I'm Still Watching You: Update on Observing a Person in a Home Environment

Jens Kessler* Matthias Schmidt* Sandra Helsper* Horst-Michael Gross*
*Neuroinformatics and Cognitive Robotics Lab,
Ilmenau University of Technology,
Ilmenau,Germany

***Abstract*— Recently, a mobile robot was enabled to observe persons within their home in an unobtrusive way. This behavior is necessary during the long idle periods of the robot, where the robot has to stay somewhere near the user to be able to recognize new commands and tasks. Particle swarm optimization is used to find an optimal position. In this work, the ongoing progress of our approach is presented with focus on long term, real world capable improvements. Therefore, the online update of a person distribution and an elevation map are introduced. Finally, experiments show the robustness of the found optimal observation position.**

***Index Terms*— Social navigation, observing a person, particle swarm optimization, hard and soft criteria**

I. Introduction

The idea of using a mobile robot assistant in home and public environments has gained more and more attention during the last years. Recently, there are several prototypical systems available, that aim to help elderly people [1], [2], to assist care givers in hospitals [3], to guide people in supermarkets and home improvement stores [4], [5], or to provide an intelligent video-conferencing system [6]. The work presented here is focused on home environments and contributes to the ALIAS project (ALIAS = Adaptable Ambient LIving ASsistant). ALIAS has the goal of developing a mobile robot system to "interact with elderly users, monitor and provide cognitive assistance in daily life, and promote social inclusion by creating connections to people and events in the wider world" [7]. Here, especially the aspect of monitoring the user is the main focus of the presented work. To be precise: an *unobtrusive* observation behavior is the goal of the monitoring task. This behavior of a mobile robot is required whenever the robot is idle and has to stay in the vicinity of the user to recognize uttered or gestured commands of the user, without disturbing the user in its current activity like leisure, reading, or cooking. Within this context, the presented work contributes to the domain of social robotics and social compliant robot navigation. However, it is *not* concerned about interaction with a person, but only to enable the robot to signal readiness for interaction. To do so, we rely on the theory of proxemics, presented by Hall [8]. This paper is structured as follows: in section II current work within the domain of robotic person observation is shown, followed by a short summary of our recent work (see III). In the sections IV, V, and VI the main improvements of our current approach are presented, while section VII covers the experiments with a real-world capable observation system.

This work was financed by the project AAL-2009-2-049 "Adaptable Ambient Living Assistant" (ALIAS) co-funded by the European Commission, the Federal Ministry of Education and Research (BMBF), and the Thuringian Ministry of Science.

II. Related work

Psychologists investigated the human-to-human interaction in public areas very carefully since the 70s of the last century. One of the foundations is the work of Hall [8], who first introduced the concept of proxemics around a human being to support different modes of interaction. There is a space for non-interaction, public interaction, interaction with friend, and also an intimate space for interaction with very close relatives (see table I).

zone	interval	example situation
close intimate	0.0m - 0.15m	lover or close friend touching
intimate zone	0.15m - 0.45m	lover or close friend talking
personal zone	0.45m - 1.2m	conversation between friends
social zone	1.2m - 3.6m	conversation to non-friend
public zone	from 3.6m	no private interaction

TABLE I
PSYCHOLOGICAL DEFINITION OF THE PERSONAL SPACE AS SPECIFIED IN [8]. THIS SPACE CONSISTS OF FIVE ZONES, EACH SUPPORTING DIFFERENT ACTIVITIES AND DIFFERENT COMMUNICATION INTENTIONS.

Proof, that the personal space also exists between humans and robots, is given by exhaustive wizard of Oz experiments done within the COGNIRON project [9], [10], [11]. However, non of these works tried to solve an autonomous task with a robot.In our work, the distance of conversation space between the robot and a non-friend interaction partners is used to signal readiness, and the robot places itself at 2.4 meters distance towards the person at a feasible position autonomously. There are only a few works, dealing with finding an observation position using a mobile robot. Some publications focus on the task to observe a person to shoot pictures on social events [12], [13]. Here, the social distance is not in the focus of the observation pose, but the rotation of the robot in a way, that the image composition is good (faces are centered). Additionally to that, in [6] an approach is shown, where the robotic cameraman also creates an illumination model of the room and includes that into the optimization function to find the best position. This approach uses also the proposed Particle Swarm Optimization (PSO), but a different set of optimization criteria. A different approach is shown by [14], where test persons have to point out good waiting positions for a shopping assistant robot. From these positions a set of features is calculated, received from

manually labeled maps and person trajectory densities, and a support vector machine is trained to classify appropriate and inappropriate waiting positions. However, the goal was to place the shopping robot at a position, which does not disturb other customers during their shopping. But, the approach does *not* ensure the visibility of the assisted customer during the waiting time. So, the customer has to come back to the robot to continue assistance. Lately, the idea of [6] was used to present an approach to observe a person in a socially acceptable manner by using a set of hard and soft criteria and also the PSO to find the optimal position [15]. Since this paper improves this approach to provide a real world capable system, it is described in more detail in section III. Here, we use and maintain an elevation map to improve the visibility checks of our optimization approach. Elevation maps are pseudo 2D maps, where each cell represents the estimated height of obstacles over an assumed ground plane. In real world applications the elevation map of the robots environment is not known at the beginning and has to be estimated over time to reflect also the changes of the environment. There are different techniques known to estimate an elevation map. First, in [16] the measured height values are taken as they are without considering any measurement errors. A large group of publications exist, where the Kalman update is used to update the elevation value of each cell [17], [18]. Each approach uses different solutions to cover the characteristics of the mapped environment. In the work of [19] a maximum estimator is used to reduce sensor noise and in [20] a forget factor is introduced to include new observations into the map. An interesting work is presented in [21], where the localization errors and the sensor noise are used to update the elevation map with a full error propagation.

III. The Optimization Approach: A Short Summary

With the presented optimization approach in [15], the robot finds a position $\mathbf{x}$ with direct sight to the person at a distance of 2.4 meters, and looks towards that person with angle ϕ. This is the preferred spatial configuration of the robot and the user to signal readiness.

To compute a feasible position, the robot has to use (i) an occupancy map, (ii) the current person position, (iii) the current robot position, and (iv) a distribution of person occupancy, which reflects the most likely places the person usually rests at. The persons are tracked via the camera system and the laser scanner of the robot. The view direction is not estimated. The occupancy density is computed with all detected persons, not regarding their identities. These are the boundary conditions of the optimization process. Additionally, an elevation map is now used to predict the visibility $v(\mathbf{x}, \phi)$, to adapt the driveability $d(\mathbf{x})$ and to improve the soft criteria $c_{podf}(\mathbf{x}, \phi)$, which are all described later in this paper.

The optimization system consist of three main parts: (i) the elevation update process, (ii) the adaptation of the person occupancy density function, and the core of the optimization process (iii) the particle swarm optimization. The data flow is briefly sketched in Fig. 1. At this point a short introduction should be given to the presented approach from [15].

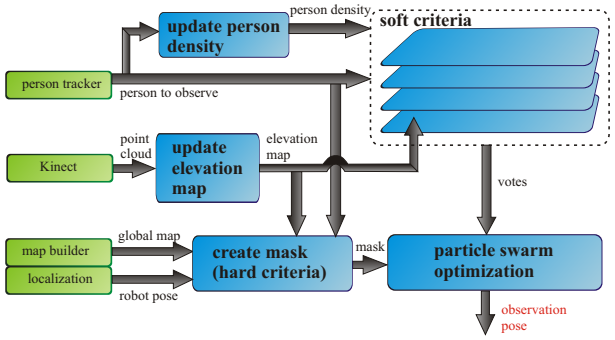


Fig. 1. Overview of the observation system (blue modules): there are two processes for adaptation towards the real world. First, the process to adapt the elevation map with incoming depth data, and second the process of updating the person density. From all information the hard criteria are extracted as a mask and the soft criteria are than used to distinguish the optimal position within that mask.

A. The optimization function

The optimization function f, defined in 1, reflects the different criteria to be considered, and fuses these criteria into a single function. There are two hard criteria to reflect physical properties to constrain the search space and mask out impossible search positions. These are the driveability $d(\mathbf{x})$, and the visibility of the person $v(\mathbf{x}, \phi)$. Both functions d and v are binary functions defined within the map space. Additionally, a set of soft criteria has to ensure: (i) an appropriate distance to the user (c_{dist}), (ii) the ability of the robot's sensors to detect a person from an observation pose (c_{det}), (iii) to perceive the person from the front (c_{front}), and (iv) how much of the person's occupancy distribution can be observed from that observation pose (c_{podf}). Since these criteria are no hard criteria, an optimal compromise between the linear superimposed criteria has to be found. So, the resulting optimization function is defined as follows, where all α_i values are set to $\frac{1}{N}$:

$$\begin{aligned} f(\mathbf{x}, \phi) \quad = \quad & d(\mathbf{x}) \cdot v(\mathbf{x}, \phi) \cdot [\alpha_1 \cdot c_{det}(\mathbf{x}) + \alpha_2 \cdot c_{dist}(\mathbf{x}) \\ & + \alpha_3 \cdot c_{front}(\mathbf{x}) + \alpha_4 \cdot c_{podf}(\mathbf{x}, \phi)] \end{aligned} \tag{1}$$

Note, that two new soft criteria are introduced here: (v) to stay near walls and (vi) to *not* stay within usual walking paths of the observed person. Both criteria are described later in section VI. Also, the soft criterion of c_{front} is not used any more, since it leads to unstable results when the person rotates or when small measurement errors on the view direction occurred.

B. Particle swarm optimization

The optimization problem is simply, to find the best values of $(\mathbf{x}, \phi)$ that maximizes the output of $f(\mathbf{x}, \phi)$, which is the optimal observation position. Our solution to the defined optimization problem uses the particle swarm optimization (PSO) approach. It is a well known technique (see [22], [23]) to find a global optimum by sampling from a defined optimization function, and uses a mixture of directed and random search within the search space to iterate towards the optimum. Each particle consists of a state $\langle x, y, \phi \rangle$,

where only $\langle x, y\rangle$ are part of the current optimization space, and a speed vector $\langle v_x, v_y\rangle$ also residing within that space. Note that ϕ is chosen to view directly towards the current person position. Particles are randomly initialized and the optimization function $f(\mathbf{x}, \phi)$ is calculated for each particle. For each particle two distinctive particle positions have to be remembered: one is the *local best* position that particle i could encounter, called $p_{[i]}^{[loc\ best]}$, and one is the *global best* value of all particles *ever* measured in all iterations, called $p^{[glob\ best]}$. The update of the particle speed and position is calculated iteratively:

$$\begin{aligned} \mathbf{x}_{t+1}^{[i]} &= \mathbf{x}_t^{[i]} + \Delta t \cdot \mathbf{v}_t^{[i]} \\ \mathbf{v}_{t+1}^{[i]} &= K\Big[\mathbf{v}_t^{[i]} + c_1 \cdot r_1 \cdot (p_{[i]}^{[loc\ best]} - x_{t+1}^{[i]}) \\ &\quad + c_2 \cdot r_2 \cdot (p^{[glob\ best]} - x_{t+1}^{[i]})\Big] \end{aligned} \tag{2}$$

The variables r_1 and r_2 are random numbers from an interval $[0..1]$. The parameters c_1 and c_2 are chosen to prefer either the local best particle or the global best parameter. We balance both parameters to $c_1 = 2.5$ and $C_2 = 2.5$. The update of the position and speed component is simple, but needs the constriction factor K to guarantee convergence.

$$K = \frac{2}{\left|2 - \theta - \sqrt{\theta^2 - 4\theta}\right|}\ , with\ \theta = c_1 + c_2, \theta > 4 \tag{3}$$

C. Implementation of the single criteria

1) Driveability: Here, we discuss the first criterion of equation 1: $d(\mathbf{x})$. This function selects all cells which could be reached by the robot. This function is either zero, when the cell is not reachable, or one, when this cell is reachable. This is simply done by dilating the occupancy map by the robot radius to assume a point like robot. Then flood-filling at the current robot position is applied to efficiently extract all reachable cells. The resulting mask defines $d(\mathbf{x})$ and a set X_d of reachable cells.

2) Visibility: The next function is the visibility criterion $v(\mathbf{x}, \phi)$. The task is to check every cell of X_d, if the person could be seen from that cell. In the previous approach this was done by assuming obstacles with infinite height. Now, with the help of the elevation map, the robot could estimate the visibility of the person with much more information. How the elevation map is build, will be described later. Here, the core idea how the visibility is checked efficiently is briefly described on a simple example (see Fig. 2).

To check visibility, a line of cells is processed from the center of a circle towards each cell of the circle perimeter. This covers an 2D area. On each line an elevation profile, coming from the elevation map, is checked for visibility, which is now an one dimensional problem. Each line to check visibility is processed in an ordered way from the center, where the observer is placed, to the perimeter. In each step, the slope $\Delta y/\Delta x$ of the line of sight from the center towards the current cell of the height profile has to raise compared to the previous cell. If that is not the case, the elevation value of that cell is not visible. In Fig. 2 this appears for cells 4 and 6. If the height of a cell is not visible, the current maximum slope $max(\Delta y/\Delta x)$ is used to calculate a virtual point at this cell and replace the height value with a *virtual height* (see Fig. 2, red circles). The new virtual height is guaranteed to be visible. The test for visibility is simple now: each cell could be tested, if the object to observe is above the virtual elevation profile, and therefore visible, or not.

3) Sensor distance: The sensor distance criterion $c_{det}(\mathbf{x})$ has to consider the ability to detect a person with a certain sensor. Due to our sensors the recognition distance s_{max} of a person is limited to 3 meters. So, the sensor distance $d_s = |\mathbf{x}_i - o_t|$, which is the Euclidean distance from the observed cell $\mathbf{x}_i$ towards the person position o_t, and the maximal sensor distance s_{max} is used to calculate c_{det}:

$$c_{det}(\mathbf{x}) = \begin{cases} 1 & ,if\ d_s < s_{max} - 1 \\ \frac{1}{1+exp(d_s - s_{max} - 0.5)} & ,else \end{cases} \tag{4}$$

4) Social distance: The social criterion c_{dist} should keep the robot away from the observed person. As Hall [8] explains, the social distance, where persons do not consider to interact with each other, is around 2.4 meters and above. This value is our social distance to ensure that an observed person feels comfortable. To consider this fact, a circular function is defined around the person, using the parameter σ_d to define the thickness of the circle:

$$c_{dist}(\mathbf{x}) = e^{-\frac{(d_s - 2.4)^2}{2\sigma_d^2}} \tag{5}$$

5) Person occupancy distribution: The function c_{podf} describes, which part of the person occupancy density function $p(o)$ can be seen from the given cell with a given view direction. A typical distribution of a corridor is shown in Fig. 3 (right). This criterion should prefer observation positions, where most parts of the distribution could be observed at the same time to keep the robot at its place, even if the person walks within the room. Like the calculation of the person

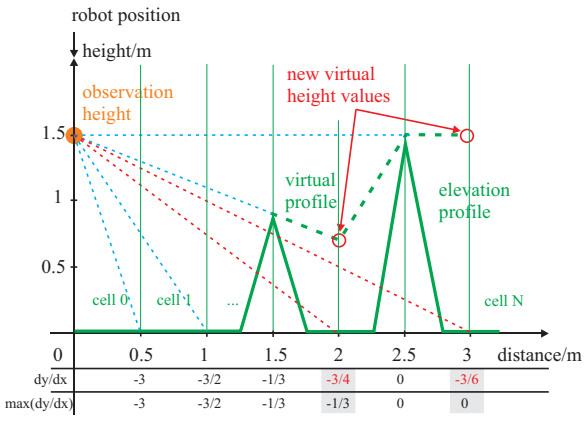


Fig. 2. Checking the visibility of a cell. The check is done from cell 0 (on the left) to cell N (right): if the slope $\Delta y/\Delta x$ from one cell to the next raises or is equal (shown by a dashed blue line), the point is visible. If the slope decreases (dashed red lines), the point on the elevation profile is invisible and is replaced by a virtual point (red circles). At the bottom the real slope and the virtual slope, that is used for interpolation, is shown.

visibility, the visibility of the distribution function has to be calculated. This is again done by casting multiple rays from the hypothetical camera view into the viewing cone and to check each cell, if a hypothetical person could be seen. Those cells are added to the visible mask X_{fv}. Now, all density values within the mask are summed up from $p(o)$.

$$c_{podf} = \sum_i p(o = \mathbf{x}_i) \ , if \ \mathbf{x_i} \in \ X_{fv} \tag{6}$$

IV. Updating the person occupancy distribution

One background process which is needed for the optimization, is the creation and maintenance of the person occupancy density function. In [15], the person occupancy function was recorded by a static Kinect camera and the point cloud was filtered by using the OpenNI library [24], so that only person points remain. These points were then used to build a voxel space histogram to estimate the probability distribution.

In the current version, the function representation is reduced to a 2D cell grid histogram. The histogram has exactly the same size as the global map. Since the person tracker of the robot is continuously running, all detected person hypotheses are now summed up in the cell bins, if the person is within the global map area. So, each cell counts how often a person was detected in the cell $o(x, y)$. Besides the cell count, the overall number of observations o_{total} is also counted. From these two values the probability of observing a person in that cell is simply estimated by:

$$p\left(o = (x, y)\right) = \frac{o(x, y)}{o_{total} \cdot x_{res}^2} \tag{7}$$

where x_{res} is the bin size in meters of all cells. The distribution is updated every N valid person observations, where N is chosen to be 10 in our system. With a tracker update rate of 200 ms, every 2 seconds a new distribution is created. In this way, the system is able to very efficiently keep the person occupancy density function up to date for the whole life time of the system.

V. Updating the elevation map

The second background process is the estimation of the elevation map. Here, the height of objects on the ground should be estimated. There are also some interesting challenges, which need to be addressed to get a usable elevation map. The principle idea is very simple: with the help of a Kinect camera, a dense point cloud is recorded and transformed into the global coordinate system. Now, the points could be assigned to grid cells and one cell covers a set of points. Each cell stores a height value, which estimates the height above a defined ground plane. The height values have to be updated with the help of the corresponding points from the point cloud.

A. Challenges on updating the elevation map

There are a number of challenges regarding the map update. First, there are a number of points for each cell, but only one representative height value could be used to update the elevation map. Second, not everything from a large structure like a door or a wall could be seen. This is due to the fact, that the camera has a certain viewing volume and all points outside this volume could not be observed (see Fig. 3, left). Here, especially the selection process from the first problem can issue some false representative values. Lastly, common measurement errors could lead to false measurements, introduced by (i) the sensor itself or (ii) by the transformation towards global coordinates, since localization errors of the robot could lead to false associations of points from the recorded point cloud to cells. Nevertheless, for the sensor noise problem there are a number of solutions known, and these solution are presented in section V-D and experimental results are shown in VII-A.

B. Initializing the map

The map is initialized with the occupancy map of our environment. Here, obstacles are used to set the height above ground to h_{max}, which is typically 1.8 meters, and all other cells are set to h_{min}, which is typically 0 meters. At this point, all obstacles are assumed to have the same height. The real height is estimated during the update process, so the elevation map could capture some of the properties of a 3D model.

C. Filtering wall structures

The aim of the elevation map is to model the object height from objects, which are placed on the ground and could potentially block the view towards other objects. Therefore, the point cloud has to be filtered to avoid unwanted results. First, and most simple, all points above the height h_{max} are erased from the point cloud, since no measurements from the ceiling should be included into the map. Otherwise, the elevation map has the ultimate height of the ceiling. Next, the remaining points are assigned to their corresponding cells. From each point set, a good representation value for the height of that cell should be found. For horizontal oriented surfaces, like a table, this could be done by simply calculating the mean height or choosing the maximum height $h_t^{[m]}$ as elevation, since all values of a cell have a low variance in the height dimension. This is different with vertical structures, like walls, since the mean height is a bad estimate of the elevation due to the high variance in the height dimension. Also, the maximum height is not stable, since the viewing volume of the camera is restricted and the vertical size of the structure could not be observed completely. This leads to different heights of a cell, depending on the camera position in the room. So, often the highest point currently observed does *not* represent the correct height of this cell since the true height is not observable. This is shown in Fig. 3 on the left side. To sum up, partially observable vertical structures have to be updated differently than full observable non-vertical structures. But how to separate vertical structures,

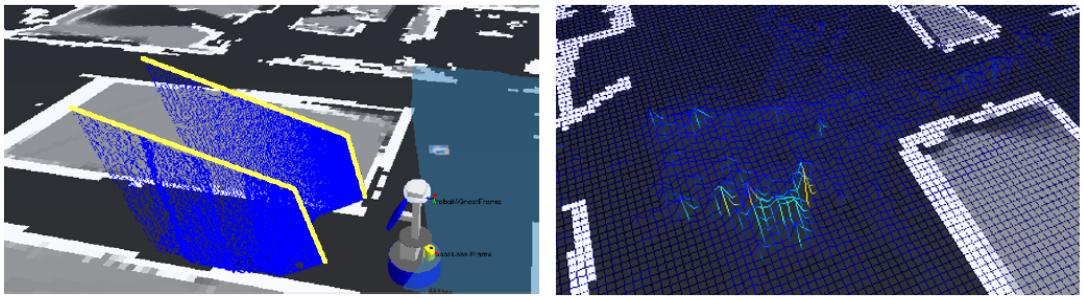


Fig. 3. On the left: possible error sources of the elevation map update: (i) the restricted viewing volume, where the yellow lines denote the border of the viewing volume. In vertical structures, like the walls seen in the image, parts could not be observed, which are important to estimate the correct height. (ii) Multiple points per cell lead to the problem of choosing a correct representative value. In all cases in this paper it is the maximum height of the points assigned to one grid cell. (iii) localization errors: if the robot is not correctly localized, the assignment of the points from the point cloud is done wrong. On the right: a typical person occupancy distribution of the corridor of our lab. Here, the normalized histogram is shown, which is updated every 2 seconds. High (yellow) values mean a high likelihood, that a person could be observed at that position. Low (blue) values denote a low likelihood.

like tables or chairs, from horizontal ones, like walls or lockers? In [18] the variance in height is used per cell to apply a simple threshold to classify both cases. This approach led in our experiments to a correct classification rate of 93.33 %. In contrast to that, a classification was implemented, where the difference of the maximum height and the minimum height of the points of one cell was used to classify vertical structures. This is twice as fast and leads to a correct classification rate of 91.40 %, which is sufficient in our case. Note, that this classifiaction approach is only evaluated in home environments and works well there. For in-between surfaces wrong classifications could occur, which could lead to slightly wrong elevation estimates.

D. Updating the map

1) Update of vertical structures: Structures from the current depth image, which are classified as "vertical", are assumed to be not completely observed. That is why the height value is simply the maximum of the old height value $h_{t-1}(x, y)$ and the currently measured height value $h_t^{[m]}(\mathbf{x})$, which is the maximum of all points in a cell:

$$h_t = max(h_{t-1}, h_t^{[m]}) \tag{8}$$

2) Update of non-vertical structures: If the structure is classified as non-vertical it is assumed to be completely observed and so, a sufficient representative could be found. The update could be done by (i) using the Kalman filter update from [18], or (ii) a running mean approach could be used ($\bar{h}_t = \alpha \bar{h}_{t-1} + (1-\alpha) h_t^{[m]}$), or (iii) the mean value of the last n measurements is used. The Kalman update uses the estimation of the current height $h_t^{[m]}$, the variance σ^2 of the height measurement, the previous estimated height h_{t-1}, and the variance of the own state estimation σ_{t-1}^2 of the last estimation step:

$$\begin{aligned} h_t &= h_{t-1} + \frac{\sigma_{t-1}^2 \cdot (h_t^{[m]} - h_{t-1})}{\sigma t - 1^2 + \sigma^2} \\ \sigma_t^2 &= \sigma_{t-1}^2 - \frac{(\sigma_{t-1}^2)^2}{\sigma_{t-1}^2 + \sigma^2} \end{aligned} \tag{9}$$

All described methods are compared in section VII-A and because of the experimental validation version (iii) is preferred.

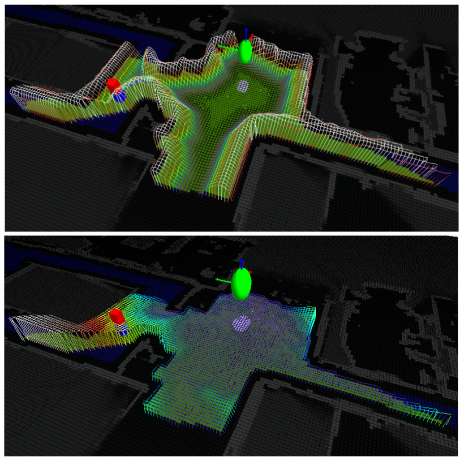


Fig. 4. Examples of the function from the wall criteria c_{wall} (top) and the path function c_{path} (bottom). Here, the environment of the wide corridor scenario is shown.

VI. Stay near walls & Avoid walking paths

In section III it was mentioned, that during our experiments, the soft criterion of the frontal view c_{front} was erased from the set of soft criteria due to stability problems. Also, two additional criteria are introduced, which are presented in the next sections. For both criteria the resulting functions are shown in Fig. 4.

A. Staying near walls

This criterion should enable the robot to stay near walls. It is calculated on a binary mask of the global map, where walls are selected. Then, the distance transform, described in [25], is used to calculate for every free cell the distance d to the nearest wall. With the optimal distance d_o the value for each vote is calculated as follows:

$$c_{wall}(x) = exp\left(-\frac{(d(x) - d_o)^2}{2\sigma_{wall}^2}\right) \tag{10}$$

B. Avoiding walking paths of the user

This criterion uses the aforementioned person distribution to select those cells, where the number of person observations is above a defined threshold. All cells above this threshold are marked as possible walking paths and the distance transform is also calculated to get the nearest distance to a path. The robot should not be placed near path cells. The result of the distance transform is used to prefer large distances with the parameters γ for the steepness and d_{min} to set the minimal distance to the path :

$$c_{path}(x) = 1/\left(1 + exp(-\gamma(d(x) - d_{min}))\right) \qquad (11)$$

VII. Experiments

There were two scenarios chosen for our experiments. First, a wide corridor (about 5 meters width) of our institute with two possible sitting locations, and our home lab with three sitting locations. We were interested in two points: first, which elevation map update method is robust against sensor noise and changes within the environment? And second, what parameters are best for the particle swarm optimization to get stable results with minimal computational effort?

A. Comparison of different update methods for elevation maps

The first experiment investigates the different update methods: here, the elevation map was initialized with height zero and than a sequence of measurements with an empty floor was done. Afterwards, a chair was placed on the floor and the adaptation was recorded until it converged. Then, the chair was removed and again convergence was observed. For all experiments, the classification into vertical and non-vertical structures with the minimum-maximum difference was used with a threshold of 0.25 meters for the difference value. All parts of the chair were correctly recognized as horizontal structures.

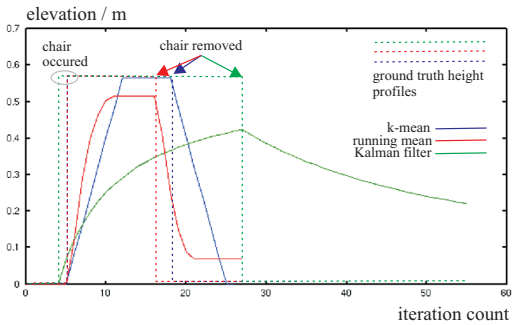
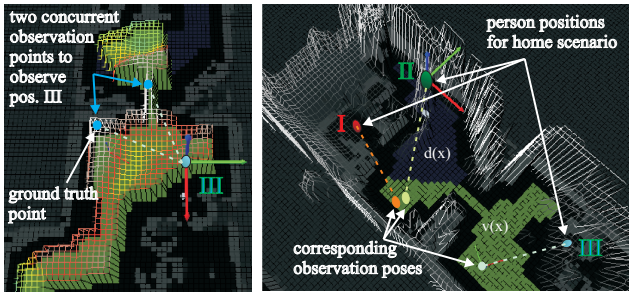


Fig. 5. Comparison of all three elevation map update methods at one cell with an occurring and afterwards vanishing chair. The dashed lines are the ground truth elevation values. The thick lines are the measured adapted height values. The corresponding colors to the used methods are shown on the right.

In Fig. 5 it could be seen, that the k-mean value is the most robust and fastest adaptation method. While the running mean tends to be slower and introduces minor errors in the estimated values, the Kalman filter approach from [18] is insufficient, since it contains high estimation errors and slow adaptation rates. The main reason for that effect is the constantly decreasing variance, since this system was designed to estimate static elevation maps. In our system we used the k-mean adaptation method for all remaining experiments. However, we also tried to compensate the error of elevation values due to wrong localization, but none of the tested methods was able to correct such errors, if most observations come from a wrong localized robot.

B. Stability of the particle swarm



a) b)

Fig. 6. a) shows the full optimization function for the home scenario. Note, that this particular person position leads to two very equal local maxima, which in turn leads to a random choice of the PSO of one of these maxima. In b) all tested person positions for the home scenario are shown. Also, the corresponding best observation positions are shown. The colors of the positions correspond to the colors in the statistical results from Fig. 7.

In the two tested scenarios, a person is placed on each of the five possible resting positions (see Fig. 6b for the three positions of the home scenario). The person occupancy distribution was learned before the experiments and re-used and further updated during the experiments. It consists mainly fife peaks, each on a resting position. For each position a brute force calculation of all cells was performed to get the ground truth, where the best observation cell is. Then, the parameters of the PSO (number of particles, number of iterations) are set and after the defined number of iterations, the resulting position is compared with the ground truth position. This procedure was repeated 20 times for each possible resting position. So, for all positions in each scenario the mean error and the error variance is calculated. The results are shown in Fig. 7. From that figure it could be seen that 30 particles are enough to compute a stable result, and using more particles does not decrease the error significantly.

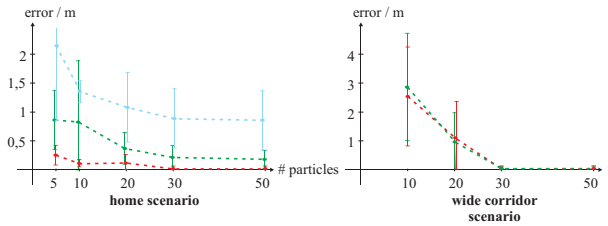


Fig. 7. The mean error with variance for all positions in the two investigated scenarios with respect to particle count. In all cases, 30 iterations are used for the particle swarm.

Note, that in the third position of the home scenario the error after using 50 particles is still high (around 80 cm). This is caused by the fact, that the optimization function is multi-modal here. The equal good positions are shown in figure 6 a). This multi-modality leads the PSO to randomly choose one of both positions. This leads to a high error in the statistical investigations. It could also be seen, that the wide corridor scenario is more stable than the narrow and jagged home scenario. This is simply caused by the ability of the particles to move more freely in wide areas as shown in Fig. 4. Still, the results for the narrow scenario are within 20 cm of error, which is sufficient for our purpose. But, problems with the PSO should be expected on very narrow, corridor-like structures, where a simple brute-force approach may be more reliable. Since 30 particles seems to be somehow sufficient, the number of iterations are investigated further. The results are shown in table II

# iterations	avg. error	σ
5	1.35m	1.23m
10	0.88m	1.05m
20	0.37m	0.27m
30	0.03m	0.08m
50	0.02m	0.05m
100	0.03m	0.06m

TABLE II

INVESTIGATION ON CONVERGENCE DEPENDING ON THE ITERATION COUNT. THE NUMBER OF PARTICLES IS FIXED TO 30.

From Fig. 7 and II it could be seen, that 30 particles and 30 iterations are enough to get a good convergence. Since one calculation costs on average 18,3 ms, the overall calculation time is 16 seconds. Also, gradient ascention was tried to further reduce the number of sampling steps and decrease the calculation time. But the results were worse in terms of average error than all trials from Fig. 7 and are not further discussed. The bad performance of gradient search is due to the form of the optimization function with many local maxima and wide planes without significant slope (see Fig. 6a)).

VIII. CONCLUSIONS

In this paper an update of our observation approach is shown. Now, an elevation map is used to predict the visibility of a person and the person density is constantly updated. The investigation of the elevation map updates revealed, that the simple k-mean value is best suited to update the map. Anyhow, it is not possible to suppress errors from a wrong robot localization. To overcome this issue, a full SLAM approach has to be developed in future. Also, the optimization function could sometimes lead to multiple equal good observation positions, as it was shown in position three of the home scenario. The question, how to reliably chose one of these points remains to future investigations. Anyhow, all of these points are good places to observe a person, so bad positions could be suppressed in general.

a) Comparison to the previous version:: In the updated version it appeared that the particle swarm is neither more stable nor faster. In fact, by using the elevation map the processing time has increased from 10 to 16 seconds, but now the system could simply include *more* possible observation points due to the improved visibility check. Additionally, we rejected the use of the person's view direction since this value leads to large oscillations with only small amounts of sensor noise. We further added two new criteria to keep the robot near walls and away from walking paths.

REFERENCES

[1] H.-M. Gross et al., "Progress in developing a socially assistive mobile home robot companion for the elderly with mild cognitive impairment," in *Proc. IEEE/RSJ-IROS*, San Francisco, USA, 2011, pp. 2430–2437.

[2] H.-M. Gross et al., "Further progress towards a home robot companion for people with mild cognitive impairment," in *Proc. on IEEE-SMC*, Seoul, South Korea, 2012, pp. 637–644.

[3] F. Weisshardt et al, "Making high-tech service robot platforms available," in *Joint Int. Conf. ISR/ROBOTIK2010*, 2010, pp. 1115–1120.

[4] T. Kanda, M. Shiomi, Z. Miyashita, H. Ishiguro, and N. Hagita, "A communication robot in a shopping mall," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 897–913, 2010.

[5] H.-M. Gross, H.-J. Boehme, Ch. Schroeter, St. Mueller, A. Koenig, E. Einhorn, Ch. Martin, M. Merten, and A. Bley, "Toomas: Interactive shopping guide robots in everyday use - final implementation and experiences from long-term field trials," in *Proc. IROS*, St. Louis, 2009, pp. 2005–2012.

[6] Ch. Schroeter, M. Hoechemer, St. Mueller, and H.-M. Gross, "Autonomous robot cameraman - observation pose optimization for a mobile service robot in indoor living space," in *Proc. ICRA*, Kobe, Japan, 2009, pp. 424–429.

[7] F. Walhoff and E. Bourginion, "Alias project description," ALIAS home page, http://www.aal-alias.eu/content/project-overview.

[8] E.T. Hall, *The hidden dimension*, Doubleday, NY, 1966.

[9] K. Dautenhahn et al, "How may i serve you? a robot companion approaching a seated person in a helping context," in *Proc. HRI*, 2006, pp. 172–179.

[10] K. Koay et al, "Exploratory study of a robot approaching a person in the context of handing over an object," in *AAAI Spring Symposia*, 2007.

[11] L Takayama and C. Pantofaru, "Influences on proxemic behaviours in human-robot interaction," in *Proc. IROS*, 2009, pp. 5495–5502.

[12] M.J. Kim, T.H. Song, S.H. Jin, S.M. Jung, G.H. Go, K.H. Kwon, and J.W. Jeon, "Automatically available photographer robot for controlling composition and taking pictures," in *Proc. IEEE/RSJ IROS*, 2010, pp. 6010–6015.

[13] Z. Byers, M. Dixon, K. Goodier, C. Grimm, and W. Smart, "An autonomous robot photographer," in *Proc. IEEE/RSJ IROS*, 2003, pp. 2636–2641.

[14] T. Kitade, S. Satake, T. Kanda, and M. Imai, "Understanding suitable locations for waiting," in *Proc. ACM/IEEE HRI*, 2013, pp. 57–64.

[15] J. Kessler, D. Iser, and H.-M. Gross, "I'll keep you in sight: Finding a good position to observe a person," in *Proc. IEEE/RSJ IROS*, 2012, pp. 4392–4398.

[16] I. Kweon, *Modeling Rugged Terrain by Mobile Robots with Multiple Sensors, PHD Thesis*, Carnegie Mellon University, Pittsburgh, 1991.

[17] A. Kleiner and C. Dornhege, "Real-time localization and elevation mapping within urban search and rescue scenarios," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 723–745, 2007.

[18] P. Pfaff, R. Triebel, and W. Burgard, "An efficient extension of elevation maps for outdoor terrain mapping," in *In Proc. of the International Conference on Field and Service Robotics*, 2005, pp. 165–176.

[19] C. Ye and J. Borenstein, "A new terrain mapping method for mobile robots obstacle negotiation," in *In Proc. of the UGV Technology Conference at the 2002 SPIE AeroSense Symposium*, 2003, pp. 21–25.

[20] S. Kagami, K. Nishiwaki, J.J. Kuffner, K. Okada, M. Inaba, and H. Inoue, "Vision-based 2.5d terrain modeling for humanoid locomotion," in *In Proc. of the IEEE International Conference on Robotics and Automation*, 2003, vol. 2, pp. 2141–2146.

[21] I. Miller and M. Campbell, "A mixture-model based algorithm for real-time terrain estimation," *Journal of Field Robotics*, vol. 23, no. 9, pp. 755–775, 2006.

[22] R. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proc. of Congress on Evolutionary Computation*, 2001, vol. 1, pp. 81–86.

[23] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. of the Congress on Evolutionary Computation*, 2000, vol. 1.

[24] WillowGarage, "Openni," http://www.openni.org.

[25] G. Borgefors, "Distance transformations in digital images," in *Computer Vision, Graphics, and Image Processing*, 1986, vol. 3.