Combining Computer Graphics and Computer Vision for Probabilistic Visual Robot Navigation

B. Heigl[a], J. Denzler[ab], and H. Niemann[a]

[a]Lehrstuhl für Mustererkennung, Universität Erlangen–Nürnberg, Erlangen, Germany
[b]Department of Computer Science, University of Rochester, Rochester (NY), USA

ABSTRACT

In this contribution we present how techniques from computer graphics and computer vision can be combined to finally navigate a robot in natural environment based on visual information. The key idea is to reconstruct an image based scene model, which is used in the navigation task to judge position hypotheses by comparing the taken camera image with a virtual image created from the image based scene model. Computer graphics contributes to a method for photorealistic rendering in real–time, computer vision methods are applied to fully automatically reconstruct the scene model from image sequences taken by a hand–held camera or a moving platform. During navigation, a probabilistic state estimation algorithm is applied to handle uncertainty in the image acquisition process and the dynamic model of the moving platform.

We present experiments which proof that our proposed approach, i.e. using an image based scene model for navigation, is capable to globally localize a moving platform with reasonable effort. Using off–the–shelf computer graphics hardware even real–time navigation is possible.

Keywords: robot navigation, particle filters, image based rendering

1. INTRODUCTION

In the past three years it could be observed that computer vision and computer graphics are moving close together. Augmented reality is an active research area, where both communities need each other. In this paper we present another area where topics usually found in computer graphics — namely the so–called *image based rendering technique* — are an important part of a classical computer vision task — namely *visual navigation* of an autonomous mobile system. The contribution of our work is, that we show

1. how an image based scene model can be used to create synthetic but nevertheless photorealistic images in real–time, taken from any virtual viewpoint,
2. that our image based scene model is capable also to model specular reflection and geometric structure in the world, where the use of a geometric model (for example CAD model), is almost impossible, and
3. how virtual views of the scene can be used to globally localize a moving platform with high accuracy based on visual information using a probabilistic localization approach.

The image based scene model that we use will allways need the position and projection parameters of the hand–held camera for each frame of the input sequence. To get them, structure from motion techniques can be applied. In previous works[1,2] we have shown, how to extend the methods known from literature to be able to handle especially the case of image sequences consisting of hundereds of images, which are necessary when recording light fields.

Our results can be used in several ways. First, the image based scene model can be used to predict the appearance of the scene assuming a certain position of the camera in the world. This includes also existence of certain features (light straight line, corners, etc) dependent on the viewpoint, for example in the case of specular reflections. In our work we concentrate on the scene model; the features that are used for localization of the robot are based on the whole image, i. e. we compare pixel by pixel the image which has been predicted, with the image, which is seen

Further author information: (Send correspondence to B. Heigl)
B. Heigl: E-mail: heigl@informatik.uni-erlangen.de
J. Denzler: E-mail: denzler@cs.rochester.edu

by the camera. Secondly, because of the scalability of the quality and resolution of the scene model, the approach can be used either for global localization, when almost no information about the position of the robot is available, or for local refinement of an position estimate. The latter one might be necessary for navigating between doors or accurate docking tasks. Third, our work can be used in any environment, where a classical geometric model is difficult to reconstruct or almost impossible. And finally, the image based scene model can be ideally combined with any other state estimation problem besides localization and navigation, where some information extracted out of the image must be compared with the expectation given a certain state estimate. Thus, the model fits optimally into the framework of probabilistic state estimation using particle filters.

The approach presented in this paper is related to and motivated by two previous publications of other authors on probabilistic localization and visual navigation. In the paper of Fox, e. a.[3] an approach has been presented for probabilistic self–localization of a mobile system based on classical robotic sensors. The approach was extended in the work of Dellaert, e. a.[4] to visual information from the ceiling of a museum. In that class of probabilistic methods the observation model — spoken in terms of statistics *the likelihood* of observing information given the current state of the system — must be evaluated repeatedly. This means for visual navigation that features computed in the camera image must be compared with features which would have been observed if being at a certain position. The comparison in combination with the a priori information of being at a certain position leads to the a posteriori probability of the position. Self–localization then means maximizing the a posteriori probability with respect to the unknown position.

In the work of Dellaert[4] a map of the ceiling is manually built and for each position the corresponding camera image was rendered. This can be done without much effort, because the camera points perpendicular to the ceiling. As a result the degree of freedom of the system is three, i. e. the movement in a plane parallel to the ceiling and rotating around an axis which is parallel to the optical axis. Having such a configuration, the rendered images depend only on the position of the system and the rotation angle, since the distance to the ceiling is known and constant. This makes rendering fast and map representation as simple as possible.

In our work we extend this approach by using the idea of the so–called *light–field* or *lumigraph* approach.[5,6] In order to represent a scene, no explicit and complete geometric model needs to be constructed. In contrast, the scene is represented by a certain amount of images together with an in accuracy scalable, local geometric approximation . With this representation photo–realistic effects like mirroring or specular reflection can be modeled even for very complicated objects, where geometric modeling is very difficult or nearly impossible. Having such an image based model of a scene certain algorithms exist for fast rendering of *new and yet unsighted* views. This means that given an arbitrary position in the world, the corresponding image which would be seen by the camera can be computed. This is exactly the demand for visual navigation, where no fixed or known relationships for the camera/scene configuration exist. Another advantage is that the light–field, i. e. the model of the scene, can be reconstructed automatically without user interaction by methods from structure from motion.[1]

The idea and the structure of our paper is the following. First a light–field is reconstructed by using only a very rough or no geometrical approximation, computed automatically by the approach described in a paper of Heigl, e. a.[7] (Section 2). During self–localization, particle filters are used which propagate the conditional probability of being at a position in the world given the observed data (Section 3). Section 4 describes how to bring together these two approaches for the task of visual navigation. In Section 5 we present experimental results for docking a robot to a certain position in the world based only on visual information and a model of the world presented by the automatically generated light–field. The results show at least, that our image based model is accurate enough to outperform localization and navigation based on odometry information. The paper ends with a summary and an outlook to future work (Section 6).

2. IMAGE BASED MODELING AND RENDERING

For visual robot navigation, we need a powerful visualization model which can be acquired easily and which can be used to render views in real–time. As we don't want to restrict our environment to special geometric, surface, or illumination properties, we need a model which is able to create photo–realistic views in spite of complex geometry and specular effects.

One concept fulfilling these requirements is the so–called *light–field*[5] or *lumigraph*[6] approach, which is an image based method for visualizing scenes in real–time. The main idea is the following. If a single view is taken from a scene, it can be interpreted as a bundle of viewing rays coinciding in the projection center of the camera. Having

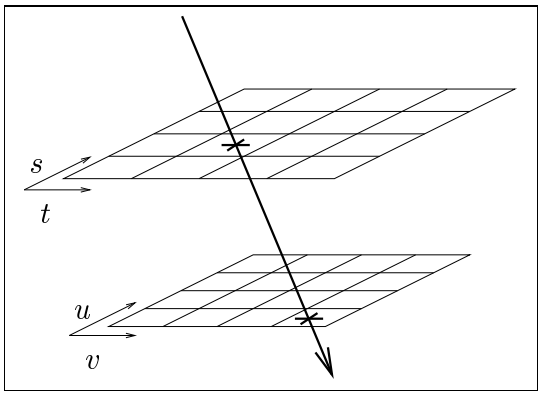

Figure 1. The light–field data structure. A viewing ray is parameterized by a quadruple (u, v, s, t). For each viewing ray a color value is stored.

many of such scene views, one gets a more or less dense sampling of all possible viewing rays within the scene. To render new virtual views, the required viewing rays must be interpolated from the discrete sampling. If information about scene geometry is known, it can be used to improve the interpolation. In contrast to geometry based methods, this approach implicitly handles non–Lambertian effects like mirroring and specularities and even complex geometries like fur and hair.

The light–field data structure provides a discrete 4–D parameterization of viewing rays by connecting discrete grid positions on two fixed parallel planes. Figure 1 shows such a configuration. On each plane, a local coordinate system is defined, which is able to address each point by two coordinates s, t or u, v, respectively. A quadruple (u, v, s, t) specifies one point on each of the two planes. The intersection line between those points corresponds to the according viewing ray. Usually, these coordinates are integer values and therefore only fixed discrete samples of viewing rays can be stored in a given data structure of this type. There exists a special interpretation of the two planes which is useful when creating a light–field from real camera images. Suppose the case that the grid points on the st-plane are projection centers of cameras. By connecting one st-point with any grid point in the uv-plane, all viewing rays of the camera view are specified which correspond to a pixel color value.

This fixed spatial arrangement requires either that the camera view points are ordered in a grid or that this fixed data structure has to be resampled from arbitrary views. The first alternative is difficult to achieve, as it requires a complicated technical equipment for moving the camera. The second has the disadvantage, that image information has to be resampled twice: first for resampling the data structure and second to render a virtual view. Furthermore, one has to choose a fixed sampling resolution for all images, which leads to subsampling or oversampling effects.

To avoid all these disadvantages caused by using the light–field data structure, we have developed a new method for rendering virtual views *directly* from real camera views.[7] Depth information given by local depth maps can be considered to improve rendering quality in an adjustable quality.

The basic principle of the method is the possibility to project points of a plane into a camera by a single 3×3 projective mapping matrix $\boldsymbol{B}$. This process also can be reverted so that each pixel of an image is projected onto a plane by multiplication of its coordinates with $\boldsymbol{B}^{-1}$. Having a real camera (subscribed with the image number i) and a virtual camera (subscribed with V), the whole mapping from the real camera to the virtual one via a given plane can be calculated by the multiplication with the 3×3 matrix $\boldsymbol{B}_V \boldsymbol{B}_i^{-1}$. Figure 2 visualizes this procedure.

If the scene surface corresponds to this plane and the surface is Lambertian, this mapping will result in an optimal rendered view. Notice, that if the virtual view point is near the view point of the real camera, even specularities and small deviations of the plane from the true geometry affect the rendered view just slightly. This effect can be exploited when having many real views from many different view points by suitably weighting those real camera views which are most adjacent to the virtual view.

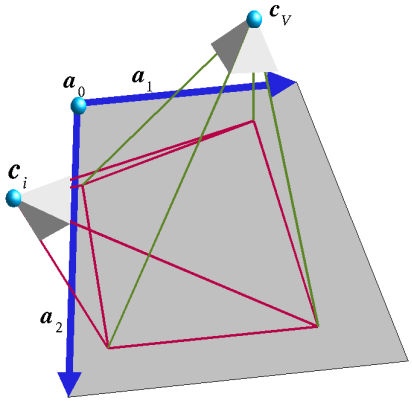

Figure 2. Mapping a camera image into a virtual view via a given plane.

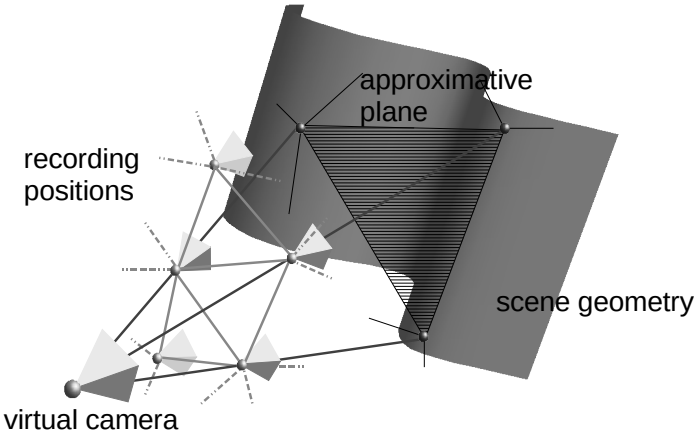

Figure 3. Drawing triangles of neighboring projected camera centers. The scene geometry is approximated by a single plane.

To determine these neighboring real cameras, their projection centers are projected into the virtual camera and transformed to a net of triangles by Delauny–Triangulation. Within one such triangle, parts of those images are superimposed which correspond to the triangle corners. In the implementation each triangle is drawn for each contributing image once, therefore three times. To get the corresponding pixels, the upper described transformation is applied backwards resulting in the transformation matrix $\boldsymbol{B}_i\boldsymbol{B}_V^{-1}$. The weighting factor of each overlayed triangle is 1 at the corner which corresponds to the contributing image and 0 at the others. In between the weighting factors build a flat ramp, similar to Gouraud Shading.

In the simplest case we can use a single plane to approximate the scene surface for the upper described mapping process. This may result in ghosting artifacts at those parts of the surface, where the scene geometry largely deviates from this plane. To reduce this effect, approximating triangles can be used instead of one single plane.

The corners of these triangles can be calculated from a depth value which is available by a simple look–up in the depth map corresponding to a real camera. The principle can be seen in Figure 3. The connection lines of the virtual camera center with each camera center of the recording positions are intersected with the scene surface resulting in a corner of an approximating triangle. To get this intersection point, we can use the distance of a recording camera to the scene in that viewing direction which is given by the upper noticed connection line. This distance can be determined by a simple look-up in the depth map corresponding to the real camera view.

The single triangles of the projected net may become very large if the virtual camera is situated very close to the recording positions and as a consequence, the 3-D triangles also cover large parts of the scene surface and the approximation of the scene geometry becomes worse. To avoid this effect each triangle can be subdivided further by inserting three new points at the medians of the sides.

By this procedure, the reconstructed scene surface changes with the change of the virtual view point. This approximation covers adaptively exactly all those parts of the scene geometry which are relevant to the actual view. For example, invisible concavities are not reconstructed. So the costs to render the triangles are reduced enormously compared to the case, when using a consistent geometrical model for all virtual viewing points commonly.

Notice, that the whole method also is applicable without any changes, when the virtual camera is situated between the recording positions and the scene. In this case, the same formulas for mapping are applied. The camera centers of the recording positions are projected by the usual multiplication with the projection matrix, ignoring the fact, that the projected point then lies behind the virtual camera, resulting in a mirrored projected triangle net. This triangle net exactly reflects which real views contribute to a given image pixel in the same sense as mentioned above.

3. PROBABILISTIC SELF–LOCALIZATION AND NAVIGATION

In the this section, we summarize a framework for probabilistic localization, basically in accordance with the work of Dellaert, e.a..[4] Knowing the position of the robot in world coordinate system, a sequence of small motions of

the robot to a predefined goal position can be done followed again by a self–localization. We will denote this loop of localization and movement as docking mechanism.

3.1. The Framework

In the previous section we have introduced an image based scene model. The scene model is now applied to vision based localization of a mobile platform. Vision based localization means, that based on an image $\boldsymbol{o}_t$ taken at time step t the position and heading $\boldsymbol{x}_t$ of the moving platform is estimated. An advantage of localization of a moving platform is, that information of the relative change of position and heading $\boldsymbol{m}_t$ is available, too. This relative movement information usually comes from the odometry of the moving platform. Odometry is a very accurate cue in the sense of small, relative motions, but highly inaccurate in a global sense.

The described localization problem can also be seen as a state estimation problem of a dynamic system. The dynamic is given by the performed motion $\boldsymbol{m}_t$ at time step t. The state is the position and heading $\boldsymbol{x}_t$ of the platform. The observation is the image taken at time step t. Since each of these quantities, i.e. the motion and the observation, are usually disturbed by noise, we define them as random variables. The main goal is to estimate the position of a robot given a sequence of actions $\boldsymbol{m}_t$ each of them followed by observations $\boldsymbol{o}_t$. Written in a probabilistic framework we seek for the most likely position

$$\boldsymbol{x}_t^* = \underset{\boldsymbol{x}_t}{\operatorname{argmax}}\, p(\boldsymbol{x}_t|\mathcal{H}_t) \tag{1}$$

$$= \underset{\boldsymbol{x}_t}{\operatorname{argmax}}\, \frac{1}{c} p(\boldsymbol{o}_t|\boldsymbol{x}_t) \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{m}_t) p(\boldsymbol{x}_{t-1}|\mathcal{H}_{t-1}) d\boldsymbol{x}_{t-1} \tag{2}$$

where $\mathcal{H}_t = \{\boldsymbol{m}_t, \boldsymbol{o}_t, \boldsymbol{m}_{t-1}, \boldsymbol{o}_{t-1}, \ldots, \boldsymbol{m}_0, \boldsymbol{o}_0\}$ is denoted as the history at time t. The step from (1) to (2) arises from applying the Bayes rule (with c being a normalizing constant independent of the argument $\boldsymbol{x}_t$) and assuming a Markov state for $\boldsymbol{x}_t$, i.e. the state $\boldsymbol{x}_t$ only depends on the previous state $\boldsymbol{x}_{t-1}$ and the currently chosen action $\boldsymbol{m}_t$. More details concerning this step can be found, for example, in the original paper about Condensation[8] or in textbooks about state estimation.[9]

The right hand side of equation (2), which involves a recursion from time step $t-1$ to time step t, can be divided into two steps for interpretation reasons: First, given a probability measure over the possible positions at time step $t-1$, $p(\boldsymbol{x}_{t-1}|\mathcal{H}_{t-1})$ and a certain movement action $\boldsymbol{m}_t$, from which we know the statistical properties $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{m}_t)$, we can update the a priori probability for being at a certain position $\boldsymbol{x}_t$ at time t. In the second step, the robot senses observations at position $\boldsymbol{x}_t$, modeled by $p(\boldsymbol{o}_t|\boldsymbol{x}_t)$ which allows for updating the belief $p(\boldsymbol{x}_t|\mathcal{H}_t)$ for being at that position at time step t.

Most approaches to the problem of localization of a mobile platform differ in the treatment of the likelihood function $p(\boldsymbol{o}_t|\boldsymbol{x}_t)$. Feature based approaches use a manually created scene map, for example CAD models, to match the observed features in the image with expected ones in the map. High correspondence in the feature matching process then means confidence in the estimated position $\boldsymbol{x}_t$. Landmark based algorithms work in a similar way. In our approach, the lightfield provides an image based scene model, which is the most general case, since such a model also allows a feature matching strategy in the process of computing the likelihood function. Additionally, the model is generated automatically as described in the previous section and can model specular reflections, which are viewpoint dependent. Such artifacts cannot be treated by pure geometric scene models.

3.2. Particle Filters

The problem is now, how to compute and propagate $p(\boldsymbol{x}_t|\mathcal{H}_t)$ over time. Since in general the probability density functions involved in this process cannot be given in closed forms, especially the likelihood function which depends on the sensed data, one cannot solve (2) directly. The famous Kalman filter[10] and the extensions of it (for example, the extended Kalman filter[11]) has been used over 20 years in computer vision and robotic for state estimation. The Kalman filter is an adequate way for solving (1), if the underlying assumptions (Gaussian noise, linear state transition, unimodal state distribution for $\boldsymbol{x}_t$) are fulfilled. In most cases, when working with images in natural scenes — which means high background clutter and ambiguities — at least the treatment of the state distribution as a unimodal Gaussian one is violated. Especially in the beginning of the localization process, when several positions and headings $\boldsymbol{x}_t$ are plausible, the approximation of $p(\boldsymbol{x}_{t-1}|\mathcal{H}_{t-1})$ by any kind of multimodal distribution is more

natural and useful. Then of course, the problem is how to evaluate the integral in (2), which can be done in a straight forward way for Gaussian densities.

During the past years so–called *particle filters* got an enormous interest in computer vision[12,8] and robotics.[3] Particle filters allow to estimate and propagate moments of certain probability density functions without having an explicit formulation of the density. They are also denoted as Monte Carlo Methods[13] or Condensation algorithm.[12]

Without going into detail (see for example the paper on original paper about the Condensation[12] for a deep discussion), particle filters can be briefly summarized as follows. We approximate $p(\boldsymbol{x}_t|\mathcal{H}_t)$ by a set of m so–called *particles*. Each particle consists of a *state value*, $\boldsymbol{x}_{t,i} \in \mathbb{R}^n, 0 \leq i < m-1$, and a *probability* or plausibility, $p_{\boldsymbol{x}_{t,i}} \in [0.1]$, of being in this state. The number m of particles has a direct influence on how accurate the density $p(\boldsymbol{x}_t|\mathcal{H}_t)$ is approximated by this particle set. It can be shown that for $m \to \infty$ the particle set converges weakly towards the density $p(\boldsymbol{x}_t|\mathcal{H}_t)$.[12]

To propagate the density over time, i.e. to evaluate (2), the corresponding particle set must be propagated, which includes the application of the dynamic model $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{m}_t)$ to the state set and the evaluation of the likelihood function $p(\boldsymbol{o}_t|\boldsymbol{x}_t)$. For this, particles $\boldsymbol{x}_{t,i}$ are drawn from the particle set with probability proportional to $p_{\boldsymbol{x}_{t,i}}$, and propagated by drawing a sample $\boldsymbol{x}_{t+1,i}$ from $p(\boldsymbol{x}_{t+1,i}|\boldsymbol{x}_{t,i}, \boldsymbol{m})$. The probability $p_{\boldsymbol{x}_{t+1,i}}$ of the new particle is then updated by $p(\boldsymbol{o}_t|\boldsymbol{x}_{t+1,i})$, including a final normalization such that $\sum_i p_{\boldsymbol{x}_{t+1,i}} = 1$. Problems, which have to be solved, are the sampling mechanism (likelihood weighted sampling, factored sampling, importance sampling[13]), i.e. how to draw samples from a probability distribution, and the number of particles to approximate the density with the necessary accuracy.

The main point of our approach is, that this frameworks can be used in an ideal manner with the image based scene model. Every particle $\boldsymbol{x}_{t+1,i}$ in the particle set represents a hypothesis for the position and heading of the mobile platform together with a probability $p_{\boldsymbol{x}_{t+1,i}}$, which measures the likelihood that the platform is at position $\boldsymbol{x}_{t+1,i}$. The probability $p_{\boldsymbol{x}_{t+1,i}}$ is updated by $p(\boldsymbol{o}_t|\boldsymbol{x}_{t+1,i})$, which makes it necessary to compare the expected image with the taken one. Having the image based scene model, for an arbitrary position and heading $\boldsymbol{x}_{t+1,i}$ the expected image can be rendered efficiently as described in the previous section. In the next section we will explain the whole process in more detail.

4. BRINGING TOGETHER COMPUTER VISION AND COMPUTER GRAPHICS

The last two sections have shown how computer vision and computer graphics grow together. For self–localization using particle filters the likelihood in (2) must be evaluated repeatedly. This means that from an environmental map virtual views of the scenes must be rendered very quickly. This is a classical computer graphics task. Since the rendered images shall be compared with images of the scene taken by a camera, photo–realistic rendering is necessary. Both demands are fulfilled by the light–field approach. On the other side for reconstructing the light–field of a scene automatically classical computer vision algorithms (structure from motion) are necessary, which closes the loop: computer vision needs computer graphics and vice versa. The complete approach for localization of a mobile platform, for which we show experiments in the next section, can be summarized as follows:

1. A light–field is reconstructed automatically based on the method described in Section 2. The light–field serves as environmental map of the scene.
2. In the scene a docking position is defined in world coordinates.
3. The moving platform is initialized arbitrarily in the scene, which means no a priori information about its position in world coordinates is provided. As a result $p(\boldsymbol{x}_0)$ is uniformly distributed.
4. The system then moves through the following steps until it reaches a high confidence of being at the final docking position:
   - compute the maximum of $p(\boldsymbol{x}_{t-1}|\mathcal{H}_{t-1})$ to get the estimated position $\boldsymbol{x}_{t-1}$
   - compute a small movement $\boldsymbol{m}_t$ based on the difference to the final docking position (translation in the x/y–plane and rotation around the z–axis)
   - take an image from the scene

Figure 4. The robot's scene environment. In the test sequence, the robot starts from the left and moves towards the left elevator button.

Figure 5. The autonomous moving platform with a mounted stereo head. We only use one of the cameras.

- update the estimate of the position as described in Section 3

In the next section we show an experiment which illustrates the capabilities of this approach by localizing a robot during its movement towards an elevator button.

5. EXPERIMENTAL RESULTS

In this section we describe experiments which use the idea of our approach but performs no active movement yet. Our scene for testing was a wall with three elevator doors as Figure 4 shows.

Before doing localization, an image based scene model must be generated, which is capable to render new, virtual scene views. For this task, we moved our robot (see Figure 5) in front of the scene and took 28 images with the robot's camera. In this example instead of applying our structure from motion approach, we used the robot's odometry to determine the camera movement, because this example is not enough textured to extract and track enough point features to get a robust motion estimation. The scene geometry has been approximated by a single plane, which has roughly been estimated to be similar to the wall.

To test the localization capability of our approach and to simulate the navigation task, we let the robot move towards the elevator button between the two left elevator doors. The movement enclosed translation and rotation within the plane of motion. During this movement we recorded 20 frames with the robot's camera. We used the information of the robot's odometry to get a rough estimate for the camera movement between each image pair.

In each of the following expermients, we initialized the position of 195 particles in a large area in front of the elevators randomly with arbitrary orientations in the range from 0 to 2π. In the left top image of Figure 6 a top view of the area in front of the elevators is shown with the positions of the particles after initialization, which are drawn as gray dots.

In a first experiment, we used the robot's odometry as the actions $\boldsymbol{m}_t$, which describe the relative changes of the position and heading. Figure 6 shows a comparison of the real camera view with the rendered views of the estimated position and the position which is given by odometry. The initial position of the odometry was adjusted manually such that it is very close to the true position of the robot.

As the estimation of the robot's position is modeled as a whole density, we have the problem to decide for a single pose estimation. For efficiency reasons we chose the parameters of that particle number i as estimation which has the maximum probability $p_{\boldsymbol{x}_{t,i}}$. This approximation in theory corresponds to a maximum likelihood estimation instead of to a maximum a posteriori estimation. Being aware of this, we found the position estimates always sufficiently accurate.

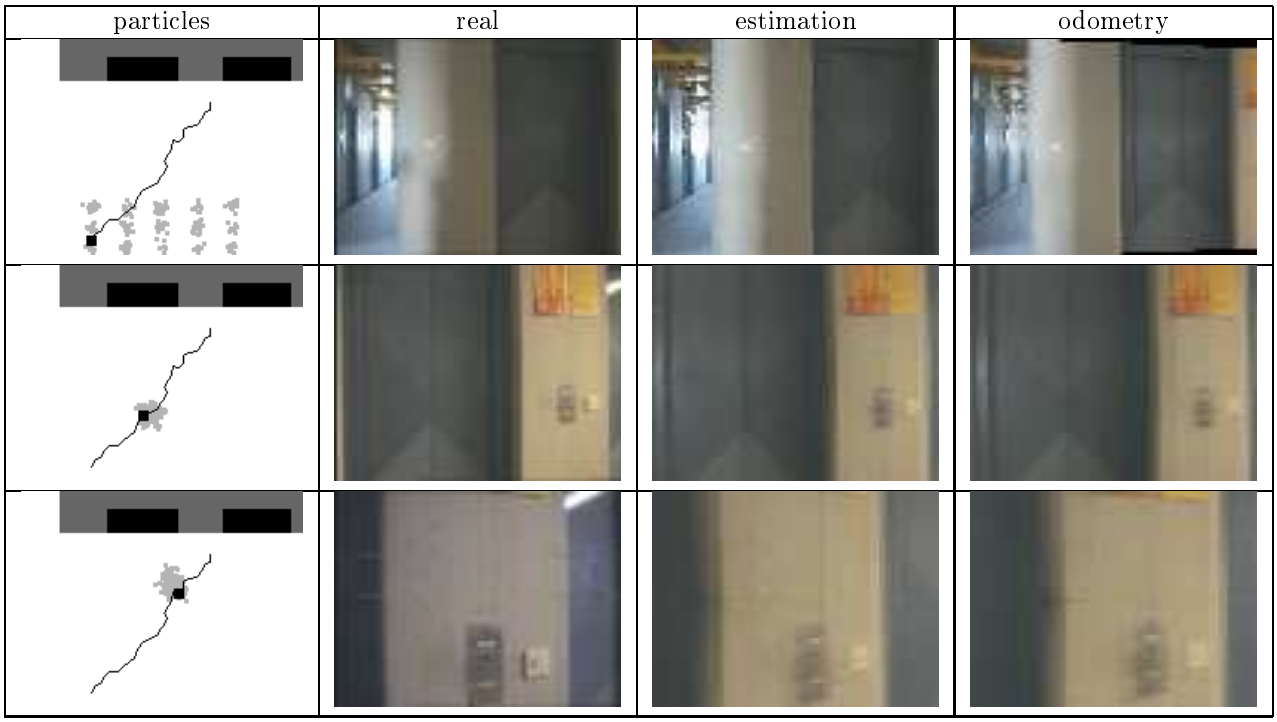

Figure 6. An experiment for testing the localization capability of our approach when using the robot's odometry. The left images show a schematic top view of the area in front of the wall (gray area) with the elevator doors (black rectangles). The positions of the particles are drawn as gray dots, the moving path as given by odometry is drawn as dark line, the actual position of the robot is drawn as a small black square. The right three images show the comparison of real camera views (left) with rendered images at the estimated most likely position (middle) and at the position which is given by odometry (right). From top to bottom, the situation at the 1st, 10th, and 20th frame of the test sequence is shown.

Figure 7. An experiment similar to the one shown in Figure 6 except that the used information about the robot's movement is a perturbation of the odometry. This path is drawn as a gray line. The initialization step was the same as in Figure 6, the two rows here show the situation at the 10th and the 20th frame of the test sequence.

In this experiment, the rendered view for the position which is given by the odometry is very similar to the real camera view. Therefore, the estimation of the true position is just a small adjustment in comparison to the real camera view. But the initial position of the robot has been determined totally automatically and very accurate. The accuracy can be verfied by comparing the rendered images related to the estimation with the real camera image, which are both shown in the first row of Figure 6.

To test the capability of our approach for the case when the odometry is erroneous, we made a second experiment by perturbing the odometry information and using this perturbed information as actions $\boldsymbol{m}_t$. Figure 7 shows the localization results for this experiment in the same wise as Figure 6. The initialization step there was the same as in Figure reff:ergreal and therefore is omitted.

It can be seen, that the particles are ordered not as compactly as in the first experiment, so because of the erroneous odometry, an additional uncertainty has been introduced. But nevertheless, the estimation of the robot's pose is comparable to the first experiment as can be verfied by comparing the rendered images for the estimation. Notice that in this example the used odometry was so erroneous that in the 20th frame, the elevator button even was not visible any more.

The computation time using software rendering was 0.14 seconds for each particle. A coarse estimation of the computation time using hardware accelerated rendering with off–the–shelf graphics accelerator boards, which has not been installed in our mobile platform, is approximately 0.0088 seconds per particle. Thus, in our experiments, every 1.73 seconds we get an update of the position estimate which seems to be fast enough to localize a robot.

6. SUMMARY

The idea of combining the photo–realistic and very efficient visualization capabilities of computer graphics with probabilistic self–localization approaches known from computer vision enables global robot localization in arbitrary natural environments. The complete framework, starting from lightfield reconstruction and ending with robot navigation, could only be briefly described in this paper. Due to lack of space, we could not go into detail describing

- the structure from motion approach we usually use for lightfield reconstruction. This is described elsewhere.[1,2] In our experiments we have used a simpified method for reconstructing the lighfield, for reasons, we have described in the experimental section.
- some general problems of the particle filter approach, like the influence of the number of particles on the state estimation results.
- how we decide for a movement while having an a posteriori distribution over the position of the mobile system. This makes in general path planing mechanisms necessary, which is beyond the scope of the paper.

Nevertheless we claim, that image based scene models are a promissing alternative to classical geometric based models. More than that, image based scene models in combination with efficient hardware accelerated rendering are the more general case of a scene modeling, because the appearance of the scene is modeled. The experiments have shown, that even with a simple measure for comparing two images (in our case, the pixel wise difference between two color images) probabilistic state estimation algorithms, like particle filters, return a higly accurate position estimate in reasonable time.

The benefits of our approach lie in the modeling technique, which allows to handle scenes or reflectance properties, that are difficult to model with pure geometric approaches. Additionally, the model can be constructed fully automatically. One of the main probleme of image based scene models is the amount of image data which must be stored. Although compression techniques for lightfields exits[14] a smart way must be found to devide the world of the robot into small areas, each area being represented by a smaller lightfield. In a coarse localization step, for one of these areas must be decided.

In our future work we will concentrate on showing localization and also navigation in a more complex environment. Also more experiments with quantitative evaluations will be performed. Finally we will prove in future experiments, that our scene model allows such an accurate position estimation, that even navigation in narrow spaces is possible, for example passing through a door.

ACKNOWLEDGMENTS

This work is partially funded by the German Research Foundation (DFG) under grant SFB603/TPC2 and the research grant DE 735/1-1. Only the authors are responsible for the content. We also like to thank Reinhard Koch, University of Kiel, for the helpful discussions on the efficient lightfield representation and rendering.

REFERENCES

1. R. Koch, M. Pollefeys, B. Heigl, L. V. Gool, and H. Niemann, "Calibration of hand-held camera sequences for plenoptic modeling," in *Proceedings of the 7^{th} International Conference on Computer Vision (ICCV)*, pp. 585–591, IEEE Computer Society Press, (Corfu), September 1998.
2. B. Heigl and H. Niemann, "Camera calibration from extended image sequences for lightfield reconstruction," in *Workshop Vision, Modeling and Visualization*, pp. 43–50, (Erlangen, Germany), Nov. 1999.
3. D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," tech. rep., Carnegie Mellon University, 1998.
4. F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.
5. M. Levoy and P. Hanrahan, "Lightfield rendering," in Rushmeier,[15] pp. 31–42.
6. S. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in Rushmeier,[15] pp. 43–54.
7. B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. V. Gool, "Plenoptic modeling and rendering from image sequences taken by a hand–held camera," in *Mustererkennung 1999*, W. Förstner, J. Buhmann, A. Faber, and P. Faber, eds., pp. 94–101, Springer, (Heidelberg), September 1999.
8. M. Isard and A. Blake, "A smoothing filter for condensation," in *Computer Vision - ECCV 98*, H. Burkhardt and B. Neumann, eds., pp. 767–781, 1998.
9. Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press, Boston, San Diego, New York, 1988.
10. R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering* , pp. 35–44, 1960.
11. A. Gelb, ed., *Applied Optimal Estimation*, The MIT Press, Cambridge, Massachusetts, 1979.
12. M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," in *Computer Vision - ECCV 96*, A. Blake, ed., pp. 343–356, (Berlin, Heidelberg, New York, London), 1996. Lecture Notes in Computer Science.
13. M. Tanner, *Tools for Statistical Inference*, Springer Verlag, London, Berlin, Heidelberg, New York, Paris, Tokyo, Hong Kong Budapest, 1993.
14. M. Magnor and B. Girod, "Hierarchical coding of light fields with disparity maps," in *Proc. International Conference on Image Processing (ICIP-99),* Kobe, Japan, pp. 334–338, IEEE Signal Processing Society, October 1999.
15. H. Rushmeier, ed., *Computer Graphics (Proceedings of SIGGRAPH 96)*, (New Orleans, Louisiana), Addison Wesley, Aug. 1996.