StreetMap - Mapping and Localization on Ground Planes using a Downward Facing Camera

Xu Chen[1], Anurag Sai Vempati[1,2], and Paul Beardsley[1]

Abstract— **This paper describes a system to map a ground-plane, and to subsequently use the map for localization of a mobile robot. The robot has a downward-facing camera, and works on a variety of ground textures including general texture like tarmac, man-made designs like carpet, and rectilinear textures like indoor tiles or outdoor slabs. Such textures provide a basis for measuring relative motion (i.e. computer mouse functionality). But the goal here is the more challenging one of absolute localization.**

The paper describes a complete working pipeline to build a globally consistent map of a given ground-plane and subsequently to localize within this map at real-time. Two algorithms are described. The first is a feature-based approach which is general to any ground plane texture. The second algorithm takes advantage of the extra constraints available for common rectilinear textures like indoor tiling, paving slabs, and laid brickwork. Quantitative and qualitative experimental results are shown for mapping and localization on a variety of ground-planes.

I. Introduction

Localization is a key functionality for mobile robots, and it is achieved in diverse ways. A well-established and powerful approach is to use SLAM or offline reconstruction to generate a map, which can be registered to a canonical coordinate frame if needed, and to subsequently localize against the map via 2D data (camera images) or 3D data (laser, depth sensor). Alternatively an external camera setup in the environment can track the object of interest. Common wireless technology for localization includes GPS and D-GPS for outdoor use, or Ultra Wideband (UWB) for spaces that have been instrumented with UWB anchor tags.

These approaches are challenged by our application in which a social robot is deployed amongst humans, indoors or outdoors. The crowd is dynamically changing and, during busy periods, it can be sufficiently dense to block lateral-facing sensors on the robot from viewing the fixed surroundings, or to block a view of the robot from an external camera setup. UWB localization also ceases to operate in dense crowds, while GPS is not suitable for our scenario because it does not work indoors.

This motivates the work in this paper, in which the robot has a downward-facing camera and localizes off imagery of the ground-plane. Our target workplaces are regular and even ground planes suitable for walking or wheeled platforms, indoors or outdoors, hence we assume the robot moves in a 2D world. The first step is offline, traversing the full deployment area, and creating a globally consistent 2D map of the ground-plane. Subsequently, an online system matches live camera imagery to the map, and thereby does localization. There are two cases for online localization - (a) the robot starts at an arbitrary position on the map and initializes its location automatically by place recognition, (b) the ground-plane texture is too ambiguous for place recognition, due to for example repeated carpet design or tiling. In this case, the starting location of the robot is supplied a priori - for example, the robot starts in a pre-defined location, or it has multiple sensor modalities, and other sensors like vision and wireless are at least periodically providing a location estimate as the crowd density changes.

The system is demonstrated on ground textures that cover a large variety of real-world scenarios. For indoor scenes - tiled floors and carpets. For outdoor scenes - paving slabs, laid brickwork, and tarmac. In fact, two separate algorithms are described. The first algorithm is the most general and is feature-based. The second algorithm has been developed specifically for rectilinear textures, and takes advantage of the constraints that are available when localizing on a tiled floor, paving slabs, or laid brickwork.

The **contributions** of the work are:

- A case study of an end-to-end pipeline for mapping a ground-plane and subsequently localizing within it, demonstrated on a variety of ground-plane textures.
- A demonstration of feature-based localization for the challenging case of tarmac, to demonstrate the power of the approach.
- A demonstration of the benefits of using constraints arising from rectilinear texture, for both the map generation and online localization stages.

In the remainder of the paper, Section II describes related work, Section III describes the feature-based approach, Section IV covers the line-based approach that utilizes constraints arising from rectilinear texture, and Section V gives experimental results.

II. Related Work

A. Monocular SLAM and Visual Odometry in a 3D Scene

In recent years, great advances have been witnessed in the area of visual SLAM and odometry. Two main divisions are appearance-based [1], [2] and feature-based [3]. Appearance-based methods obtain the relative transformation between frames by minimizing a photometric error while feature-based approaches first extract and match feature points in

[1]Xu Chen was at the time of this work with, and Anurag Sai Vempati and Paul Beardsley are with, Disney Research Zurich, Stampfenbachstrasse 48, 8006, Zurich, Switzerland. xuchen@student.ethz.ch

[2]Anurag Sai Vempati is with the Autonomous Systems Lab at ETH Zurich, Leonhardstrasse 21, 8092, Zurich, Switzerland.

the images and then minimize a re-projection error. For the task of re-localization, feature-based method is favored due to the efficiency of bag of words retrieval algorithm [4] and its robustness against changes in lighting and scene. Feature-based approaches can be further divided based on the type of correspondences. One way is to directly estimate the fundamental matrix or homography via 2D-2D correspondences. Another is to first triangulate 3D points and estimate the camera projection matrix via 2D-3D correspondences. In this work we deal specifically with planar ground surfaces as viewed by a downward facing camera. Owing to the inherent 2D nature of the scene, robust camera tracking can be achieved by using 2D-2D correspondences, and there is no need to work in 3D, incurring unnecessary computation plus uncertainty in the depth estimation.

B. Downward Facing Camera for Localization

To the best of our knowledge [5] is the first work proposing to use downward facing camera for localization. They use a prebuilt map mosaic and track the camera pose along the map based on visual appearance. Later works such as [6]–[8] and [9] use similar approaches for visual odometry based on feature correlation, template tracking or ICP. Our system differs from these previous works in its ability to build a globally consistent map and to localize globally within the map, which is important for ground robots' localization.

C. Line-based Localization

Grounds with salient linear patterns are a common occurrence in man-made environments. There have been several works that use lines for localization. [10] treat lines detected by a Line Segment Detector (LSD) [11] as local landmarks and use them for localization and mapping. However, according to our experiments, boundary lines in some cases lack distinctiveness which causes LSD to fail. In [12] the authors use lanes on the road to estimate camera pose. However, they only consider the simple case where lines are sparse and are always visible. Our method is shown to work on a variety of ground textures with more challenging linear patterns of varying densities.

III. Feature based System

The texture on some common ground surfaces, e.g. like the ones shown in Fig. 1, actually provide rich and distinct information which enables reliable image matching and retrieval. The feature based system exploits this information to first incrementally build a globally consistent map of the ground-plane during an exploration phase and then to localize within it at real-time.

The map for the feature based system is represented as a set of images together with their corresponding camera poses $\mathbf{C}$. As illustrated in Fig. 2, during the mapping process the global poses of individual images $\mathbf{C}$ are determined by accumulating relative transformation $\mathbf{T}$ from the local tracking thread and performing a global optimization when an area is revisited. Frame-to-frame homography is used to determine relative camera poses as the camera explores new areas. In parallel, a relocalization routine matches every incoming image with the images already existing in the map. The relative transformation between the matched pair is then estimated and is used together with the global pose of the matched image to determine the camera pose for new image.

With a prebuilt map, global initialization and relocalization can be performed by retrieving the most similar map frames to the current camera frame with bag of words algorithm [4] and estimating their relative transformation. For some artificial ground textures with repeating patterns, e.g. indoor carpet, a rough initial position needs to be manually provided to resolve the ambiguity for global initialization. Potentially, other sensors such as UWB could also provide such information. The global relocalization is performed every few frames to save computation, and the camera poses in-between are obtained by accumulating the relative motion between successive frames.

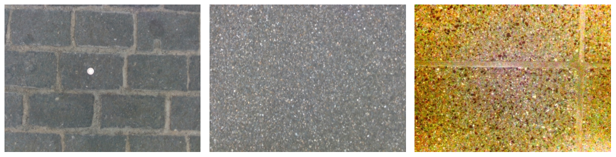

(a) Laid brickwork (b) Tarmac road surface (c) Indoor textured tiles

Fig. 1: Common ground textures that are handled by our system.

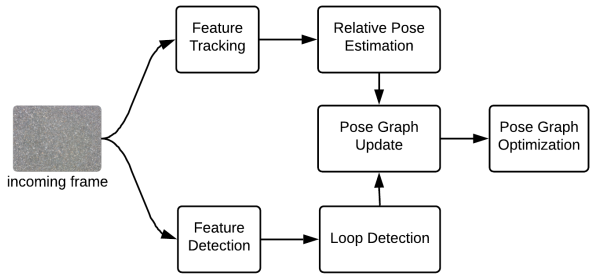


Fig. 2: Pipeline of the feature-based mapping process. For an incoming image, its relative transformation to the previous image is estimated with the tracked features. Loop closure is detected through feature matching. Relative transformations from the local tracking thread and the loop closure thread are fused using a pose graph.

A. Relative Transformation from Homography

Since the camera looks downward at the ground, we assume the only object in the scene is the planar ground, hence the relative transformation between images can be fully described by a homography.

To estimate the homography, SURF features [13] are used. For evaluating the transformation between successive images, feature correspondences are obtained through Lucas-Kanade tracking [14]. A homography is also used in the case of loop closure. SURF features are firstly detected and extracted, and then matched through nearest neighbor search in the descriptor space. First to second ratio test is used to further filter outliers. Once the correspondence is established, the homography between the current and retrieved image is estimated by minimizing the reprojection error in a RANSAC fashion. The homography can be represented as

$$\mathbf{H} = \mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{d} \tag{1}$$

where $\mathbf{R} = \mathbf{R_x}\mathbf{R_y}\mathbf{R_z}$ is a rotation matrix representing the relative camera rotations around the x, y, and z-axis respectively. $\mathbf{t} = [t_x, t_y, t_z]^T$ represents the relative translation, $\mathbf{n} = [n_x, n_y, n_z]^T$ is the plane normal and d is the distance between the image plane and the ground plane.. As shown in [15] for a given $\mathbf{H}$ there could be up to eight sets of $\{\mathbf{R}, \mathbf{t}, \mathbf{n}\}$ satisfying Equation (1) . While four pairs among them can be eliminated by assuming that two cameras always locate on the same side of the plane. The rest four possible solutions are induced by ambiguity in the normal direction of the plane. In our case, the normal direction of the ground plane is fixed, therefore a unique set of $\{\mathbf{R}, \mathbf{t}, \mathbf{n}\}$ can be obtained, hence the relative transformation $\mathbf{T} = [\mathbf{R}|\mathbf{t}]$ is solved.

B. Global Optimization and Loop Closure Detection

The drift is an inevitable problem in SLAM which becomes even more severe in the downward facing setup due to the small field of view and any non-flatness of the terrain. The small field of view limits the number of features per image which in-turn increases the sensitivity to the noise in the position of feature points. The non flatness of the terrain violates the assumption of homography, hence impacts the accuracy of relative motion estimation. Pose graph optimization [16] is a common and effective method to reduce drift. Camera poses $\mathbf{C}$ are represented as nodes in the graph and the relative transformations $\mathbf{T}$ between frames are edges.

To construct the graph, the transformations from the current image to its previous image and to its spatially overlapped images (on loop closure) are estimated. The spatially overlapping images are detected by retrieving the most similar images (in terms of number of feature correspondences) among all the previous images while eliminating the recently visited ones. The retrieved images are then verified by geometric check, and given that they pass the check, the transformations from the current image to each of them are evaluated and added to the graph.

Every time a loop closure is detected, all the existing poses in $\mathbf{C}$ are updated by minimizing the following error.

$$\mathbf{C} = \arg\min_{\mathbf{C}} \sum_i \sum_j \|\mathbf{C_i} - \mathbf{C_j}\mathbf{T_{ij}}\|^2 \tag{2}$$

where $\mathbf{T_{ij}}$ is the relative pose between the image pair (i, j).

We use the Gauss-Newton implementation in the open-source framework g2o [17] for the optimization. After traversing the whole mapping area and optimizing the complete pose graph, keyframes and their poses are stored as the map.

IV. Line based system

Besides the feature points, straight lines are also common patterns on some grounds in the form of brick boundaries and can also be used for localization. This is especially interesting on grounds with no sufficient feature points, e.g. texture-less tiles and paving slabs (Fig. 3). In the scope of this work we only consider the structure where lines are rectilinear, but extension to other patterns is possible. Using the rectilinearity assumption allows us to impose priors on the map that enable for globally consistent map building and drift correction without the need for loop closures.

We assume that the camera remains the same height above the ground and its image plane is parallel to the ground plane. Given a camera that is oblique to the ground plane, it is straightforward to rectify the imagery to a fronto-parallel view. Under these assumptions the camera pose can be represented by three parameters $[x_c, y_c, \phi_c]$ where x and y are its position and ϕ is its yaw angle.

The map for the line based system is represented as a set of lines on the 2D ground plane. Each line is parametrized by four parameters $[\phi^w, \gamma^w, p_1^w, p_2^w]$. ϕ denotes the orientation of the line. Under our rectilinearity assumption it can only take either 90 if the line is parallel to the y-axis or 0 if parallel to the x-axis. If $\phi = 90$, γ denotes the position of the line on the x-axis and $[p_1, p_2]$ defines the range of the line along y-axis. Otherwise γ is the position on the y-axis and $[p_1, p_2]$ is the range along x-axis. These parameters are defined in a global world frame, hence the superscript w.

In both mapping and localization phase, lines are first detected and parametrized by the position of its two end-points (x_1^c, y_1^c) and (x_2^c, y_2^c) in the camera coordinate frame (denoted by superscript c). Then the lines are transformed into the world frame and matched with existing map lines. The discrepancy between the detected lines and their corresponding map lines is used to correct the pose estimation of the camera. During mapping, the detected lines are used to update the map while during localization phase the map is fixed.

On some grounds where the brick boundaries are sparse, it could happen that no boundary is observed for several frames, and as a result camera pose can't be determined. In another case where the bricks are too dense, the data association could suffer from ambiguity. An Inertial Measurement Unit (IMU) can be used to solve both these problems. Therefore, we fuse the measurement from lines and the data from an IMU together using an Extended Kalman Filter (EKF).

Note that due to the ambiguity of lines, the line based system cannot provide the global position, but only the position within one tile of brick. Also drift correction is not performed explicitly via loop closures, but implicitly via nearest neighbor line matching. Consequently, an initial guess is necessary to initialize the system during run-time localization. Similar to the initialization of feature-based system on repeating patterns, the initial pose of the camera can be provided either manually or potentially with the aid of other sensors.

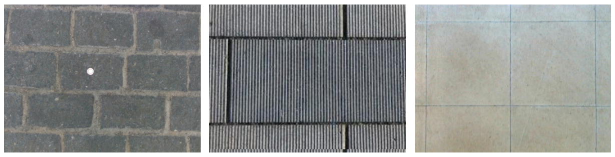

(a) Outdoor laid brick (b) Outdoor paving slabs (c) Indoor white tiles

Fig. 3: Boundary lines commonly observed from the view of a downward-facing camera.

A. Line segment detection and Parametrization

Hough transform [18] is a common option for line detection. However in our case the prior knowledge about the line pattern can be exploited to make the detection more robust and accurate, which cannot be incorporated in the Hough transform. Therefore we use sequential RANSAC instead and incorporate the prior knowledge to reject impossible line model proposals at early stage.

The line detection process is defined as following. Firstly, the edge image is extracted using a Canny edge detector [19]. A line model is then proposed by fitting a small subset of the edge points. It is then checked whether the fitted line satisfies the prior knowledge about the structure. The prior knowledge includes:

- Rectilinearity: new line must be nearly parallel or orthogonal to already accepted lines in the same image.
- Brick size: the gap between new line and accepted lines must be larger than the smallest possible brick size.

If the proposed line satisfies the above constraints, it is then scored by the number of points lying in its neighbor. After reiterating this procedure for a certain number of times, the line with the highest score is accepted if its score is above a certain threshold. Once a line is accepted, all points supporting this line are removed from the edge image and the above procedure is repeated. If no line with sufficient score can be fitted, the algorithm terminates.

After lines are detected, their endpoints need to be determined. For each intersection point resulting from the line fitting, it will be checked whether the two intersecting lines have supporting edge points on either sides. If it's not the case for a intersecting line, the intersection point is determined as an endpoint of that line. If a line extends to the image boundary, then the intersection of the line and the boundary is regarded as an endpoint. Note that in the latter case, the end point might be updated as more information of that particular line is available in successive frames.

B. Data Association and Map Management

Detected lines first needs to be projected into the world coordinate frame and transformed into the map line representation for data association. This transformation from the detected endpoints of a line $[x_1^c, y_1^c, x_2^c, y_2^c]$ to the map line representation $[\phi^w, \gamma^w, p_1^w, p_2^w]$ is given by

$$\phi^w = \arg\min_{\phi^w \in \{0,90\}} \left\| \phi_c^w + \arctan\left(\frac{y_2^c - y_1^c}{x_2^c - x_1^c}\right) - \phi^w \right\| \tag{3}$$

$$\Delta\phi = \phi^w - \arctan\left(\frac{y_2^c - y_1^c}{x_2^c - x_1^c}\right) \tag{4}$$

$$\begin{bmatrix} x_i^w \\ y_i^w \end{bmatrix} = \begin{bmatrix} \cos(\Delta\phi) & -\sin(\Delta\phi) \\ \sin(\Delta\phi) & \cos(\Delta\phi) \end{bmatrix} \begin{bmatrix} x_i^c \\ y_i^c \end{bmatrix}, i = 1, 2 \tag{5}$$

$$\gamma^w = \begin{cases} x_1^w, & \text{if } \phi^w = 90 \\ y_1^w, & \text{otherwise} \end{cases} \tag{6}$$

$$p_i^w = \begin{cases} y_i^w, & \text{if } \phi^w = 90 \\ x_i^w, & \text{otherwise,} \end{cases} \tag{7}$$

where x_c^w , y_c^w and ϕ_c^w are the current camera pose in the world coordinate frame.

After lines are transformed into the world frame, they can be matched to the lines in the map and the lines in a candidate pool via nearest neighbor search within a certain range in the parameter space. The candidate pool is a buffer preventing outliers from being added to the map. If matched to a map line, a pose innovation can be calculated to update the EKF. If matched to a candidate line, the confidence in that candidate increases by one. A candidate line is added into the map when its confidence is above a certain threshold or is deleted if it has not been matched for certain number of frames. Finally if no match can be found for a detected line, it is added into the candidate pool. This pipeline is illustrated in Fig. 4.

C. Sensor Fusion with EKF

As mentioned before, an EKF is used to fuse IMU and pose measurement from lines. For this purpose, we use an open-source framework MSF-EKF [20]. IMU is used for the model update step of the EKF and lines are used for the measurement update. If a detected line is matched with a map line, the innovation is calculated by

$$\Delta x_c^w = \gamma_m^w - \gamma_d^w \text{ , if } \phi_d^w = 90 \tag{8}$$

$$\Delta y_c^w = \gamma_m^w - \gamma_d^w \text{ , otherwise,} \tag{9}$$

where the subscript d means detected line and m for map line. The innovation for the yaw is given by Equation (4) . A line can only provide measurement in its lateral direction, therefore we set the variance in the other direction to a very large number so that the filter only updates for the lateral direction of each line. Also this means that our system will drift in one direction if all lines are parallel.

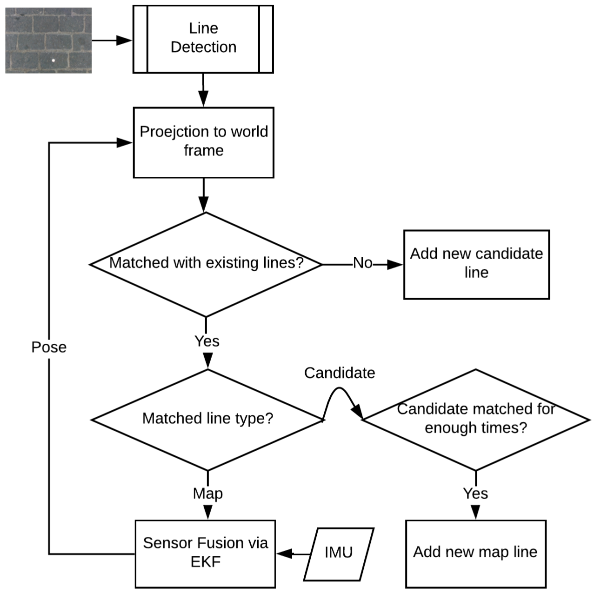


Fig. 4: The pipeline for the line-based mapping process. Detected lines in current frame are first transformed into the map line representation in the world coordinate frame and matched with existing map lines and candidate lines. If no match is found, a new candidate line is added. If matched to a map line, a pose innovation is calculated and applied to the EKF. If matched to a candidate line, the confidence in that candidate increases. A candidate line is added into the map after being matched for certain amount of times.

V. Experimental Results

The experimental setup is a manually-operated trolley with a laptop and downward-facing Intel RealSense ZR300[1], as shown in Fig. 5. The camera is mounted with its image plane parallel to the ground plane, which is achieved by manually adjusting the camera so that the internal accelerometer registers only vertical acceleration while standing still. The RealSense camera captures RGB images for the feature-based and line-based systems, plus an IMU reading for the line-based system.Since we consider 2D localization and the z-axis of the IMU is aligned with the gravity direction, only x- and y- accelerations and angular velocity for yaw angle are read by the line-based system.

A. Feature-Based System

The feature-based system is tested on tarmac, indoor textured tiles, and carpet, as shown in Fig. 6, and on laid brickwork as shown in Fig. 12b. The system is capable of online localization starting from a random point on the map, by using place recognition. This was done for the experiments on tarmac, which is a particularly challenging texture, thus demonstrating the potential of the method to work on other textures too.

[1]https://software.intel.com/en-us/realsense

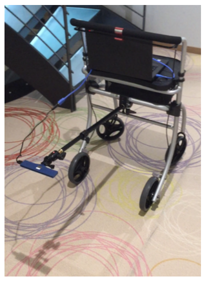


Fig. 5: Experimental setup - an Intel RealSense ZR300 mounted on a manually pushed trolley.

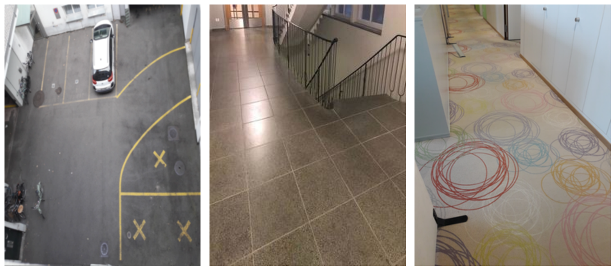


Fig. 6: Example ground-planes used in the experiments.

1) Feature Matching: The trolley is driven in multiple loops on a ground-plane, and the feature matching performance is evaluated by the number of feature matches between images. Three loops are done for tarmac, laid brickwork, and indoor tiles, while forward-backward trajectories are done for a corridor with carpet. Fig. 7 shows similarity matrices between individual images for all the four sequences. The main diagonal shows self-similarity of an individual image (as expected) while the offset diagonals show the peak in similarity when an area is re-visited, indicating that loop closures can be reliably detected based on the similarity measure. The unusual results for carpet in Fig. 7d,7h arise from the ambiguity in matching due to the carpet's repeated pattern.

Explicit detection of a repeated pattern on a ground-plane like carpet requires specific processing and is a goal for future work. However it is also handled implicitly in our current approach, in which there is system initialization at an approximately known position on the ground-plane, followed by online estimation, so that localization can be maintained even with a repeated design.

2) Map Mosaics: Fig. 8 shows generated map mosaics for indoor tiles, laid brickwork, and carpet. Fig. 9 shows a larger-scale example in which a map mosaic is generated for a parking lot. The mosaics are qualitatively correct, this being most visible in the parking lot example.

3) Numerical Results: This section contains a quantitative evaluation of the accuracy of the system for the parking lot and the indoor tiles. A Leica tracker[2] provides ground truth. Because there is an unknown offset in rotation between the coordinate frame of the Leica and the coordinate frame of our system, estimated trajectories are aligned to the ground

[2]https://leica-geosystems.com/

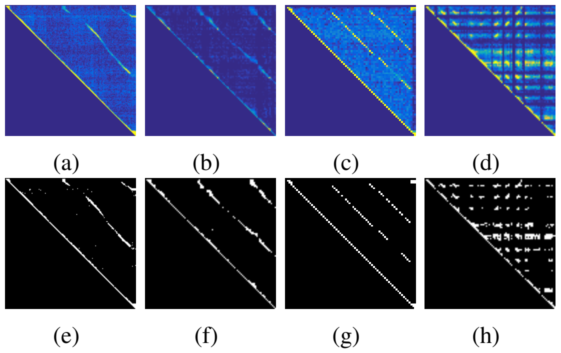


Fig. 7: *Top row:* Similarity matrices D for (a) tarmac, (b) laid brick, (c) indoor tiles, and (d) carpet, where D_{ij} denotes the number of matched features between frame i and frame j. *Bottom row:* Binary thresholded versions, There is a strong signal for detecting loop closure, as described in the main text.

TABLE I: Mean Error in Position

Mean Error [m]	tarmac	indoor tiles
ORB-SLAM	0.1105	-
StreetMap [mapping]	0.1162	0.1098
StreetMap [online localization]	0.1298	0.1106

truth with a rigid transformation.

For comparison, ORB-SLAM [3] is used as a state-of-the-art system. ORB-SLAM works for the parking lot, but quickly loses tracking on laid brickwork due to the drift along the normal direction of the ground plane, and it fails to initialize on the indoor tiling, while our system can still work, thanks to the incorporation of planar scene constraints using homography and the 2D motion assumption.

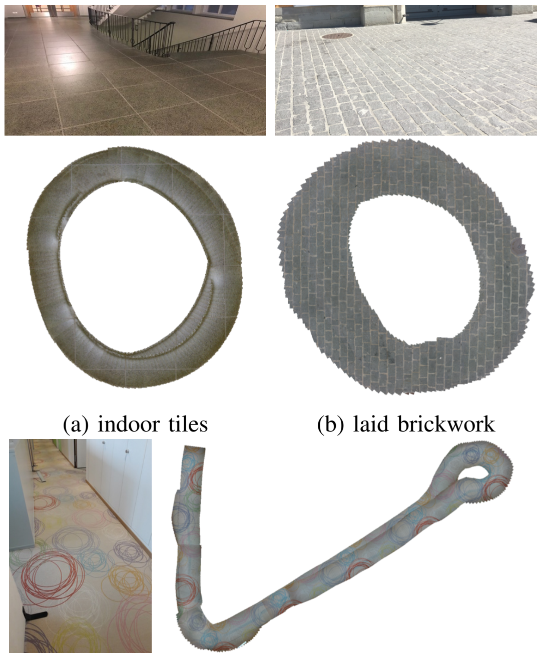
Fig. 10 shows estimated trajectories for the system in the parking lot, both in mapping mode and during online localization, compared with ORB-SLAM and ground truth. Fig. 11 shows the estimated trajectory for the system in mapping mode on the indoor tiling, compared with ground truth. Good agreement is evident for both ground-planes.

Mean errors are given in Table I. The online localization accuracy is slightly worse than mapping accuracy because, during the mapping, the trajectory is globally optimized via pose graph. Additionally any error in the map propagates to online localization.

In summary, our system is in good agreement with ground truth, is comparable to ORB-SLAM when the latter is operational, and is capable of handling more varied ground textures than ORB-SLAM.

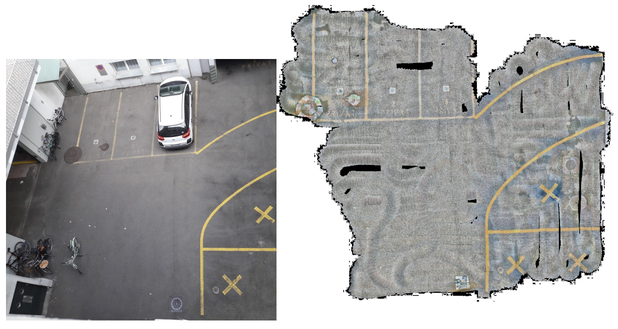
B. Line-Based System

The line-based system is tested on three types of ground-plane - paving slabs, laid brickwork, and indoor tiles, as shown in Fig. 12. Two types of quantitative measurement are reported in Table II. Firstly, for paving slabs and indoor tiles, the size of each tile is measured by hand and used as ground truth to evaluate the accuracy of the generated map. (For



(c) carpet

Fig. 8: Map mosaics for indoor tiles, laid brickwork , and carpet. The diameters of the circular trajectories are about 3m, and the full length of the trajectory on carpet is about 7m. Qualitative evaluation can be done by zooming on the tile or brickwork mosaics to inspect the rectilinearity of lines.



(a) scene (b) mosaic

Fig. 9: A parking lot with tarmac ground, and its map. The dimensions of the area is about 8x8m.

laid brickwork, the elements are not very regular in size and positioning, and the same evaluation cannot be performed). Secondly, the amount of drift is measured on loop closure once the platform has returned to its start position.

C. Computation and Storage Requirement

We report the processing time and storage requirement of both systems, running on a laptop with an Intel i7 CPU. Feature-based re-localization takes on average $91ms$ per frame. In our experiments we perform such re-localization every 6 frames, and keep track of the pose in between by accumulating estimated relative motion, which takes $10ms$ per frame. To store the map, namely the keyframes and poses,

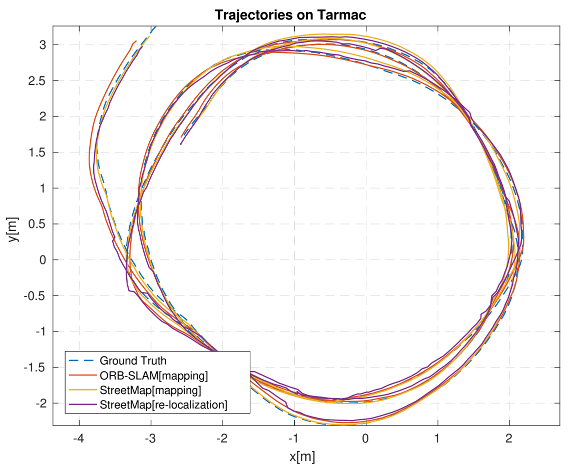


Fig. 10: Trajectories estimated by ORB-SLAM, our system in mapping mode and our system in localization mode along with the ground truth on the parking lot with tarmac. All three methods produce similar trajectory estimations which match well with the ground truth.

TABLE II: Mean Error in Position

	Tile Size Error[m]	Drift (x,y) [m]
Paving slabs	0.038 ± 0.0106	$(0.049, 0.041)$
Laid brickwork	-	(0.012,0.005)
Indoor tiles	0.0005 ± 0.0033	(0.019,0.020)

for the tarmac parking lot in Fig. 9 space of 418 *MB* is required.

The processing time for the line-based system depends on the complexity of the rectilinear pattern. For sparse pattern like indoor tiles in Fig. 12 it takes 4ms per frame. For dense laid brickwork it takes 19ms per frame. The map for the indoor tiles illustrated in Fig. 12 takes up 606 *Byte* only, thanks to the compact representation of the map as a set of lines.

VI. Conclusion

This paper has described a system for mapping a ground plane, and doing localization off the ground map, on a mobile robot with a downward-facing camera. The approach has the advantages of being purely onboard, working indoors or outdoors, and it is not affected by the presence of a surrounding human crowd. The latter is a particular motivation for this work, since a social mobile robot mingling in a crowd may experience periods of being densely encircled by humans, and this adversely affects both lateral-facing sensors and wireless technology like UWB. Experimental results showed the system working on a variety of challenging ground textures including tarmac and specular tiles.

As future work, we will study the robustness of the two systems against condition changes, e.g. different illumination during mapping and localization, and also their effectiveness with general camera viewpoints, e.g. oblique camera, for more flexible mounting choices. Two systems - for general texture, and for rectilinear texture - are studied separately in this work for scientific evaluation, however they could be potentially integrated for better performance. For instance when both types of landmarks are present, the feature-based system can substitute the IMU used in the line-based system to resolve ambiguity. Although the system has been shown to work on varied ground textures, it would be interesting to further investigate techniques for generality e.g. deep features or image patches.

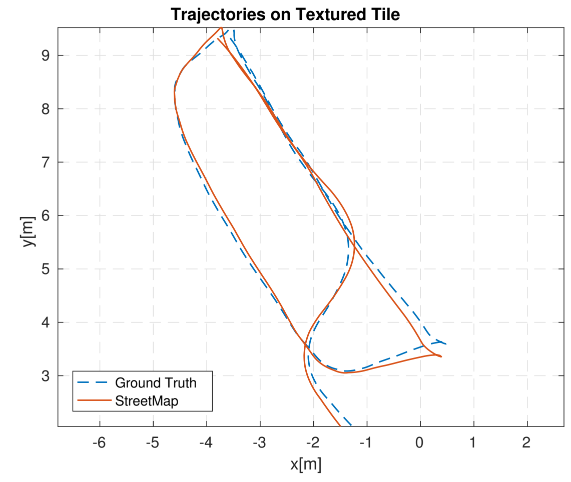


Fig. 11: Trajectory estimated by our system in mapping mode and the ground truth on the indoor tiles. In general, the estimation matches well with the ground truth.

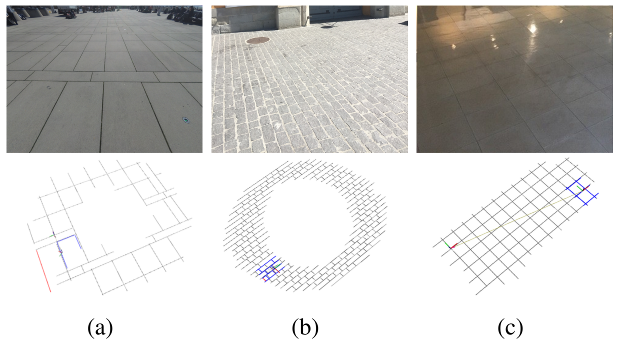


Fig. 12: *Top row:* Three common types of ground-plane - paving slabs, laid brickwork, and indoor tiles. *Bottom row:* Generated maps with explicit element boundaries. Blue and red lines are boundary lines detected in the current camera view, where the blue ones are those successfully matched with map lines and the red lines are those without corresponding map lines. Dimensions for the mapped areas are 3-6m.

References

[1] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on*

Robotics and Automation (ICRA), 2014.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *ECCV*, September 2014.

[3] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[4] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012.

[5] A. Kelly, "Mobile robot localization from large-scale appearance mosaics," *The International Journal of Robotics Research*, vol. 19, no. 11, pp. 1104–1125, 2000.

[6] N. Nourani-Vatani and P. V. K. Borges, "Correlation-based visual odometry for ground vehicles," *J. Field Robot.*, vol. 28, no. 5, pp. 742–768, Sep. 2011. [Online]. Available: http://dx.doi.org/10.1002/rob.20407

[7] H. Fang, M. Yang, R. Yang, and C. Wang, "Ground-texture-based localization for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 463–468, Sept 2009.

[8] R. Gonzalez, F. Rodriguez, J. L. Guzman, C. Pradalier, and R. Siegwart, "Combined visual odometry and visual compass for off-road mobile robots localization," *Robotica*, vol. 30, no. 6, p. 865878, 2012.

[9] I. Nagai and K. Watanabe, "Path tracking by a mobile robot equipped with only a downward facing camera," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 6053–6058.

[10] R. Gomez-Ojeda, F.-A. Moreno, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: a Stereo SLAM System through the Combination of Points and Line Segments," *arXiv preprint arXiv:1705.09479*, 2017.

[11] R. G. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, April 2010.

[12] T. K. Xia, M. Yang, and R. Q. Yang, "Vision based global localization for intelligent vehicles," in *2006 IEEE Intelligent Vehicles Symposium*, 2006, pp. 571–576.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[14] J. Shi and C. Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun 1994, pp. 593–600.

[15] E. Malis and M. Vargas, "Deeper understanding of the homography decomposition for vision-based control," INRIA, Research Report RR-6303, 2007. [Online]. Available: https://hal.inria.fr/inria-00174036

[16] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, winter 2010.

[17] R. Kmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3607–3613.

[18] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972. [Online]. Available: http://doi.acm.org/10.1145/361237.361242

[19] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov 1986.

[20] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.