Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, Vancouver, Canada. DOI: 10.1109/IROS.2017.8206067

Sampling-based Methods for Visual Navigation in 3D Maps by Synthesizing Depth Images

Peer Neubert, Stefan Schubert and Peter Protzel

***Abstract*— Camera-based navigation within a given three-dimensional map enables heterogeneous robotic systems to share maps and use more abstract environment models like floor plans. This paper builds upon our previous work, and addresses the problem of how to combine 3D distance information from the map and the current visual image from the robot's camera in order to navigate within this map. The underlying assumption is that features which cause depth changes are also likely to create visual gradients. Based on this assumption, the similarity of visual image and depth images that are synthesized from the 3D map can be used to evaluate pose hypothesis. This paper integrates this idea into a Monte Carlo localization approach and additionally presents its application to path following. The presented approach is evaluated on a synthetic datasets that provides perfect knowledge of the ground truth, as well as two real-world datasets acquired by a heterogeneous robotic team: a proof-of-concept dataset in a scattered indoor environment, and a challenging corridor dataset.**

I. INTRODUCTION

Usually, robots are equipped with a broad range of different sensors and exploit different kinds of a priori known or externally provided information. Combining information from different sources (e.g., by means of sensor data fusion) is a fundamental building block in applied robotics. Sharing information across different sensor modalities (e.g., between a given 3D map and the current view from the robot's camera) is a particularly challenging task. In this paper, we deal with the problem of pure 2D vision based mobile robot navigation in a priori given 3D maps. We demonstrate experiments on a particularly interesting use case of heterogeneous robot teams, where one larger robot has powerfully 3D sensors to create a map and a second robot uses a cheaper and more lightweight camera to navigate within this map. Also, a single robot that captured 3D information at very low rates (e.g., a robot that has to stop to create scans [1]) can benefit from a vision based localization relative to the previous 3D frame. 3D structure is less prone to appearance changes due to weather or lighting conditions. Moreover, such 3D maps could be provided not only by other robots, but also from external sources like construction plans. We do not aim to replace approaches that also exploit additional information provided by the maps (e.g., from texture mapped 3D models), but we want to demonstrate that the sole geometric information can be sufficient for localization.

The underlying algorithmic idea borrows some ideas from a biologically inspired algorithm for visual homing of dessert ants [2]. In contrast to extracting and matching discrete

The authors are with Faculty of Electrical Engineering and Information Technology, Technische Universität Chemnitz, Germany `firstname.lastname@etit.tu-chemnitz.de`

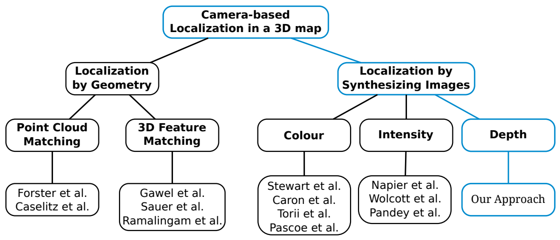

Fig. 1. Taxonomy of camera-based localization approaches in 3D maps. This work fills a gap by using synthesized depth images from the map for comparison to visual images.

landmarks from the different sensor modalities, we examine the question: Would the world look like this image if I were at that pose? While the ant algorithm warps visual images, we build upon synthesized depth images from the known 3D map and integrate them into Monte Carlo based localization.

In earlier work [3], we discuss an image processing pipeline that is able to provide reasonable similarity measures for input pairs of visual and depth images and sketch opportunities for the application on mobile robot navigation tasks. In this current paper, we integrate this earlier work in a Monte Carlo Localization system and present an approach for trajectory following based on this idea. We use a caching system to keep the computational effort for synthesizing depth images feasible and exploit properties of the image similarity measure to reduce the number of required particles. Finally, we present results on a synthesized dataset and real world experiments on a heterogeneous robot team.

II. RELATED WORK

The idea of using a visual camera to localize a robot in a known map of the 3D world is quite old. "Shakey the Robot" [4] enhanced edges in the visual image to obtain landmarks (e.g., doorways or wall corners) and compared them with the expected view from its currently assumed position. This was used to correct small errors in Shakey's pose estimate.

Fig. 1 provides a coarse taxonomy of more recent approaches to camera based localization in 3D maps. There are two basic groups: Members of the first group perform a localization with the geometrical information from both the map and the camera. Forster et al. [5] used a structure-from-motion based approach to conduct a camera-based *dense* reconstruction of the environment's geometrical structure. The resulting reconstruction is then used to perform an ICP-based (Iterative Closest Point) point cloud matching to obtain the current pose of the camera. Caselitz et al. [6], [7] showed that a vision-based *semi-dense* reconstruction, which is obtained from the Visual SLAM algorithm ORB-SLAM [8], is also sufficient for a subsequent ICP-based pose estimation. Gawel et al. [9] built also upon a semi-dense reconstruction by ORB-SLAM, however, for localization of the camera within the map they used the geometrical information from both the camera and the map to conduct a 3D feature matching. [10] and [11] extract skyline representation from images to localize within digital elevation models of mountain regions and coarse 3D city models respectively.

The second approach to a camera-based localization within a given 3D map is to use the map in order to synthesize images at arbitrary positions, allowing the usage of image processing based pose estimate algorithms. Given the raw depth information of the 3D map, most existing approaches require additional information in order to be able to perform a localization. Therefore, the approaches of Stewart et al. [12], Caron et al. [13], [14] and Pascoe et al. [15], [16] build upon a map-based RGB-image synthesis with a subsequent image matching approach for pose estimate. Napier et al. [17], Wolcott et al. [18], and Pandey et al. [19] showed that a previously given map enhanced with intensity data is sufficient to conduct a camera-based pose estimate. The presented approach in [17] is partially similar to our approach but requires additional laser intensity data. As a brute force search, they start with an extensive sampling of synthetic images which are generate from the intensity values of the map around an initial pose guess. On these synthetic images, an edge detection with a subsequent patch normalization is applied. Finally, the synthetic images are matched to the current camera view by a mutual information maximization.

We fill the remaining gap in Fig. 1 by solely using depth information obtained from the 3D maps.

III. ALGORITHMIC APPROACH

Comparing features between different sensor modalities (in our case a 3D map and panoramic visual images) is a challenging task. Our algorithmic approach builds upon the idea of evaluating synthesized depth images at hypothetical robot poses. Instead of computing features that can be matched between both modalities (like Shakey's doorways), we ask the question: How would the panoramic depth image look like if the robot actually *were* at a hypothetical position in the world? We evaluate the likelihood of this hypothetical position based on the assumption that parts of the world that cause depth changes are also likely to create visual image gradients.

How we create the synthesized images and compare them to visual images is content of the following two sections III-A and III-B. The subsequent sections describe how this integrates into Monte Carlo Localization and how it can be used to follow a given target trajectory.

A. Synthesizing depth images

Given a 3D point cloud map and an assumed position of the robot and its camera, we can synthesize a depth image by projecting the distance to the closest 3D points on the virtual camera plane (for details see [3]). This is a typical computer graphics task for which efficient methods exist.

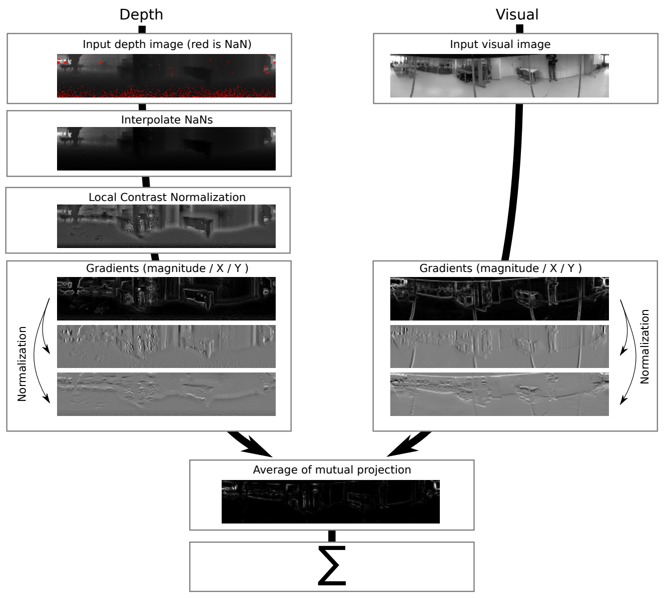
Depth
Visual
Input depth image (red is NaN)
Input visual image
Interpolate NaNs
Local Contrast Normalization
Gradients (magnitude / X / Y)
Gradients (magnitude / X / Y)
Normalization
Normalization
Average of mutual projection
$\sum$

Fig. 2. Overview of the image processing pipeline. The red pixels in the input depth image are NaN values. See text for details.

The usage of normalized spherical coordinates for all images facilitates efficient implementation (i.e., the camera “plane“ is the unit sphere). This allows to directly use a part of the spherical image to select a particular field of view, e.g., a panoramic image at a certain orientation of the robot. Parts of the synthesized image for which the map does not provide information are filled with NaN values.

B. Comparing visual and depth images

The image similarity computation is based on a holistic comparison of the visual and the depth images. Fig. 2 summarized the processing steps. A more detailed explanation and evaluation can be found in [3]. We compute gradient based features independently for both images and add up the average lengths of the mutual projections. Careful normalization is essential for this approach. The visual images are normalized to a total magnitude sum equal one in each image. This lowers the importance of individual pixel gradients if there is a lot of texture in the image.

Additionally to this global normalization, a local contrast normalization (LCN) is applied to the depth images. The LCN computes mean m and standard deviation σ in a $[(2k+1)\times(2k+1)]$ neighbourhood around each pixel $D(i,j)$ and scales and shifts the value of this pixel by:

$$D(i,j) \leftarrow \frac{D(i,j)-m}{\sigma} \tag{1}$$

Since m and σ are computed for each pixel from its individual neighbourhood and this LCN is applied to all synthesized depth images, an efficient implementation is crucial. An outline of our algorithm based on integral images is provided in Algorithm 1. Its runtime is linear in the number of pixels and independent of the radius k of the contrast normalization. Although this algorithm can handle NaN values, we interpolate NaN values from surrounding non-NaN values before LCN (this can be done efficiently based on integral images as well).

Algorithm 1: Efficient LCN using integral images

```
Data: Image $I$, range of LCN $k$ in pixels
Result: Local contrast normalized image $N$
   // Apply LCN on I
1  Function N = LCN(I, k)
      // prepare integral images
2     $iiVals \leftarrow$ computeIntegralImage(I)
3     $iiSquaredVals \leftarrow$ computeIntegralImage($I^2$)
4     $iiNoNaN \leftarrow$ computeIntegralImage(not(isnan(I)))
5     $N \leftarrow$ NaN($h_I, w_I$)
6     foreach Not NaN pixel $(y, x)$ in $I$ do
7         $[x_1, x, y_1, y_2] \leftarrow$ getAreaCornerCoordinates($x, y, k$)
8         $n \leftarrow$ queryIntegralImage($iiNoNaN, y_1, y_2, x_1, x_2$)
9         $m \leftarrow$ queryIntegralImage($iiVals, y_1, y_2, x_1, x_2$)
10        $m_2 \leftarrow$ queryIntegralImage($iiSquaredVals, y_1, y_2, x_1, x_2$)
          // Apply Steiner translation theorem
11        $\sigma \leftarrow \sqrt{m_2/n - (m/n)^2}$
12        $N(y, x) \leftarrow \frac{I(y,x) - \frac{m}{n}}{\sigma}$
13    end
   // Value in J is sum of all top left values in I
   // NaNs are counted as zeros
14 Function J = computeIntegralImage(I)
15    $J \leftarrow$zeros($h_I + 1, w_I + 1$)
16    foreach Pixel $(y, x)$ in $I$ do
17        $J(y + 1, x + 1) \leftarrow \sum_{i<y,j<x} I(i, j)$
18    end
   // Get sum of rectangular area
19 Function v = queryIntegralImage($J, y_1, y_2, x_1, x_2$ )
20 $v \leftarrow J(y_2 + 1, x_2 + 1) - J(y_2 + 1, x_1) - J(y_1, x_2 + 1) + J(y_1, x_1)$
```

The gradient features G_v (visual) and G_d (depth) are computed using vertical and horizontal Sobel filters. To obtain the similarity s between visual and depth images, we evaluate the mutual projections of these gradients by

$$s = \sum_{pixels} \left\| \frac{\overrightarrow{G}_v \cdot \overrightarrow{G}_d}{\|\overrightarrow{G}_d\|^2} \cdot \overrightarrow{G}_d \right\| + \sum_{pixels} \left\| \frac{\overrightarrow{G}_d \cdot \overrightarrow{G}_v}{\|\overrightarrow{G}_v\|^2} \cdot \overrightarrow{G}_v \right\| \quad (2)$$

C. Monte Carlo Localization based on synthesizing depth images

For practical application of the above similarity measure for mobile robot navigation, it is crucial to take care of the number of synthesized depth images and image comparisons. The described evaluation of visual and synthesized depth images seamlessly integrates into particle filter based Monte Carlo Localization (MCL). Each particle represents a hypothesis of the current robot pose. The weight of each particle is computed from the accordance of the synthesized depth image at this pose with the current camera image.

The nature of our similarity evaluation approach allows some modifications of vanilla MCL to keep the computations feasible. Algorithm 2 outlines the computational steps. We exploit two paths to save computations: We extensively reuse data, and we reduce the number of required particles.

As already mentioned, the number of synthesized images has large influence on the runtime. Normalized spherical coordinates allow to use a *single panoramic depth image* for the evaluation of *all orientations* at this map position.

Algorithm 2: Visual-Depth Monte Carlo Localization

```
Data: 3D Map $M$, image sequence $I_{1:n}$, odometry measures $U_{1:n}$
Result: Particle set $P$, each $p_j \in P$ is a robot pose
   // initialize particle set
1  $P \leftarrow$initParticles()
2  $D_{cache} \leftarrow$ prepareDepthImageCache(M)
   // process each image
3  for $t = 1 : n$ do
4      $S_{cache} \leftarrow$ prepareSimilarityCache(M)
5      foreach $p_j \in P$ do
           // sample motion from odometry
6          $p_j^u \leftarrow$ applyOdometrie($p_j, U_t$)
           // query cache for existing similarity
              evaluation S of this pose
7          if isInCache($S_{cache}, p_j^u$) then
8              $S \leftarrow$ queryCache($S_{cache}, p_j^u$)
9          else
               // query cache for existing
                  synthesized depth image
10             if isInCache($D_{cache}, p_j^u$) then
11                 $D \leftarrow$ queryCache($D_{cache}, p_j^u$)
12             else
13                 $D \leftarrow$ synthesizeDepthImage($M, p_j^u$)
14                 $D_{cache} \leftarrow$insertInCache($D_{cache}, p_j^u, D$)
15             end
               // compare images
16             $S \leftarrow$ compareVisualDepth($I_t, D$)
17             $S_{cache} \leftarrow$insertInCache($S_{cache}, p_j^u, S$)
18         end
           // adjust particle orientation from S
19         $p_j^u \leftarrow$ adjustOrientation($p_j^u, S$)
           // obtain sampling weight from S
20         $w_j \leftarrow$ getWeight($S$)
21     end
       // importance sampling
22     $P \leftarrow$ importanceSampling($w, P$)
23 end
```

Line 16 calls a function that computes the visual depth image similarity from equation 2 for a set of orientation differences α between the visual and the depth image.

In normalized spherical image coordinates and under planar motion (where α resembles the yaw angle) this can be simply done by circular shifts of image columns. The result S of line 16 hold the similarity for all α-shifts. This function can directly be used to implement a visual compass for a robot that is equipped with a panoramic camera and navigates in a known 3D map. The structure of the used image processing pipeline depicted in Fig. 2 enables to compute the gradient image features once for each panoramic depth and visual image and only recomputes the mutual projections for each α-shift individually.

To further speed up processing, we implement two data caches: Line 2 instantiates a memory for synthesized depth images. All synthesized depth images are stored in this cache. They are reused whenever a new depth image is queried for a pose for which a nearby depth image is available in the cache (line 10). Since we synthesize panoramic depth images, the cache query is orientation independent. This cache saves computation time, but comes at the cost of a new parameter: the maximum distance of poses for which a cached image is used. This parameter influences the resolution and accuracy

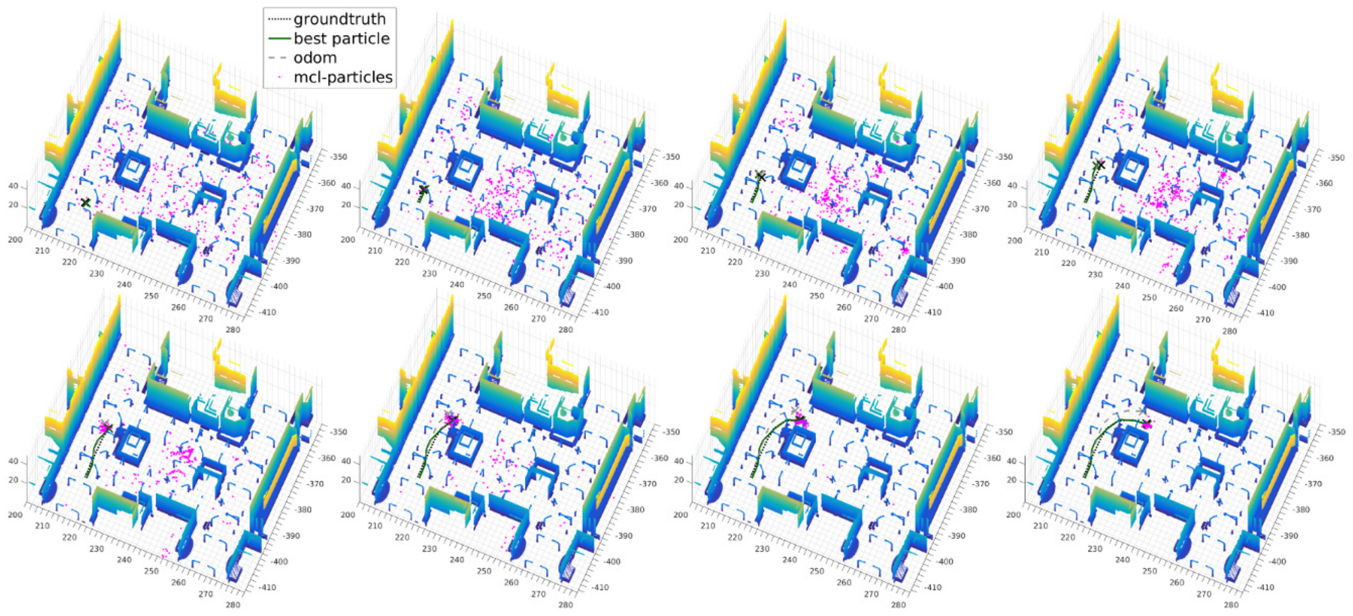

Fig. 3. Illustration of the particles during first iterations (row major order) of the Monte Carlo Localization Algorithm 2 (after importance sampling). Initially, the magenta coloured particles are uniformly spread over the map. In each iteration, the odometry measures and the similarity of synthesized depth images to the current visual image are evaluated. After few iterations, the particles converge to the ground truth trajectory (black dotted line).

of the resulting localization. Fortunately, the choice of its value is facilitated by its clear geometrical meaning. This cache has the advantage that, dependent on the task, it might be filled offline *in advance*. For large maps, the cache can be stored on disk, and parts near the current set of particles can be loaded into memory.

The second cache (created in line 4) depends on the current robot image and cannot be precomputed. It contains the results of similarity evaluations between visual and depth images (line 17). These comparison results can be reused between different particles at very similar positions, but with potentially different orientations. Since we expect a high density of particles at some parts of the map, the resulting reuse rate can be quite high.

The number of required particles depends on the number of degrees of freedom that have to be represented by their distribution. We can exploit the structure of the image processing pipeline to reduce the number of particles that are required to represent the variance of the orientation of the particles. The time for image synthesis and gradient feature computation (including LCN) is much higher than the time for evaluation of the mutual projections. When comparing a depth and visual image, it is computationally cheap to evaluate the whole set of α-rotations. We can then use this information to adjust the orientation of the particles (line 19) before computing the weight. This adjustment can be done deterministically by assigning the best orientation or by fusion with the particle's orientation estimate. This adjustment can also vary over time. We will see in the evaluation that in particular for the initial set of particles in a kidnapped robot scenario, an initial adjustment can reduce the number of required particles.

For computing the weights in line 20 it showed to be beneficial to spread the resulting similarities for equa-

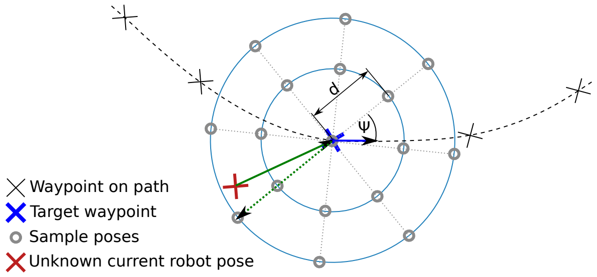

Fig. 4. Motion direction estimation for path following: The true motion direction (solid green arrow) towards a target waypoint is approximated based on the direction (dotted green arrow) from the target to the most similar synthesized depth image sampled at angles Ψ and distances d around the target waypoint.

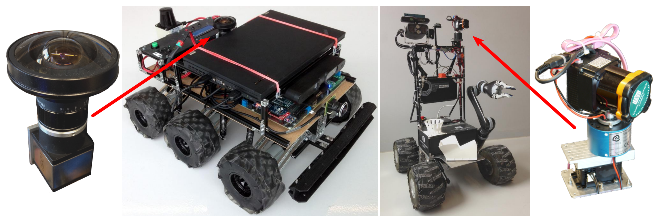

Fig. 5. The fisheye camera on the left is mounted on the small robot (about 7 kg) in the mid-left. The larger robot (> 50 kg) on the mid-right created the maps using the 3D laserscanner shown on the very right.

tion 2. They are normalized to range $[0, 1]$ and spread by $w = s^{\lambda}_{normalized}$. The results are robust to a wide range of choices for λ, in our experiments we choose $\lambda = 20$. For importance sampling in line 22, we use the low variance sampler of [20]. For practical applications, Algorithm 2 should be combined with state-of-the-art MCL extensions to control the particle distribution properties like adaptive resampling rates and particle spreading [20].

D. Application for path following

The described similarity measurement can also be used to follow a trajectory that is defined in the map. As before,

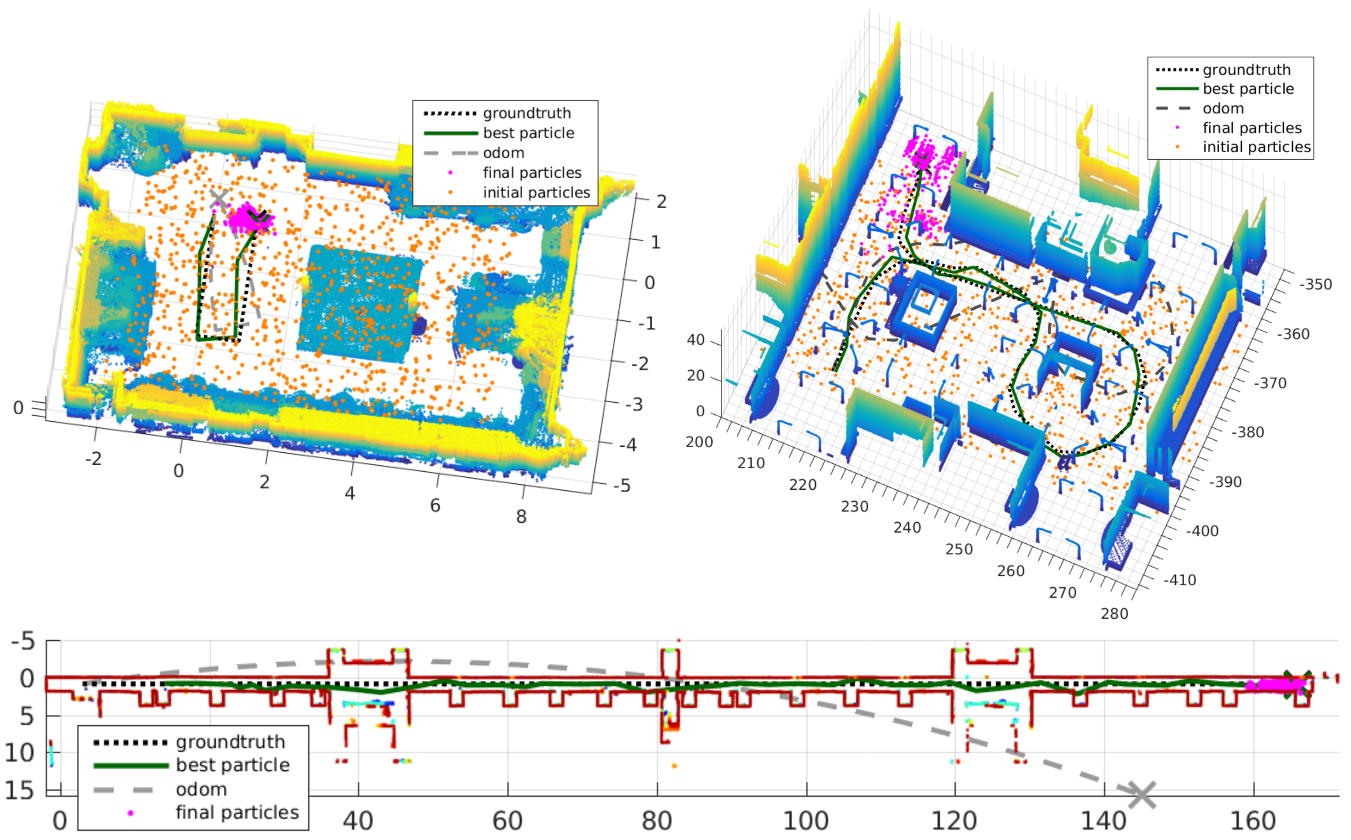

Fig. 6. Localization results on the three datasets. For the Lab and ETHZ datasets, the initial particles were spread uniformly over the whole image. For the Corridor dataset, the initial particles were spread over the first 40 metres of the corridor. The shown resulting trajectory is obtained from the particle with the highest accumulated weight.

we assume a robot with an omnidirectional camera and a 3D point cloud (again there is no need for colour or laser intensity information in the map). Further, there is a trajectory (a sequence of way points) defined in the map. This trajectory can be the outcome of a path planner, the trajectory of another robot or of a prior pass of the same robot through this environment (in a Teach & Repeat sense).

Fig. 4 illustrates our approach to estimate the required motion direction to a selected waypoint of the trajectory. The robot pose is unknown, but it is known to be in the vicinity of a certain waypoint (e.g., the start point of a planed trajectory). For a set of motion distances d and motion directions Ψ, we can synthesize depth images and compare them to the current visual image using the same approach that evaluates the particles in line 16 of Algorithm 2. Again, the similarity is evaluated for all orientation differences α. From d and Ψ of the best sample position, a motion direction towards the target waypoint can be estimated. In our experiments, we sample at 16 directions Ψ and two distances d and additionally at the waypoint itself. To increase robustness, the motion estimates can be smoothed by median filtering (in our experiments over the last 5 images). The gist of this approach is close to an biological inspired model of visual homing of the dessert ant [2]. Although we use synthesized depth images instead of warped visual views, we use the same nomenclature for ease of methodological comparison.

IV. EXPERIMENTAL RESULTS

We conduct experiments using the three datasets illustrated in Fig. 7: The *ETHZ* dataset [21] is a publicly available synthetic dataset that provides omnidirectional panoramic and depth images from a known trajectory through a simulated city environment. We do not use the provided depth images directly (since they are all from a single trajectory), but reconstruct a 3D model of the inner part of the city from which we can synthesize images. We use the sequence part 300-1200 of the original visual panoramic images (the part through the inner city) and localize each 25 images. For path following, we used each 10th image. Odometry measurements were generated by adding Gaussian noise to the provided ground truth. The other two datasets were collected with the robots shown in Fig. 5. The large robot has a high resolution 3D laserscanner that was used in combination with ICP for mapping. The small robot has a upwards looking fisheye camera with about 190^o FOV and captured the panoramic images during a second run through the environment. The camera was calibrated using [22] and provided panoramic images in spherical coordinates. This heterogeneous robot team collected data from a cluttered laboratory environment (8 images), the *Lab* dataset, and a challenging 160m *Corridor* (41 images) with significant repetition in the 3D structure. All maps contain only 3D information (neither colour, nor laser intensities). Table I pro-

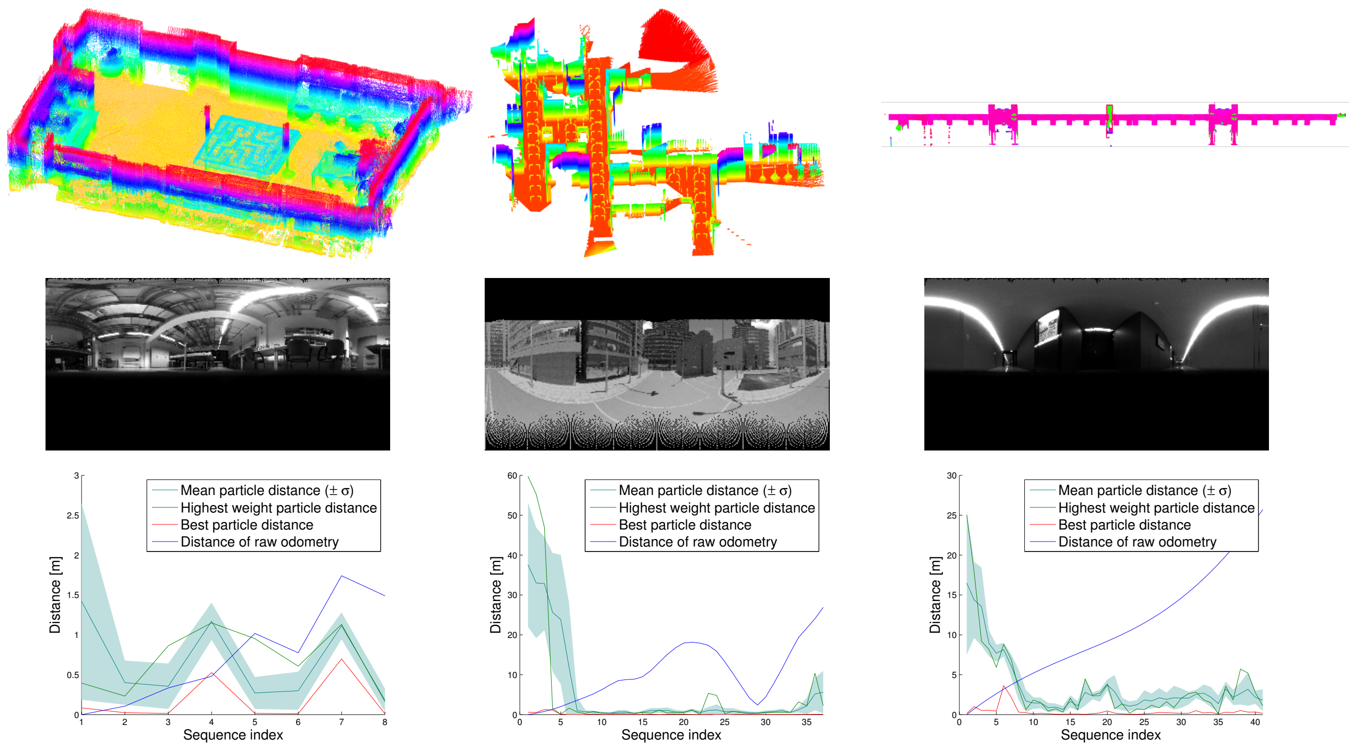

Fig. 7. Illustration of the datasets and quantitative evaluation for the (f.l.t.r.) Lab, ETHZ and Corridor datasets. The ETHZ 3D map was created from the provided depth images and the ground truth trajectory. The maps for the Lab and Corridor datasets were collected with the laser scanner of the larger robot in Fig. 5 and a manually corrected ICP. The panoramic images of the Lab and Corridor datasets were captured with the fisheye camera of the smaller robot from this figure. The vertical field of view of the panoramic image was further reduced, i.e., the salient ceiling construction of the Lab environment was not in the images.

vides the parameter settings for the following experiments.

In [3] we already evaluated properties of the image processing pipeline and the heading direction estimation. Here, we want to focus on the navigation application. Fig. 3 illustrates the first iterations of MCL Algorithm 2 on the ETHZ dataset. Since the initial pose is unknown, we spread 1000 particles uniformly over the considered city area. Based on the evaluation of synthesized depth images and odometry (raw odometry is shown in dashed grey, ground truth in dotted black), the particle set converges to the ground truth position.

The resulting trajectory of the single particle with the highest accumulated weight for the whole sequence of this dataset and the corresponding experiments on the other datasets can be seen in Fig. 6. On all datasets, the comparison of the visual images to synthesized depth images can significantly contribute to the localization. Without adjusting the particle orientation based on the best α-orientation (line 19 in Algorithm 2), the number of particles has to be increased to about 4000 for the ETHZ dataset to resemble similar results in this experiment.

TABLE I

Parameter settings

Parameter	ETHZ	Lab	Corridor
# particles	1000	same	same
λ	20	same	same
α	each 10^o	same	same
Max. Image distance for cache	0.5 m	0.25 m	same
Vertical FOV	85^o	51^o	same
Ψ	each 22.5^o		
d	$\{0m, 1m, 2m\}$		

Fig. 7 provides a quantitative evaluation of the above experiments. The mean distance of the particles to the ground truth pose quickly decreases after random initialization. The increase of the mean particle distance and variance at the end of the ETHZ trajectory is due to a slight slope of the ground. This causes the z-coordinate of the camera to increase, but is not regarded during depth image synthesis. Although the distance of the particle with the actually highest weight (green curve) can considerably differ from the ground truth pose, in almost all cases, there is a particle very close to the ground truth pose (red curve). The trajectories in Fig. 6 illustrate that reasonable path estimates can be obtained from particles with high accumulated weights over several MCL iterations.

Fig. 8 illustrates a proof-of-concept experiment for the application on path following. The robot moves along the red trajectory and successfully estimates the motion direction towards a nearby green target trajectory based on synthesized depth images as described in section III-D.

V. DISCUSSION

We described how synthesized depth images can be used in combination with a Monte Carlo approach to localize a robot that is equipped with a panoramic visual camera. Solely 3D information is used from the map, neither colour nor laser intensity information are required. Dependent on their availability (i.e., the source of the 3D map) the combination with such additional information is, of course, interesting.

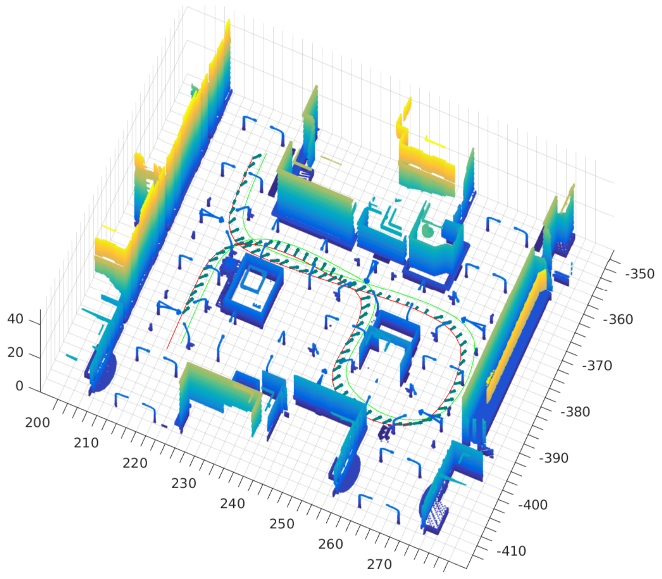

Fig. 8. Example path following result: The robot drives on the red path and estimates at each position the direction towards the green target trajectory. For visualization purposes we do no execute the motion to keep some distance between the trajectories.

Although we used two mobile robots for data collection, all experiments were run offline in Matlab. Based on the approaches for speed-up described in the algorithmic description, we do not see a principle hurdle for online application on a robot. However, a detailed evaluation of the overall runtime and the benefit of the caching stages remains open.

We observed a clear limitation regarding visual aliasing on the corridor dataset: At one end of the corridor there is a particular region for which synthesized images showed to be very similar to images captured at many places along the corridor. Particles that are initialized in this region tend to suppress hypothesis closer to the ground truth. One has to keep in mind, that the image processing pipeline and its parameters were not adapted for these datasets (they were adjusted on a set of RGB-D images in [3]). In particular, replacing the hand designed image comparison pipeline with learned features based on Convolutional Neural Networks is a promising direction for future work. More sophisticated handling of areas in the depth image that could not be synthesized from the map is also expected to be beneficial (e.g., the current interpolation approach is likely to omit some information from the skyline in an outdoor environment).

Our approach requires a known 3D map but poses only low requirements on these maps. For instance, in our robot team, the small robot could also follow the path of the large robot just based on its individual laser scans (i.e., the small robot localizes relative to a single scan). This avoids the task of building a globally consistent 3D map.

Currently, we assume planar motion and an orientation of the camera perpendicular to the ground plane. For application on uneven terrain, the normalized spherical coordinates allow to efficiently create appropriate depth images from full synthesized spherical images based on an orientation estimated, e.g., by an inertial measurement unit (IMU). The same approach could be used to integrate standard FOV visual cameras instead of panoramic cameras.

References

[1] S. Lange, D. Wunschel, S. Schubert, T. Pfeifer, P. Weissig, A. Uhlig, M. Truschzinski, and P. Protzel, "Two autonomous robots for the dlr spacebot cup - lessons learned from 60 minutes on the moon," in *International Symposium on Robotics*, 2016.

[2] R. Möller, "Local visual homing by warping of two-dimensional images," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 87 – 101, 2009.

[3] S. Schubert, P. Neubert, and P. Protzel, "Towards camera based navigation in 3d maps by synthesizing depth images," in *Proc. of Towards Autonomous Robotic Systems (TAROS)*, 2017.

[4] N. J. Nilsson, "Shakey the robot," AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, Tech. Rep. 323, Apr 1984.

[5] C. Forster, M. Pizzoli, and D. Scaramuzza, "Air-ground localization and map augmentation using monocular dense reconstruction," in *Int. Conf. on Intelligent Robots and Systems*, 2013.

[6] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard, "Monocular camera localization in 3d lidar maps," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 1926–1931.

[7] ——, "Matching geometry for long-term monocular camera localization," in *ICRA Workshop: AI for long-term Autonomy*, 2016. [Online]. Available: http://www2.informatik.uni-freiburg.de/ caselitz/downloads/caselitz16icraws.pdf

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Trans. on Robotics*, 2015.

[9] A. Gawel, T. Cieslewski, R. Dub, M. Bosse, R. Siegwart, and J. Nieto, "Structure-based vision-laser matching," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 182–188.

[10] O. Saurer, G. Baatz, K. Köser, L. Ladický, and M. Pollefeys, "Image based geo-localization in the alps," *Int. J. Comput. Vision*, vol. 116, no. 3, pp. 213–225, Feb. 2016.

[11] S. Ramalingam, S. Bouaziz, P. Sturm, and M. Brand, "Skyline2gps: Localization in urban canyons using omni-skylines," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 3816–3823.

[12] A. D. Stewart and P. Newman, "Laps - localisation using appearance of prior structure: 6-dof monocular camera localisation using prior pointclouds," in *IEEE International Conference on Robotics and Automation*, May 2012, pp. 2625–2632.

[13] G. Caron, A. Dame, and E. Marchand, "Direct model based visual tracking and pose estimation using mutual information," *Image and Vision Computing*, vol. 32, no. 1, pp. 54 – 63, 2014.

[14] A. Torii, R. Arandjelovi, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[15] G. Pascoe, W. Maddern, and P. Newman, "Direct visual localisation and calibration for road vehicles in changing city environments," in *Int. Conf. on Computer Vision Workshop (ICCVW)*, 2015, pp. 98–105.

[16] G. Pascoe, W. Maddern, A. D. Stewart, and P. Newman, "Farlap: Fast robust localisation using appearance priors," in *Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[17] A. Napier, P. Corke, and P. Newman, "Cross-calibration of push-broom 2d lidars and cameras in natural scenes," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 3679–3684.

[18] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *2014 IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, Sept 2014, pp. 176–183.

[19] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Jour. of Field Robotics*, vol. 32, no. 5, pp. 696–722, 2015.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[21] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," in *Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[22] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006.