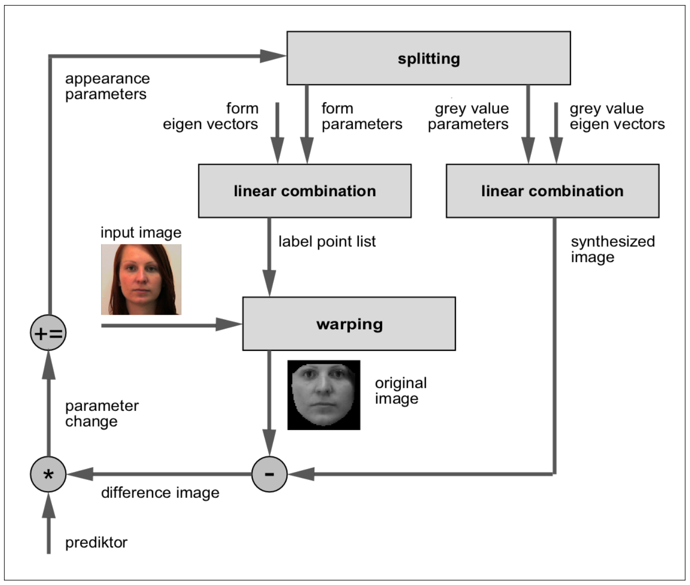
We used a leave-n-out strategy to train and test the classifiers. The partitioning was such, that every identity was in the test data set exactly once. For gender, the test dataset consisted of 2 identities (male and female), for age of one identity per age group and for facial expressions of one person showing all facial expressions. For person identification we used 3 images from each person for training, 2 for validation, and 2 for testing. The results were averaged over the multiple training cycles performed for each recognition task. We compared the following network types: Multi Layer Perceptron (MLP), Radial Basis Function (RBF), Nearest Neighbor (NN), and Generalized Learning Vector Quantization (GLVQ). The number of inputs corresponds to the number of extracted features, that is, ICA fit values or appearance parameters, respectively. The MLPs had two trainable layers with 40 neurons in the hidden layer. The GLVQ network used 20 neurons per class. The centers of RBF networks were initialized by GLVQ training. GLVQ and RBF networks operated on normalized, NN and MLP on un-normalized feature vectors.

3.4 Results and Conclusions

Recognition rates are shown in Fig. 11. For gender and facial expressions, recognition rates are promising. Here, the best results were obtained with AAMs and MLP classifiers or ICA with Nearest Neighbor classifiers, respectively. For both feature extraction methods (ICA and AAM), the results for age classification are only slightly better than guessing (20% due to the used 5 classes). From the confusion matrixes it can be seen, that it is possible to distinguish young and old people, Table 3. For person identification it is often not feasible to have a fixed gallery represented by a neural classifier. Alternatively, two models either ICA fit values or active appearance parameters can be compared by the normalized dot product. Therefore, the false acceptance



(a)



(b)

Figure 10. (a) Landmarks used for the construction of the Active Appearance Model. (b) Usage of the Active Appearance Model. The iterative search process begins with appearance parameter vector $\vec{0}$, that is, with mean shape and mean grey value distribution. The position is initialized by the center of the eyes as with the ICA model. The input image is warped to the current shape to produce the form normalized original image*. On the other hand, the current grey value parameters are used to produce the form normalized* synthesized image*. From the difference of these images, a parameter change is estimated for each appearance parameter with the goal to minimize the energy of the difference image. The search process converges, when this parameter change is $\vec{0}$.*

	10	**20**	**30**	**40**	**50**
10	48	15	18	10	7
20	9	36	22	6	25
30	18	32	20	15	13
40	6	19	29	23	21
50	7	32	21	19	19

(a)

	10	**20**	**30**	**40**	**50**
10	55	11	13	16	3
20	9	40	29	8	12
30	20	19	23	19	17
40	13	7	23	33	22
50	4	10	15	22	47

(b)

Table 3. Confusion matrixes for age estimation. Horizontal: true class, vertical: estimated class (a) ICA + NN (b) AAM + MLP. Each age intervall contained 98 images to be classified. It can be seen, that both methods are roughly able to distinguish young from old people. Thus, when using the system only two or three clusters should be used.

and false rejection rates were recorded for different similarity thresholds, see Fig. 12. Both methods achieve approximately the same equal error rates. Although the best recognition rates for gender and age estimation were obtained with AAMs, in the final system we deploy ICA with Nearest Neighbor classifiers only. This is a trade-off between recognition rates and processing time needed to adapt the AAM to the input image, which is about 2800ms compared to about 20ms for the subspace projection with ICA. Since the AAM proved its capability for various recognition tasks, our future work will focus on optimizing the AMM for processing speed. In contrast to the ICA subspace projection, which relies on accurately aligned frontal views, the AAM is also able to handle and estimate head pose, provided this image variation is present in the training data.

4 Control Architecture

To develop an interactive mobile robot, each researcher has to deal with a lot of different classic robotic problems, like a precise localization, a collision-free navigation and a robust Human-Robot Interaction (HRI) interface. Furthermore, another very important component of an interactive mobile robot is the application itself. Without an useful application the robot itself doesn't make sense.

That means, that each developer has to design a specific application to give this robot an useful task. We will describe a new

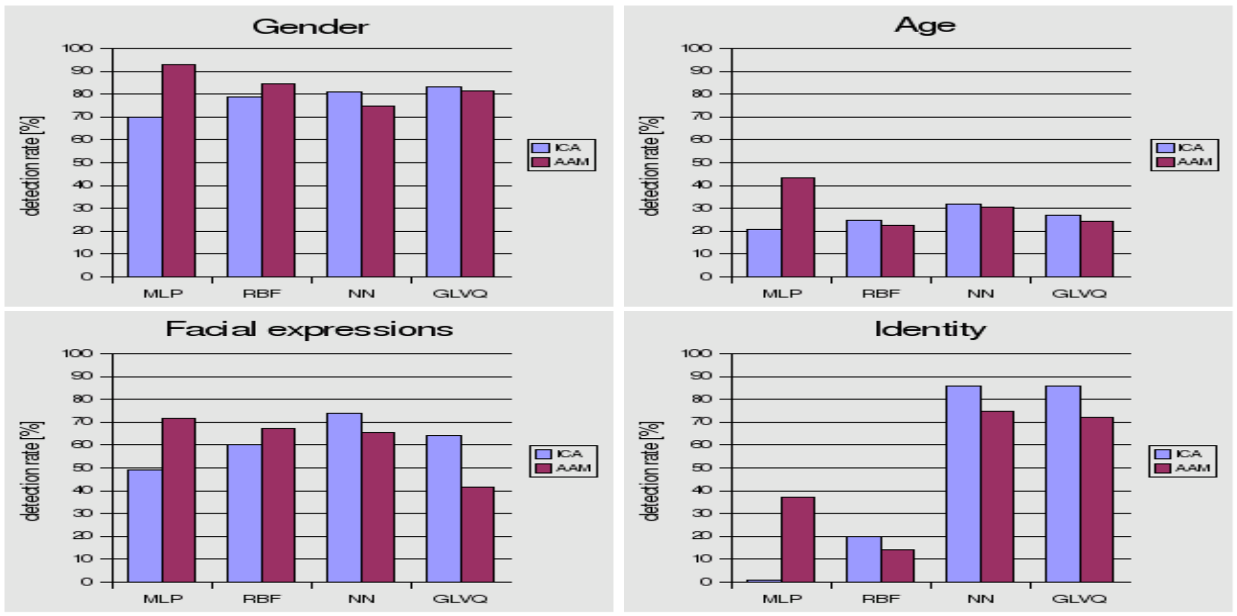


Figure 11. Recognition rates for gender, age, facial expressions and identity on validation data for feature extraction with ICA or AAM, respectively. The best recognition rates w.r.t. gender, age and facial expression were obtained with AAM and MLP or ICA and NN, respectively. The high recognition rates for AAM with MLP classifiers suggest, that the AAM produces appearance parameters which are in contrast to the ICA fit values well clustered according to the recognition tasks. For person identification the ICA feature extraction performs significantly better than the AAM. Here MLP and RBF networks fail, because of the large number of clusters. However, in the final system, person identification is done by comparing two models and using a similarity threshold for acceptance or rejection, see Fig. 12.

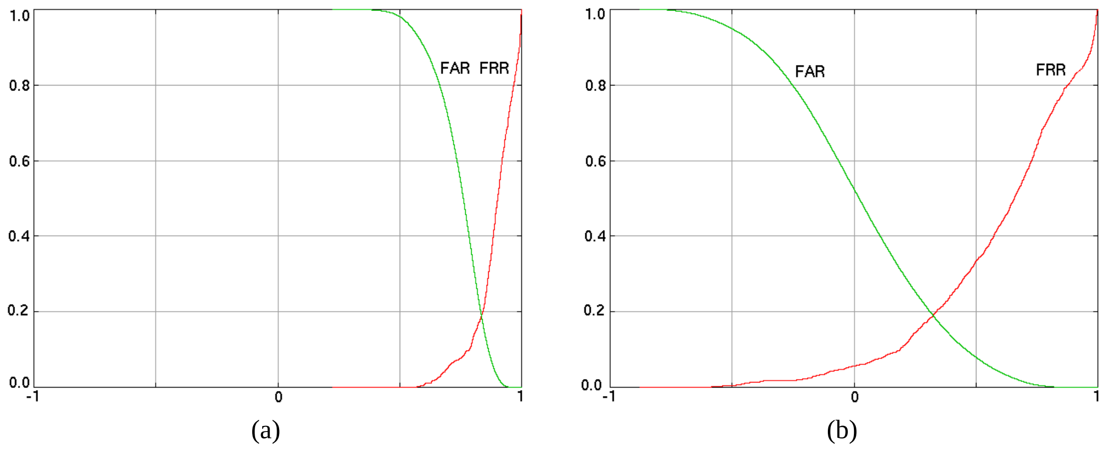


Figure 12. False acceptance (FAR) and false rejection rate (FRR) curves for person identification with (a) ICA (b) AAM. The equal error rates are approximately the same with 0.2 *for ICA and* 0.19 *for AAM, which suggests that both methods are equally well suited for person identification.*

robot control architecture, which on the one side gives programming users a very flexible system, which is modular, extensible, transparent and portable for different robot platforms, and on the other side allows a fast development of robotic applications, and which is easy to understand and use for newcomers or non-programming users.

We have implemented and tested our proposed architecture on our robots HOROS, PERSES and SCITOS (see fig. 19).

A standardized control architecture for mobile robots is still an open question today. But there are a wide range of academical and also commercial (like ERSP [11]) work. The most modern control architectures are heterogeneous and are called *hybrid*. They contain *reactive* as well as *deliberative* components.

Over the last years, many different robot system architectures were developed and introduced. A good overview is given in [6]. On the one hand, there are some architectures consisting of up to three layers (like 3T, CLARAty [27] or OROCOS), which are basically designed to deal with navigation problems and on the other hand there are some low-level architectures (like Player/Stage [31]), which only describe a kind of a robotic programming interface. Furthermore, there are existing systems in-between (like CARMEN [24]), which consist of a simple architecture and a full programming interface.

The major problem of all theses systems is, that they are basically designed to connect and control the different modules and methods (the skills) of a robot, but they are typically not designed to build a concrete robotic application. In most shown examples, mainly a few modules for collision avoidance, path planning, self localization, and other navigation skills are implemented. As far as we know, none of theses systems was tested or used with methods for Human-Robot Interaction. Furthermore, none of theses systems provides a possibility to integrate modules, which are required to build a real interactive mobile robot, like elements for a graphical user interface or a dialog management system. The only way to create an application for an interactive mobile robot based on the systems mentioned above seems to be to create a step-by-step program, which controls the whole system. Typically this can exclusively be done by a developer or programmer, which knows the whole system and desired application in detail.

This way, in the last years, several interactive mobile robots with specific architectures were built and introduced. One of theses robots is GRACE (Graduate Robot Attending a ConferencE) [30], which was build to tackle the AAAI Robot Challenge in 2003. The robot's software consists of many programs communicating via IPC (Inter Process Communication) and CARMEN [24]. The different programs exchange messages, e.g. sensor information, commands, or events. Each program provides one skill of the robot. This distribution guarantees the re-useability and the modularity of this system. But if someone wants to realize a new application with such an architecture, he has to know the whole system and all programs in detail.

The humanoid robot HERMES [4] is controlled by a behavior-based system [1], which uses a situation module (situation assessment and behavior selection) as core of the whole system. Around this core a set of skills is used to control the robot. The skills and the situation module are communicating via events and messages. The architecture for HERMES was designed to be reusable, but as far as we know, it was never used for another robot.

The robot MOBSY [35] uses two levels for the software integration. The most abstract level is the task level, where the overall behavior of the system is determined. The second level contains the robot-specific methods, which provide the different skills of the robot. The skills will be used in the task level to execute the behaviors. The architecture of MOBSY seems to be specially designed for the appropriate task and is not reusable for other applications.

The Care-O-bot [18] uses a hybrid architecture. The four most important components of the robot are the Man-Machine-Interaction module, a symbolic planner, a fact manager, and an execution module. Furthermore, the robot uses a database which contains all necessary information about the environment. Although this architecture contains explicit a Man-Machine-Interaction module and is used for different (but similar) robots, it seems not to be easy to create new applications, because some important parts (like to symbolic planner and the fact manager) have to be modified for each new task.

In the following section, we will define our requirements to a control architecture for mobile interaction-robots and introduce our system in detail.

4.1 Requirements

A modern robot control architecture has to fulfill the following common demands:

- *Modularity*: The different modules of the architecture must be functional independent and exchangeable.
- *Extensibility*: The architecture must be easily extensible with new modules.
- *Transparency*: An exchange or a modification of a single module must be transparent to the other modules.
- *Portability*: The architecture should be able to run on different robot platforms.
- *Efficiency*: The architecture must be able to run in real-time on the underlying robot.

Based on the introduced problems with existing control architectures, we derived the following additional demands on a control architecture for a mobile interaction robot:

- *Rapid Application Development*: quickly generation of a new applications
- *Customizability*: easy generation of new applications also by non-programming users and
- *Re-usability*: easy reuse of a generated application by different robot systems with different hardware components.

4.2 Basic Structure of the Architecture

To build a real robotic application which can fulfill all these demands, it is necessary to separate the robot-specific methods and skills (e.g. collision avoidance or people detection) from the application itself (e.g. a robot as an office guide or a robot as a shopping assistant). Doing this way, the demands *Customizability* and *Re-usability* can be easily fulfilled. To bring these different parts together, an abstraction layer in-between is necessary. As the result, our developed architecture consists of four layers (see figure 13 and 14).

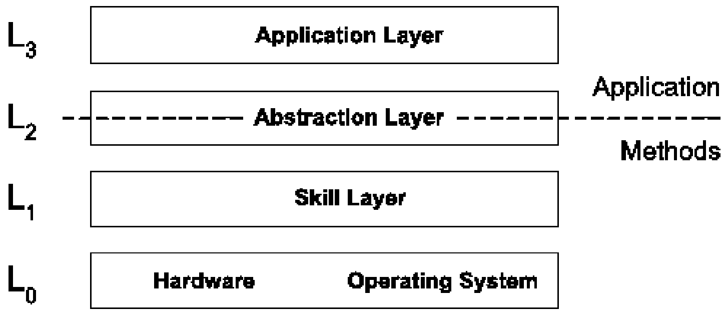
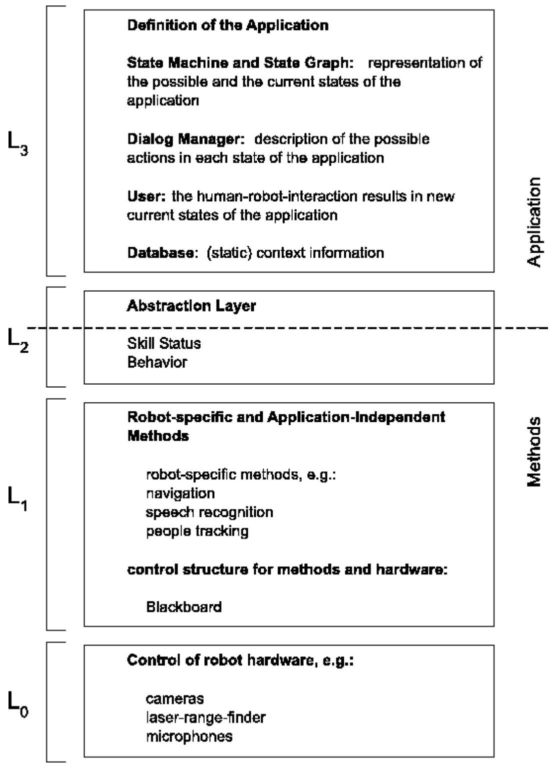


Figure 13. **The four layers of the architecture.** *The* Abstraction Layer L_2 *separates the methods from the application.*

The layer L_0 (*Hardware Layer*) encloses the robot hardware (sensors and actuators), the operating system, and the low-level interface to the hardware. The low-level sensor information will be processed in the next higher level to provide different skills, which will be executed in L_0.

In the next layer L_1 (*Skill Layer*) all required classical robotic-specific methods are located. Typically, these are modules for collision avoidance, localization and navigation, speech recognition, speech synthesis, a people- or object tracking and so on. These different robot-specific methods and skills are reusable for numerous different applications. The functionality and complexity of layer L_1 depends on the underlying robot.



L_3
Definition of the Application
State Machine and State Graph: representation of the possible and the current states of the application
Dialog Manager: description of the possible actions in each state of the application
User: the human-robot-interaction results in new current states of the application
Database: (static) context information
Application
L_2
Abstraction Layer
Skill Status
Behavior
L_1
Robot-specific and Application-Independent Methods
robot-specific methods, e.g.:
navigation
speech recognition
people tracking
control structure for methods and hardware:
Blackboard
Methods
L_0
Control of robot hardware, e.g.:
cameras
laser-range-finder
microphones

Figure 14. **The main components of the architecture:** *the method-level with robot-specific methods, the application-level with a robot-independent application and a robot-specific interface to adapt the current application to the currently used robot.*

The layer L_2 (*Abstraction Layer*) generalizes from the robot-specific skills of L_1 and provides a high-level interface to the capabilities of the robot. The skills (like navigation and people tracking) are combined in this layer to a set of high-level *Behaviors* (like people guidance). To control the skills of L_1 and to get a feedback about the execution of the different skills the so called *Skill Status* is used.

The highest layer L_3 (*Application Layer*) provides elements, which are required for a specific application of a mobile interactive robot.

An important fact is, that all layers are only communicating with their immediate neighbors. This guarantees the transparency between the different layers.

By using this strict separation, for a new application or for the usage of an alternative robot system only the application layer L_3 or the abstraction layer L_2 have to be changed rather than the whole system. Thus, the introduced demands *Rapid-Application-Development*, *Portability* and *Re-usability* can be fulfilled. Furthermore, to allow the generation of an application also by a non-programming user (*Customizability*), we use parameterizable elements in the application layer (the state graph and the dialog manager). In the following section, we discuss the specific components of the control architecture in detail.

4.3 Specific Elements of the Architecture

As shown in figures 13-15 our control architecture incorporates a *Skill Layer* L_1 and an *Application Layer* L_3, which are connected by an *Abstraction Layer* L_2.

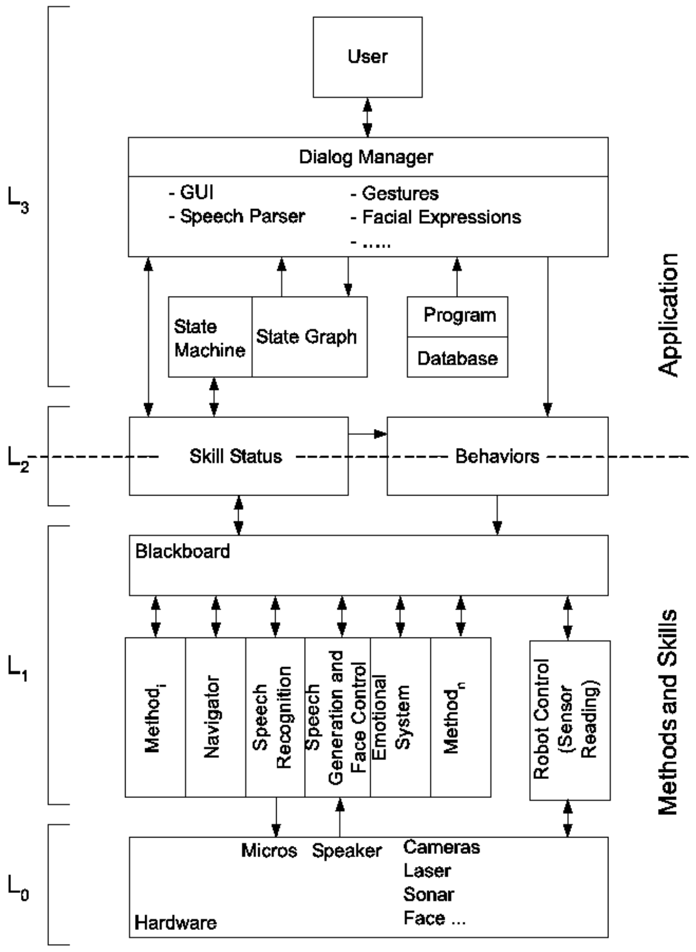


Figure 15. **The four layers in detail.** *Specific elements of the four layers of the control architecture.*

The *Skill Layer* consists of two main components: The robot-specific and application-independent methods and a *Blackboard System*. In the context of a mobile interaction-robot the methods should include at least a navigation strategy, a people detection module, a speech recognition and a speech generation (see figure 15). The realization of such methods mainly depends on the hardware components (the *Hardware Layer*) of the underlying robot system. For instance, a robot which is equipped with sonar sensors will use a different navigation method than a robot equipped with a laser-range-finder.

Furthermore, for the communication between all different methods we use a *Blackboard System* [28]. With the blackboard it is possible to share all required information between the different methods. Therefore, the blackboard can be considered as general shared data memory. The blackboard structure makes it easy for the programming users to integrate new modules or to make modifications on existing modules. The programmer only must ensure, that the interface from his module to the blackboard keeps consistent. Therefore, with a blackboard system the demands *Modularity*, *Extensibility* and *Transparency* can be easily fulfilled.

The available robot-specific methods of L_1 will be combined in the *Abstraction Layer* to a set of high-level macro-behaviors. Each of these behaviors uses a set of available skills of L_1. Typically, the available behaviors are exclusive, i.e., only one behavior can be executed at a time. As a result, the layer L_2 provides a set of high-level macro-behaviors, which can be executed without knowledge about the underlying skills of L_1. Thus, it is very easy for non-programming users to use the robot. For instance, in a state ”Create Attention” the desired behavior could be defined as ”Face the current user”. Depending on the used robot a specific system could activate the (robot-specific) person tracker and track a person by moving the pan-and-tilt camera unit. Another system without a pan-and-tilt camera unit could also track the perceived person by moving the whole robot body. So the *Behaviors* and *Skill Status* translate the application-specific behaviors in the robot-specific methods. The *Skill Status* is also used to get a feedback of the execution of the underlying robot-specific method. For instance, if a navigation task can not be fulfilled (because the way is blocked), the state machine or the dialog manager can react in an application-depended way and solve the conflict.

Depending on the general application which has to be realized, the available behaviors of L_2 will be combined in the *Application Layer*. This means, creating a new application means to combine the available macro-behaviors in a new kind. In praxis, this process of combination is mostly large-scaled and requires a lot of new programming which can be typically done only by people knowing the underlying robot-specific methods and their realizations. In order to simplify this, we propose the separate application-level depicted in figure 15. The elements of the *Application Layer* include a state graph (interpreted by a state machine), a dialog manager, the user (as the interaction partner of the robot), and a database, which contains the information about the environment. Thereby, the state graph incorporates the principle states of the application. For instance, the application ”information system” could be defined by the states ”Wait” (when no user is perceivable), ”Create Attention” (when a user is perceived) and ”Dialog” (when the user wants to interact) (see section 4.5.2 for details). Further, each state also defines a principle behavior which will be executed if the system is in this specific state. Each state is connected with at least one other state. These connections define specific conditions that have to be fulfilled to get into a state. Thereby, the conditions result from the robot-specific methods (via the *Skill Status*) or from the user (via the dialog manager). The resulting whole definition of a state graph is based on a XML-file. So an application is more configurable than programmable, and in the consequence pretty easy to generate also by non-programming people. By defining only this state graph in the application-level, already a whole application can be generated. Therefore, the mentioned demands *Rapid-Application-Development*, *Customizability* and also *Re-usability* are fulfilled.

4.4 State Machine and Dialog Manager

In our proposed architecture, the robot will be controlled by the state machine and the dialog manager. The state machine makes a suggestion, which default behavior should be executed in L_2 in the current state. The dialog manager can simply send this behavior to the Abstraction Layer or select another behavior. By doing so, the desired behaviors in each state can be be chosen more flexibly. For instance, in the state ”Create Attention” the state graph defines a principle behavior which will be executed if no other behaviors are defined in the dialog manager. Thus, the dialog manager could also define, e.g., numerous different speech outputs to attract the user’s attention in this state. Furthermore, the dialog manager can also realize, that in the context of the current application the most promising behavior among all possible behaviors in a specific state can be learned. In the next version of the control architecture, the dialog manager will also be configurable like the state graph.

4.5 Applications

We use the proposed control architecture for all of our mobile interactive robots. Subsequently, we discuss one part of the task of HOROS, to be an office information and guiding system for employees, students, and guests of our institute, in the context of the proposed control architecture.

4.5.1 Robot System HOROS

The hardware platform for HOROS is a Pioneer-II-based robot from ActiveMedia. It integrates an on-board PC (Pentium M, 1.6 GHz, 512MB) and is equipped with a laser-range-finder and sonar sensors. For the purpose of HRI, this platform was extended with different modalities. This includes a tablet PC (Pentium M, 1.1 GHz, 256MB) for touch-based interaction, speech recognition and speech generation. It was further extended by a robot face which includes an omnidirectional fisheye camera, two microphones and two frontal webcams for the visual analysis of dialog-relevant user features (e.g. age, gender, emotions).

4.5.2 The Control Architecture in the Context of a Survey Task

The office application of HOROS includes a survey task, which will be discussed in the context of the control architecture. Thereby, HOROS is standing in a hallway in our department. His task is to attract attention of people that came by. As soon as the system recognized a person near him, the robot addresses the visitor to come nearer. He then offers to participate in a survey about the desired future functionality of HOROS. Further, a people tracking module is used to detect break offs, thus if the user is leaving before finishing to survey, the robot tries to make them came back and finalize the survey. After the successful completion of the interaction or a defined time interval with no person coming back, the cycle begins again with HOROS waiting for the next interaction partner. The experiment was made in the absence of any visible staff members, so the people could interact more unbiased. The respective state graph as an element of the Application Layer L_3 is shown in figure 16.

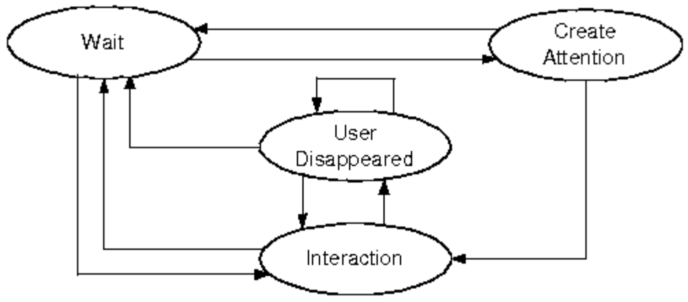


Figure 16. ***State graph for the survey task.*** *The state graph consists of only four states which already define the principle application.*

Each state of the state graph has also some defined input conditions. So if the specific input conditions of a state are fulfilled the application will get into this state. In each state, all outgoing conditions must be consistent. Exemplary, the incoming and outgoing conditions for the state ”Create Attention” are depicted in figure 17. This state can be reached only from the state ”Wait” (see figure 16) by the following incoming conditions: the application was longer than 30 seconds in the state ”Wait” (the last interaction partner of the robot left the surroundings of the robot) and at least one new user is perceived. In figure 17, two outgoing conditions are also depicted. These are as well the incoming conditions for the state ”Wait” (the perceived user left the surroundings without an interaction or the perceived user came not closer to the robot within 15 seconds) and for the state ”Interaction” (at least one perceived user came closer than 0.75 meters to the robot).

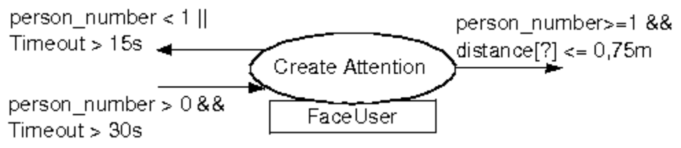


Figure 17. ***State ”Create Attention”.*** *There is one incoming condition and two outgoing conditions in the state ”Create Attention”. Further, when this state will be reached the general behavior ”Face User” will also be activated.*

The state ”Create Attention” is also defined by a general behavior ”Face User”, that will be executed if the application reaches this state via fulfilled input conditions. This behavior will be routed to the dialog manager next. If there are no other defined behaviors for this state in the dialog manager, ”Face User” will be sent back to the Abstraction Layer L_2, then to the Skill Layer L_1 and subsequently will be executed by the Hardware L_0.

Simultaneously to these processes, the robot permanently perceives its environment in the Hardware Layer L_0. Using the respective sensor readings in the Skill Layer L_1 the used methods and Blackboard variables are updated. Consequently these Blackboard variables can also result in an updated Skill Status in the Skill Layer L_2 and subsequently in a newly activated state of the state graph. State transitions in the state graph can also be caused by user inputs, e.g. via the GUI in the dialog manager.

Another application of HOROS based on our control architecture is a guidance function also in the context of our office scenario as described in [22]. The robot can guide visitors from the entrance of the building to the rooms of staff members and give information about their possible whereabouts.

4.5.3 The Control Architecture in Context of a Shopping-Assistant

PERSES is the robot that works as an interactive mobile shopping-assistant [15]. Recently, the implementation of the *Abstraction Layer* for PERSES was finished. As soon as this work was finished, it was quite easy to create the shopping-assistant application in our architecture, although PERSES (a B21r robot) is totally different to HOROS (a Pioneer-II-based robot). To describe and explain the overall interaction scheme within our architecture would go beyond the scope of this paper. Therefore, only a very abstract state graph ist discussed.

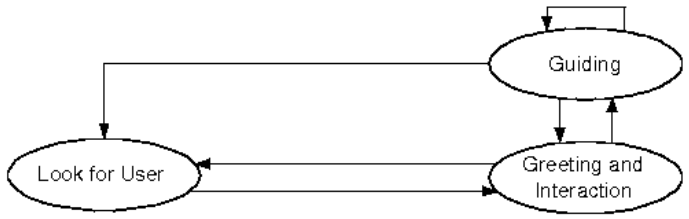


Figure 18. Basic state graph for the shopping assistant.

Figure 18 shows the possible basic structure of a state graph, which is necessary for PERSES. Of course, this graph has to be completed with different sub-graphs and conditions in the individual states. Using such a state graph, a simple dialog manager and an appropriate database, PERSES will be able to work as a shopping-assistant.

4.6 Summarizing the proposed Control Architecture

In this section, we described a control architecture for mobile interactive robots. Our architecture fulfills the demands on modern architectures, like modularity, extensibility, portability and efficiency. Based on our architecture, it is very easy for programming users to integrate new modules or to modify existing parts. Furthermore, our architecture also is designed to allow non-programming users to develop robot applications. It guarantees the rapid application development and the reusability of existing applications. We illustrated the usability of this concept in two typically applications (survey task and shopping assistant).

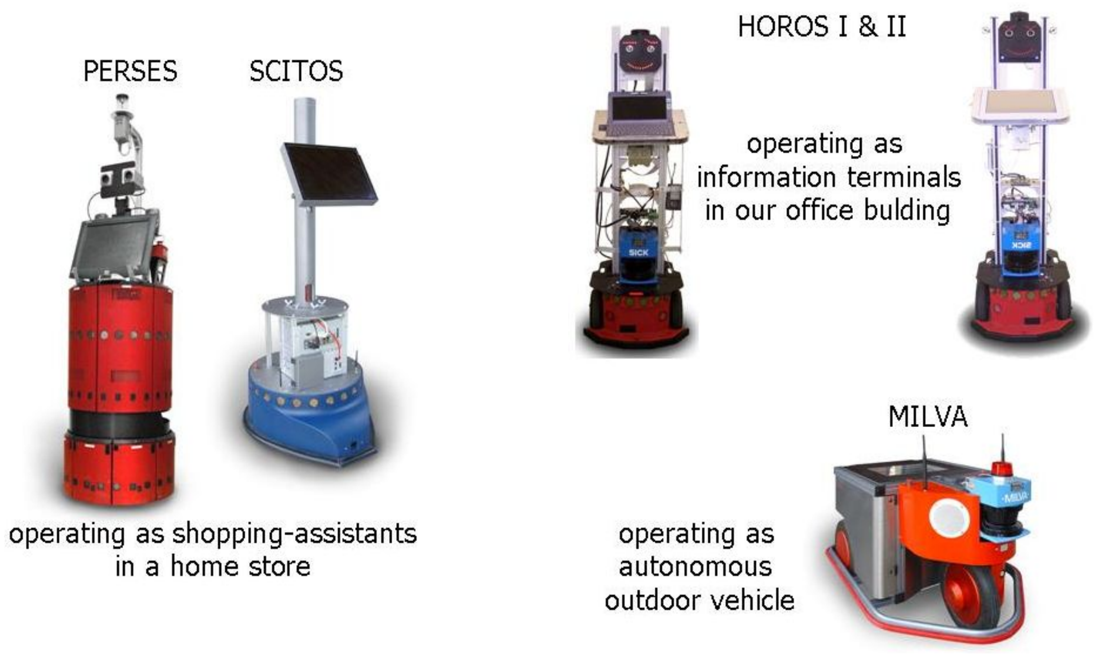

PERSES
SCITOS
operating as shopping-assistants in a home store
HOROS I & II
operating as information terminals in our office bulding
MILVA
operating as autonomous outdoor vehicle

Figure 19. Collection of robots currently used at our department. All robots are equipped with the same control architecture described so far.

Obviously, our proposed architecture is undergoing continuous changes. For example, it is to be extended with a configurable and adaptive dialog manager, which will it make still easier to develop new robotic applications.

5 Summary and Conclusions

The paper summarizes recent progress concerning three methodological aspects within the development of a mobile interactive shopping-assistant: self-localization of the robot by means of multi-modal approach within a Monte Carlo Localization (MCL) framework, estimating of identity and gender of users, and the design and implementation of a well-suited control architecture for interactive service robots.

The proposed methods for robot navigation have already been tested intensively, leading to a very robust operation of the robot in this large-scale and crowded environment. But to reach the state, where the robot can be left in the market without any external observation, we have to integrate vision-based obstacle detection techniques. These algorithms are necessary to complement laser and sonar sensors. Using the latter alone does not lead to a sophisticating obstacle avoidance because of highly varying obstacle configurations.

Concerning human-robot interaction, all methods are almost finally implemented, but extensive testing in the real operation area has to be done to estimate the robustness of the different algorithms under real-world circumstances. Additionally, currently we concentrate our efforts on speeding-up the implementation of the AAMs, because of their outstanding results for face analysis.

Having a well-designed and properly implemented control architecture is of great importance for establishing different applications of service robots. In our experience, besides the fact that implementation becomes much easier, the major advantage comes with the transfer of modules between different robots.

For our future work related to the interactive shopping-assistant, we will use a newly developed and highly sophisticated robot platform SCITOS. This robot is the result of a strong collaboration between our department, the company MetraLabs GmbH (www.metralabs.com), and the company TechnoTeam Bildverarbeitung GmbH (www.technoteam.de).

References

[1] Arkin, R.C. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[2] K.O. Arras, N. Tomatis, B.T. Jensen, and R. Siegwart. Multisensor on-the-fly localization: Precision and reliability for applications. *Robotics & Autonomous Systems*, 34(23):131 – 143, 2001.

[3] M.S. Bartlett. *Face image analysis by unsupervised learning*. Kluwer Academic Publishers, 2001.

[4] Bischoff, R. Towards the Development of “Plug-and-Play” Personal Robots. In *IEEE-RAS International Conference on Humanoid Robots*, 2000.

[5] Cootes, T.F., Edwards, G.J., and Taylor, C.J. *Active appearance models*. Springer.

[6] Coste-Maniere, E. and Simmons, R. Architecture, the Backbone of Robotic Systems. In *Proc. IEEE Internat. Conference on Robotics and Automation*, 2000.

[7] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pages 2588–2596, June 1999.

[8] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[9] Edwards, G.J., Taylor, C.J., and Cootes, T.F. Learning to identify and track faces in image sequences. In *Proceedings International Conference on Computer Vision*, page 317322, 1998.

[10] Ekman, P. and Friesen, W.V. *Unmasking the face. A guide to recognizing emotions from facial clues*. Prentice-Hall, Englewood Cliffs, New Jersey.

[11] Evolution Robotics GmbH. Homepage: http://www.evolution.com.

[12] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the AAAI National Conference on Artifical Intelligence*, Orlando, Florida, 1999.

[13] H.-M. Gross, A. Koenig, H.-J. Boehme, and C. Schroeter. Vision-based monte carlo self-localization for a mobile service robot acting as shopping assistant in a home store. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and System*, pages 265–262, Lausanne, 2002.

[14] H.-M. Gross, A. Koenig, C. Schroeter, and H.-J. Boehme. Omnivision-based probalistic self-localization for a mobile shopping assistant continued. In *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and System*, pages 1505–1511, Las Vegas, 2003.

[15] Gross, H.-M. and Boehme, H.-J. PERSES - a Vision-based Interactive Mobile Shopping Assistant. In *Proc. IEEE Internat. Conf. on Systems, Man and Cybernetics (SMC 2000)*, pages pp. 80–85, 2000.

[16] Gross, H.-M., Boehme, H.-J., Key, J., and Wilhelm, T. The PERSES Project - a Vision-based Interactive Mobile Shopping Assistant. *Künstliche Intelligenz*, 2000(4):34–36, 2000.

[17] Gross, J.J. and Levenson, R.W. Emotion elicitation using films. *Cognition and Emotion*, (9):87–108, 1995.

[18] Hans, M. and Baum, W. Concept of a Hybrid Architecture for Care-O-Bot. In *Proceedings of ROMAN-2001*, pages pp. 407–411, 2001.

[19] Hong, H., Neven, H., and v. d. Malsburg, C. Online facial expression recognition based on personalized gallery. In *Proceedings IEEE International Conference on Automatic Face and Gesture Recognition*, page 354359, 1998.

[20] M. Jogan and A. Leonardis. Robust localization using panoramic view-based recognition. In *Proc. 15th Int. Conf. Pattern Recogn. (ICPR'00)*, pages 136–139, 2000.

[21] Kanade, T., Cohn, J.F., and Tian, Y. Comprehensive database for facial expression analysis. In *Proceedings IEEE International Conference on Automatic Face and Gesture Recognition*, page 4653, 2000.

[22] Martin, C., Boehme, H.-J., and Gross, H.-M. Conception and Realization of a Multi-Sensory Interactive Mobile Office Guide. In *Proc. IEEE Internat. Conf. on Systems, Man and Cybernetics (SMC)*, pages 5368–5373, 2004.

[23] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based monte carlo localization with omnidirectional images. *Robotics and Autonomous Systems*, 48:17–30, 2004.

[24] Montemerlo, M., Roy, N., and Thrun, S. Perspectives on Standardization in Mobile Robot Programming: The Carnegie Mellon Navigation (CARMEN) Toolkit. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 3, pages 2436–2441, 2003.

[25] H. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–77, 1988.

[26] J. Neira, J.D. Tardos, J. Horn, and G. Schmidt. Fusing range and intensity images for mobile robot localization. *IEEE Transactions on Robotics and Automation*, 15(1):76–84, 1999.

[27] Nesnas, I.A., Wright, A., Bajracharya, M., Simmons, R., Estlin, T., and Kim, W. CLARAty: An Architecture for Reusable Robotic Software. In *SPIE Aerosense Conference*, 2003.

[28] Nii, H. P. Blackboard Systems. *The Handbook of Articifical Intelligence 4*, pages 1–82, 1989.

[29] C. Schroeter, H.-J. Boehme, and H.-M. Gross. Robust map building for an autonomous robot using low-cost sensors. In *Proceedings of the 2004 IEEE Conference on Systems, Man & Cybernetics (SMC2004)*, pages 5398 – 5403, The Hague, Netherlands, October 2004.

[30] Simmons, R., Goldberg, D., Goode, A., Montemerlo, M., Roy., N., Sellner, B., Urmson, C., Schultz, A., Abramson, M., Adams, W., Atrash, A., Bugajska, M., Coblenz, M., MacMahon, M., Perzanowski, D., Horswill, I., Zubek, R., Kortenkamp, D., Wolfe, B., Milam, T., and Maxwell, B. Grace: An autonomouse robot for AAAI robot challenge. In *AAAI Magazine, vol. 24, no. 2*, pages 51–72, 2003.

[31] Vaughan, R., Gerkey, B., and Howard, A. On device abstractions for portable, resuable robot code. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003.

[32] Wilhelm, T. and Backhaus, A. Statistical and Neural Methods for Vision-based Analysis of Facial Expressions and Gender. In *Proceedings IEEE International Conference on System Man and Cybernetics*, page 22032208, 2004.

[33] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor. Omnidirectional vision for robot navigation. In *IEEE Workshop on Omnidirectional Vision (OMNIVIS'00)*, pages 21–28, 2000.

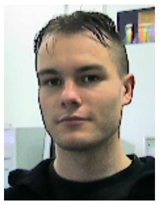
[34] Wiskott, L., Fellous, J.-M., Krüger, N., and v. d. Malsburg, C. Face recognition and gender determination. In *Proceedings International Workshop on Automatic Face and Gesture Recognition*, page 9297, 1995.

[35] Zobel, M., Denzler, J., Heigl, B., Nöth, E., Paulus, D., Schmidt, J., and Stemmer, G. MOBSY: Integration of Vision and Dialogue in Service Robots. In *Computer Vision Systems, Proceedings Second International Workshop (ICVS)*, pages 50–62, 2001.

Hans-Joachim Böhme has been an research assistant and lecturer at the Department of Neuroinformatics and Cognitive Robotics at the Ilmenau Technical University since 2002. He received his Diploma degree in Technical and Biomedical Cybernetics in 1989, his Doctorate degree in Neuroinformatics in 1991, and his Habilitation degree in Computer Science in 2002. His main research interests cover the areas of interactive autonomous robots, artificial neural networks, multi-modal human-machine interfaces, and service robot applications.

Andrea Scheidig studied Computer Science at the Technical University of Ilmenau from 1990-1996. Since 1996 she has been a scientific staff member at the Department of Neuroinformatics and Cognitive Robotics. 2003 she received a PhD in Neuroinformatics from Ilmenau Technical University. Currently, she holds a postdoc position, heading the HOROS project. Her research interests lie in the fields of user adaptive man-machine-interaction and communication, self-organization by means of senso-motorical interaction, and self-organizing action selection in systems with multiple neural agents.

Torsten Wilhelm studied Computer Science at the Ilmenau Technical University. Since 1999 he has been a PhD student at the Department of Neuroinformatics and Cognitive Robotics at the Ilmenau Technical University, where he received the PhD in 2005. His PhD research was concerned with multi-modal human-robot interaction, especially the development of algorithms for face analysis. Additionally, his research interests focus on modeling concepts for human-robot interaction and autonomous mobile robot control architectures.



Christof Schröter studied Computer Science at the Ilmenau Technical University. Since 2001 he has been a PhD student at the Department of Neuroinformatics and Cognitive Robotics at the Ilmenau Technical University. His thesis will deal with robot navigation in large-scale and crowded environments. His research interests lie in the fields of autonomous mobile robots, self localization, adaptive building of environment representations from sensorical input, and action planning and movement control.

Christian Martin studied Computer Science at the Ilmenau Technical University from 1998-2003. From 2001 to 2002 he was a visiting scientist at the Carnegie Mellon University (Pittsburgh, USA). 2001 he was one of the founders of the company MetraLabs GmbH. Since 2004 he has been a PhD student at the Department of Neuroinformatics and Cognitive Robotics at the Ilmenau Technical University. His thesis will focus on system integration for mobile interactive service robots. His research interests cover all fields of autonomous mobile robots.

Alexander König studied Computer Science at the Ilmenau Technical University from 1996-2002. Since 2002 he has been a PhD student at the Department of Neuroinformatics and Cognitive Robotics at the Ilmenau Technical University. His thesis will focus on vision-based methods for robot navigation, especially Monte-Carlo-Localization and multi-modal obstacle avoidance. Additionally, research interests concentrate on e-learning environments.

Steffen Müller studied Computer Science at the Ilmenau Technical University from 1999-2005. Since 2005 he has been a PhD student at the Department of Neuroinformatics and Cognitive Robotics at the Ilmenau Technical University. The related PhD project will emphasis self localization and navigation in dynamic real world environments as well as multi-agent-systems for adaptive generation of behavior and dialog control. Additionally, he is working on sophisticated control architectures for mobile and interactive robots.

Horst-Michael Gross has been a full professor of Neuroinformatics at the Ilmenau Technical University, Faculty of Computer Science and Automation and has been heading the Department of Neuroinformatics and Cognitive Robotics since 1993. He received his Diploma degree in Technical and Biomedical Cybernetics in 1985 and his Doctorate degree in Neuroinformatics in 1989. Among his main research interests are neural computing, autonomous robots, reinforcement learning, and vision-based human-robot interaction. He is a member of INNS and ENNS.