

# High-Precision Localization Using Ground Texture

Linguang Zhang   Adam Finkelstein   Szymon Rusinkiewicz  
Princeton University

**Abstract**—Location-aware applications play an increasingly critical role in everyday life. However, satellite-based localization (e.g., GPS) has limited accuracy and can be unusable in dense urban areas and indoors. We introduce an image-based global localization system that is accurate to a few millimeters and performs reliable localization both indoors and outside. The key idea is to capture and index distinctive local keypoints in ground textures. This is based on the observation that ground textures including wood, carpet, tile, concrete, and asphalt may look random and homogeneous, but all contain cracks, scratches, or unique arrangements of fibers. These imperfections are persistent, and can serve as local features. Our system incorporates a downward-facing camera to capture the fine texture of the ground, together with an image processing pipeline that locates the captured texture patch in a compact database constructed offline. We demonstrate the capability of our system to robustly, accurately, and quickly locate test images on various types of outdoor and indoor ground surfaces. This paper contains a supplementary video. All datasets and code are available online at [microgps.cs.princeton.edu](http://microgps.cs.princeton.edu).

## I. INTRODUCTION

The Global Positioning System (GPS) receiver has become an essential component of both hand-held mobile devices and vehicles of all types. Applications of GPS, however, are constrained by a number of known limitations. A GPS receiver must have access to unobstructed lines of sight to a minimum of four satellites, and obscured satellites significantly jeopardize localization quality. Indoors, a GPS receiver either is slow to obtain a fix, or more likely does not work at all. Even outdoors, under optimal circumstances, accuracy is limited to a few meters (or perhaps a meter with modern SBAS systems). These limitations make GPS insufficient for fine-positioning applications such as guiding a car to a precise location in a parking lot, or guiding a robot within an indoor room or warehouse. To overcome the robustness and accuracy limitations of GPS, alternative localization technologies have been proposed, which are either less accurate than GPS (e.g., triangulation of cellphone towers and WiFi hotspots), or expensive and cumbersome to deploy (e.g., RFID localization or special-purpose sensors embedded in the environment). Inertial navigation and odometry, which are often used in robotics for fine-positioning tasks, require a known initial position, drift over time, and lose track (requiring manual re-initialization) when the device is powered off.

This paper proposes a system that provides millimeter-scale localization, both indoors and outside on land. The key observation behind our approach is that seemingly-random ground textures exhibit distinctive features that, in combination, provide a means for unique identification. Even apparently homogeneous surfaces contain small imperfections –

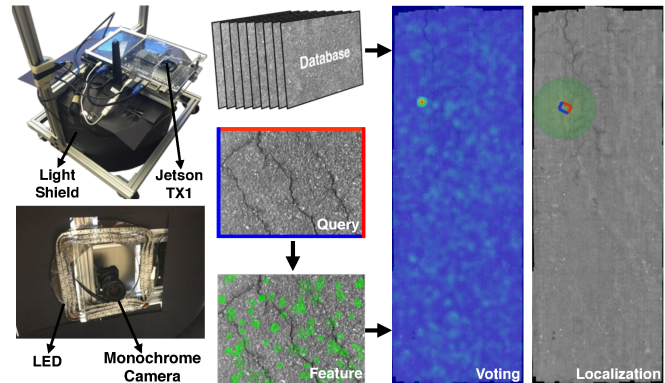


Fig. 1: System overview. Our test robot features an NVIDIA Jetson TX1 development board, which controls a Point Grey mono camera. A light shield and ring of LEDs around the camera provide controlled lighting. We first capture database images as a preprocess, and stitch them into a globally consistent map. Later, to locate the robot, we extract features from a query image, and they vote for potential image poses in the map. A peak in the voting map determines inlier features, from which we recover the pose of the query image.

cracks, scratches, or even a particular arrangement of carpet fibers – that are persistently identifiable as local features. While a single feature is not likely to be unique over a large area, the spatial relationship among a group of such features in a small region is likely to be distinctive, at least up to the uncertainty achievable with coarse localization methods such as GPS or WiFi triangulation. Inspired by this observation, we construct a system called Micro-GPS that includes a downward-facing camera to capture fine-scale ground textures, and an image processing unit capable of locating that texture patch in a pre-constructed compact database within a few hundred milliseconds.

The use of image features for precise localization has a rich history, including works such as Photo Tourism [1] and Computational Re-Photography [2]. Thus, a main contribution of our work is determining how some of the algorithms used for feature detection and matching in “natural” images, as used by previous work, can be adapted for “texture-like” images of the ground. In searching for a robust combination of such methods, we exploit two key advantages of ground-texture images. First, the ground can be photographed from much closer range than typical features in the environment, leading to an order-of-magnitude improvement in precision. Second, the statistics of texture-like images lead to a greater density of features, leading to greater robustness over time.

Our system consists of two phases: an offline database construction phase, and an online localization phase (Figure 1). We begin by collecting ground texture images and align-

ing them using global pose optimization. We extract local features (keypoints) and store them in a database, which is subsequently compressed to a manageable size. For localization, we find keypoints in a query image and search the database for candidate matches using approximate nearest neighbor matching. Because it is common for more than 90% of the matches to be spurious, we use voting to reject outliers, based on the observation that inlier matches will vote for a consistent location whereas outliers distribute their votes randomly. Finally, we use the remaining inlier matches to precisely calculate the location of the query image.

The major contributions of this paper are:

- Describing a low-cost global localization system based on ground textures and making relevant code and instructions available for reproduction.
- Capturing and making available datasets of seven indoor and outdoor ground textures.
- Investigating the design decisions necessary for practical matching in texture-like images, as opposed to natural images. This includes the choice of descriptor, strategies for reducing storage costs, and a robust voting procedure that can find inliers with high reliability.
- Demonstrating a real-world application of precise localization: a robot that uses Micro-GPS to record a path and then follow it with sub-centimeter accuracy.

The ability to localize a vehicle or robot precisely has the potential for far-reaching applications. A car could accurately park (or guide the driver to do so) in any location it recognizes from before, avoiding obstacles mere centimeters away. A continuously-updated map of potholes could be used to guide drivers to turn slightly to avoid them. The technology applies equally well to vehicles smaller than cars, such as Segways, electric wheelchairs, and mobility scooters for the elderly or disabled, any of which could be guided to precise locations or around hard-to-see obstacles. Indoor applications include guidance of warehouse robots and precise control over assistive robotics in the home.

## II. RELATED WORK

**Textures for Tracking and Localization:** Textures such as carpet, wood grain, concrete or asphalt all have bumps, grooves, and variations in color from location to location, and we typically use the overall pattern or *statistics* of this variation to recognize a particular material. Indeed computer-based modeling and recognition of textures traditionally proceeded along statistical lines [3, 4]. Moreover, researchers have successfully synthesized new texture by example using parametric [5] and non-parametric [6] models. However, when we study the *particular* arrangement of bumps and variations present at any location in real-world textures, we find that it is unlikely to be repeated elsewhere.

Kelly et al. [7] introduce a warehouse automation system in which a downward facing camera installed on each robot is used to help track the robot. They observe that ground surfaces usually exhibit cracks and scratches, and it

is possible to track the motion of the camera over a pre-constructed visual map. This work, however, still assumes a known initial location and surface textures are leveraged only for pairwise (*local*) frame matching, much as is done in an optical mouse. Other similar systems [8, 9] align the test frame with a small set of map frames determined either by an odometry or the most recent successful estimation. In contrast, our approach performs *global* localization, which could be used to initialize tracking systems such as these.

Clarkson et al. [10] demonstrate that seemingly-random textures can provide a means for unique identification. The authors observe that the fine-scale variations in the fibers of a piece of paper can be used to compute a “fingerprint” that uniquely identifies the piece of paper. Our work demonstrates that ground textures, including man-made ones such as carpet, share similar properties at sufficiently fine scales, and thus may be used for localization.

**Relocalization:** Structure from motion allows reconstruction of a large scale 3D point cloud offline, but relocating a newly captured image in the reconstructed point cloud without any initial guess about the camera position is challenging. Previous works explore direct 2D-to-3D matching [11] to estimate the 6 DoF pose of a photo with respect to a reconstructed point cloud. Li et al. [12] propose a method to leverage a co-occurrence prior for RANSAC and achieve relocalization on a larger georegistered 3D point cloud within a few seconds. Relocalization is an essential module of modern SLAM systems, such as ORB-SLAM [13], which uses a bag-of-words model for matching. Kendall et al. [14] train a convolutional neural network (PoseNet) to regress the input RGB image to the 6-DoF camera pose. Researchers have also explored using skylines from omni-images to perform relocalization [15].

All the above approaches, except PoseNet, involve large-scale feature matching, which quickly becomes a bottleneck because of the number of false matches. To speed up feature matching, more compact models can be constructed by selecting a subset of stable 3D points from the original models [16, 17]. An effective approach to handle a high outlier ratio is voting [18]. This has also proven successful in the field of image retrieval, where spatial verification is commonly applied to rerank the retrieved list of images, and variants of Hough voting have been proposed to improve efficiency and robustness [19–21]. With more sensors available, one can utilize the gravity direction [22] as an additional constraint. Baatz et al. [23] leverage both gravity direction and a 3D scene model to rectify images, transforming the 6-DOF pose estimation problem into a 3-DOF problem.

Mobile devices are ideal deployment platforms for a relocalization system. Lim et al. [24] achieve localization on a micro aerial vehicle at interactive framerates by distributing feature extraction over multiple frames. Middelberg et al. [25] achieve real-time performance by combining online camera tracking and an external localization server. Irschara et al. [26] and Wendel et al. [27] demonstrate that GPUs,

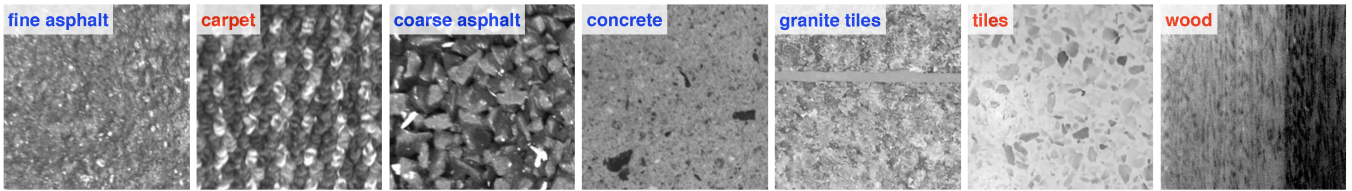


Fig. 2: Example texture patches cropped from our dataset. Outdoor and indoor textures are marked in blue and red respectively.

which are now widely available on mobile processors, can be used to accelerate localization.

Almost all of the above relocalization techniques rely on landmarks, such as buildings, that are normally positioned at least a few meters from the camera. This distance, combined with finite image resolution and inherent uncertainty in camera intrinsics, means that even a small error in feature localization results in a large uncertainty in estimated camera pose. This inaccuracy can be ameliorated by increasing the field of view of the camera [28, 29], because as more features are detected, more constraints can be introduced to improve pose estimation. Further uncertainty comes from the ambiguity in landmark identification, since it is not unusual to find buildings or parts of buildings (such as windows) that appear the same from a significant distance. Moreover, natural images used by the above systems suffer from perspective foreshortening, which brings difficulties to feature matching. Many features are not “time-invariant” in highly dynamic scenes. Thus, it is necessary to update the database frequently. Finally, these systems can be affected by changes in lighting. In contrast with these systems, our work positions the camera close to the texture being imaged and uses controlled lighting, leading to higher precision and robustness.

### III. SYSTEM

#### A. Mapping

**Hardware Setup and Data Collection:** Our imaging system consists of a Point Grey CM3 grayscale camera pointed downwards at the ground (Figure 1, left). A shield blocks ambient light around the portion of the ground imaged by the camera, and a set of LED lights arranged symmetrically around the lens provides rotation-invariant illumination. The distance from the camera to the ground is set to 260 mm for most types of textures we have experimented with. Our system is insensitive to this distance, as long as a sufficient number of features can be detected. The camera output is processed by an NVIDIA Jetson TX1 development board. Our prototype has the camera and development board mounted on a mobile cart, which may be moved manually or can be driven with a pair of computer-controlled motorized wheels. The latter capability is used for the “automatic path following” demonstration described in Section V. For initial data capture, however, we manually move the cart in a zig-zag path to ensure that an area can be fully covered. This process, while is easily mastered by non-experts, could be automated by putting more engineering effort or even through crowd-sourcing when there are more users.

**Image Stitching:** To construct a global map, we assume the that surface is locally planar, which is true even for most outdoor surfaces. Our image stitching pipeline consists of frame-to-frame matching followed by global optimization, leveraging extensive loop closures provided by the zig-zag path. Since the computation becomes significantly more expensive as the area grows, we split a large area into several regions (which we reconstruct separately) and link the already-reconstructed regions. This allows us to quickly map larger areas with decent quality. Figure 1, right shows the “asphalt” dataset, which covers 19.76m<sup>2</sup> in high detail.

**Datasets:** We have experimented with a variety of both indoor and outdoor datasets, covering ground types ranging from ordered (carpet) to highly stochastic (granite), and including both the presence (concrete) and absence (asphalt) of visible large-scale variation. We have also captured test images for the datasets on a different day (to allow perturbations to the ground surfaces) to evaluate our system. Figure 2 shows example patches from our dataset. We will make these datasets, together with databases of SIFT features and test-image sequences, available to the research community.

**Database Construction:** The final stage in building a map is extracting a set of features from the images and constructing a data structure for efficiently locating them. This step involves some key decisions, which we evaluate in Section IV. Here we only describe our actual implementation. We use the SIFT scale-space DoG detector and gradient orientation histogram descriptor [30], since we have found it to have high robustness and (with its GPU implementation [31]) reasonable computational time. For each image in the map, we typically find 1000 to 2000 SIFT keypoints, and randomly select 50 of them to be stored in the database. This limits the size of the database itself, as well as the data structures used for accelerating nearest-neighbor queries. We choose random selection after observing that features with higher DoG response are not necessarily highly repeatable features: they are just as likely to be due to noise, dust, etc. To further speed up computation and reduce the size of the database, we apply PCA [32] to the set of SIFT descriptors and project each descriptor onto the top  $k$  principal components. As described in Section IV, for good accuracy we typically use  $k = 8$  or  $k = 16$  in our implementation, and there is minimal cost to using a “universal” PCA basis constructed from a variety of textures, rather than a per-texture basis.

One of the major advantages of our system is that the height of the camera is fixed, so that the scale of a particular feature is also fixed. This means that when searching for a feature with scale  $s$  in the database, we only need to check

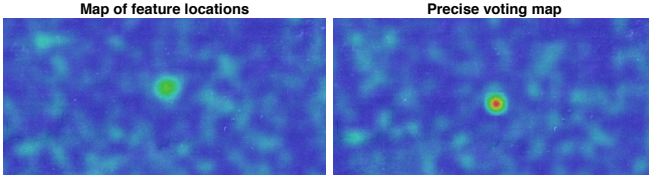


Fig. 3: Left: A simple strategy would be to vote for the locations of matched features. Right: The precise voting leads to a more defined local maximum.

features with scale  $s$  as well. In practice, to allow some inconsistency, we quantize scale into 10 buckets and divide the database into 10 groups based on scale. Then we build a search index for each group using FLANN [33]. During testing, given a feature with scale  $s$ , we only need to search for the nearest neighbor in the group to which  $s$  belongs.

### B. Localization

The input to our online localization phase is a single image. We assume that the height of the camera above the ground is the same as during mapping (or that the ratio of heights is known), so that the image scale is consistent with the images in the database.

**Feature Computation and Matching:** We first extract SIFT features from the test image and project onto  $k$  principal components, as in database construction. For each descriptor, we search for the nearest neighbor using the pre-build search index for the appropriate scale.

**Precise Voting:** Recall that we only keep 50 features per database image, so only a small subset of features will have a correct match in the database. Finding this small set of inliers is challenging, since methods such as RANSAC work poorly if outliers greatly outnumber inliers.

We instead adopt a voting approach based on the observation that, due to the randomness of ground textures, false matches are usually evenly distributed in the map. Fortunately, since true matches usually come from one or two images, they are concentrated in a small cluster. Figure 3, left, shows a heat map of feature matches in a database, with red indicating high density, green intermediate, and blue indicating low density. While we are able to build a system based on this principle, the correct features are distributed throughout the entire area corresponding to the test image. This leads to poor robustness, because there is only a moderately-high density of votes in the map near the location of the test image. The solution is to concentrate the votes: we want all of the true features to vote for the *same* point in the map, leading to a much greater difference between the peak corresponding to the true location and the background density of outliers.

In particular, each feature casts a vote for *the origin of the test image* by assuming that nearest neighbors are true matches. Denote a feature extracted from the test image as  $f_t$  and its nearest neighbor in the database as  $f_d$ . If the feature pair  $\{f_t, f_d\}$  is a true match, we can compute the pose of the test image  $T$  in world coordinates, denoted  $[R|t]_T^W$ , by

TABLE I: Performance of Micro-GPS. From left to right: texture type, elapsed time between capture of database and test sequence, number of test frames, and success rates using 8- and 16-dimensional descriptors.

Texture	Elapsed	# frames	Rate-8	Rate-16
fine asphalt	16 days	651	76.04%	95.24%
carpet	30 days	1179	99.49%	99.92%
coarse asphalt	17 days	771	97.54%	99.09%
concrete	26 days	797	83.31%	93.35%
granite tiles	6 days	862	79.47%	94.43%
tiles	18 days	1621	93.83%	98.40%
wood	0 days	311	59.48%	77.49%

composing the pose of  $f_d$  in world coordinates and the pose of  $f_t$  in the test image:

$$[R|t]_T^W = [R|t]_{f_d}^W [R|t]_{f_t}^{f_d} [R|t]_{f_t}^{f_t}, \quad (1)$$

where  $[R|t]_{f_t}^{f_d}$  is assumed to be the identity. We then vote for the location of the origin of the test image, which is the translational component of  $[R|t]_T^W$ .

Using this strategy, implemented via voting on a relatively fine spatial grid with each cell set to  $50 \times 50$  pixels, we find a much tighter peak of votes relative to the uniform background of outliers, as shown in Figure 3, right. After voting, the cell with the highest score is very likely to contain the true origin of the test image. We select all of the features in that peak as likely inliers, and perform RANSAC just on them to obtain a final estimate of the pose of the image.

## IV. EVALUATION

In order to evaluate the accuracy and robustness of a localization system, a typical approach would be to obtain ground-truth location and pose using a precise external measurement setup. However, this is impractical in our case due to the large areas mapped and the precision with which we are attempting to localize. Moreover, we are more interested in *repeatability*, rather than absolute accuracy, given that most of the applications we envision will involve going to (or avoiding) previously-mapped locations.

We therefore adopt an evaluation methodology based on comparing the query image against an image captured during mapping. Using the pose predicted by Micro-GPS, we find the closest image in the database, and compute feature correspondences (using all SIFT features in the image, not just the features stored in the database). If there are insufficiently many correspondences, we mark the localization result as a failure. We then compute a best-fit relative pose using those features. If the pose differs by more than 30 pixels (4.8 mm) in translation or  $1.5^\circ$  in rotation from the pose output by Micro-GPS, we again mark the result as a failure. Finally, given a sequence of consecutive poses that should be temporally coherent, we detect whether the poses of any frames differ significantly from their neighbors.

The performance of our system, implementing the pipeline described in Section III-B, is shown in Table I. The second column shows the elapsed time between capture of database and test sequence, which demonstrates that our system is



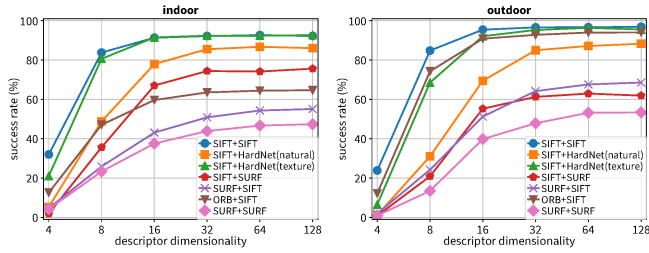


Fig. 4: The average performance of different *detector + descriptor* combinations on both indoor and outdoor datasets. The horizontal axis indicates the dimensionality of descriptors after PCA.

robust against changes in the scene. The two columns at right show performance with 8-dimensional and 16-dimensional descriptors, respectively. In the following, unless otherwise specified, we use 16-dimensional descriptors due to their good performance across all textures.

The correct pose is recovered over 90% of the time for most datasets (with independent per-frame localization and no use of temporal coherence), with the exception of the wood floor. This is because relatively few SIFT features are available in this dataset. (We have recently demonstrated that a highly repeatable feature detector can be learned in a texture-specific manner [34]. Unfortunately, such a pipeline is currently not efficient enough for a mobile platform.)

#### A. Impact of Design Decisions

**Selection of Feature:** We first evaluate the impact on accuracy of using different combinations of feature detector and descriptor. While SIFT [30] has been popular since its introduction more than a decade ago, more recent alternatives such as SURF [35], ORB [36] have been shown to achieve similar performance at lower computational cost. Even more recently, convolutional neural networks have been used in learned descriptors [37–42] to achieve much better matching performance than SIFT. In all of these cases, however, the performance has been optimized for *natural*, rather than texture-like, images. To evaluate the effectiveness of a learned descriptor on texture images, we select the recent well-performing HardNet [38] as our backbone network and learn a texture descriptor (HardNet-texture) using patches cropped from our dataset. During training, we also perform non-uniform intensity scaling to account for possible changes in exposure. Figure 4 compares the accuracy of different combinations of feature detector and descriptor. The SIFT *detector* outperforms both SURF and ORB, while both SIFT and HardNet-texture perform better than alternative *descriptors*. Because the SIFT descriptor can withstand more aggressive dimension reduction, it is the best choice for our current deployment. However, we observe that HardNet-texture shows significant improvement compared to the original HardNet optimized for natural images [43]. This suggests that domain-specific training may hold the promise for future improvements in the quality of learned descriptors.

**Choice of PCA Basis:** We next investigate whether the PCA basis used for dimensionality reduction should be specific to each dataset, or whether a “universal” basis computed from

TABLE II: For each type of texture, we evaluate the success rate (in percentage) by running our system with PCA bases computed from each texture and the union of all textures. Best numbers are bolded.

Texture	Basis							
	asphalt	carpet	coarse	concrete	granite	tiles	wood	union
asphalt	95.24	95.39	95.24	94.32	95.08	<b>95.70</b>	94.32	94.47
carpet	<b>100</b>	99.92	<b>100</b>	<b>100</b>	<b>100</b>	99.92	<b>100</b>	<b>100</b>
coarse	99.09	<b>99.35</b>	99.09	<b>99.35</b>	99.09	<b>99.35</b>	98.83	98.83
concrete	91.84	<b>93.73</b>	92.72	93.35	93.22	93.22	91.72	92.85
granite	<b>94.43</b>	93.85	94.08	93.62	<b>94.43</b>	94.08	93.16	93.97
tiles	98.27	<b>98.40</b>	97.90	98.27	98.09	<b>98.40</b>	97.59	97.78
wood	76.85	<b>78.78</b>	78.14	77.81	77.81	<b>78.78</b>	77.49	77.49

the union of different textures achieves similar performance. Table II shows localization performance for each combination of texture and PCA basis, including a basis computed from the union of all datasets. The difference caused by switching PCA basis is negligible, and we conclude that there is no drawback of using a single PCA basis computed from all of the datasets.

#### B. Downward- vs. Outward-Facing Cameras

Many existing systems focus on localization in natural images captured by an outward-facing camera. To compare our system with this approach to localization, we add an outward-facing camera (identical to the one used for Micro-GPS) to our setup and we trigger both cameras simultaneously. We use the VisualSfM structure-from-motion system [44, 45] to recover the 3D trajectory of the outward-facing camera. In order to compare the resulting trajectory to ours, we project it onto 2D (which we do by estimating the plane that the trajectory lies in using least squares), and recover the relative global scale, rotation, and translation (which we do by minimizing least-squares distance between points taken at the same timestamps).

Figure 5 compares both trajectories for two different environments, outdoors (with asphalt ground texture) and indoors (with the “tiles” texture). We observe that the trajectory of VisualSfM (in blue) is much noisier than that of Micro-GPS, with discrepancies of many centimeters. In contrast, the difference in estimated orientations is small (usually below 1 degree), suggesting that both methods were able to recover orientation successfully.

One factor that critically affects the performance of both systems is the number of SIFT features that can be detected. Our downward-facing camera detects an average of 1319 features per frame in the (outdoor) asphalt sequence and 2114 features in the (indoor) tile sequence, while the outward-facing camera detects only 637 and 256 features in the same settings. More detected features typically leads to more matched features, and hence greater localization accuracy. Nevertheless, outward-facing cameras are commonly used for tracking and, at the same speed of motion, are less susceptible to motion blur than downward-facing cameras. We conclude that our system can be used in conjunction with a tracking system based on an outward-facing camera, with comparable additional hardware and software costs.

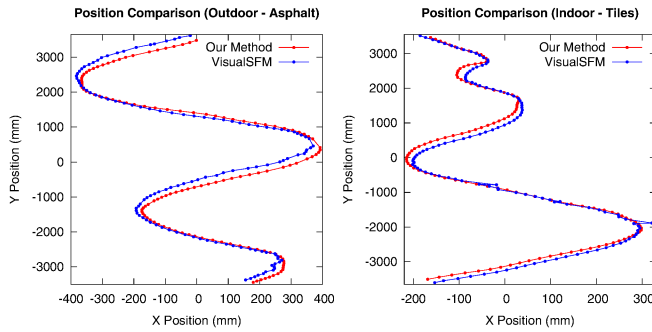


Fig. 5: Comparison of camera trajectories obtained using our system with downward-facing cameras (red lines) and a state-of-the-art structure-from-motion system using outward-pointing cameras (blue lines). Left: trajectories on the outdoor *asphalt* dataset. The distance between the trajectories is 98.8 mm on average (maximum 211.5 mm) while the mean angle between camera poses is 0.5 degrees (maximum 1.3 degrees). Right: trajectories on the indoor *tiles* dataset. The mean distance is 62.9 mm (maximum 197.7 mm) and the mean angular difference is 2.2 degrees (maximum 2.5 degrees).

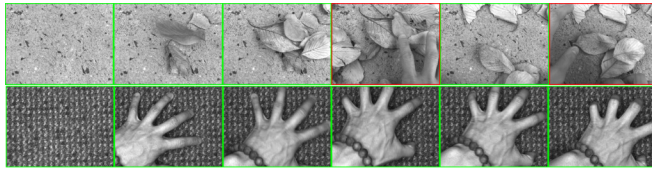


Fig. 6: Introducing occlusion and perturbation. First row: introducing occlusion by adding leaves. Second row: introducing occlusion and perturbation by scratching. The green bounding box represents success while red represents failure.

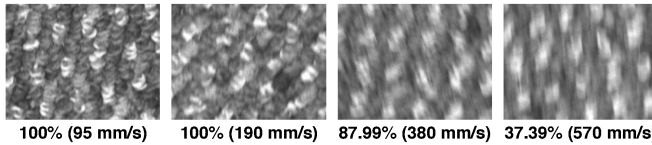


Fig. 7: Examples of motion-blurred images captured on the carpet. The actual speed and success rate correspond are shown below.

### C. Robustness

In the following section, we investigate the real-world applicability of the proposed system by stress-testing.

**Occlusion and Perturbation:** Two particular ways in which ground texture can change over time are occlusion (e.g., dirt, leaves, etc.) and perturbation of soft materials (e.g., walking on carpet). Figure 6, top, shows frames from a sequence in which more and more leaves are piled on a patch of concrete. Frames outlined in green represent success, while frames outlined in red represent failure of localization. Note that our voting procedure is robust against a substantial amount of occlusion. At bottom, we show frames from a sequence in which we scratch the carpet by hand. All frames in this sequence resulted in successful localization.

**Motion Blur:** Motion blur can easily happen when there is vibration or the camera is moving too fast. This perturbs both the feature detection and the computed feature descriptors, making localization less accurate. To evaluate how motion blur can affect the performance of our system, we use a robot which can run at a roughly-constant speed and evaluate performance by varying the speed. Unlike the previous

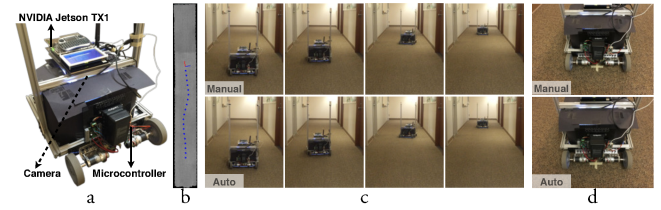


Fig. 8: A demonstration of path following. (a) Micro-GPS is implemented as a component of a mobile robot. (b) We generate a path by manually driving the robot. (c) We then use Micro-GPS to repeat the path. Screen-shots captured under manual and automatic driving modes are highly consistent. (d) The robot reaches the same ending position with high accuracy.

experiments, we deliberately lower the shutter speed to introduce motion blur. As shown in Figure 7, our system can still achieve reasonable success rate under moderate motion blur even with only 16-dimensional descriptors.

### V. APPLICATION: AUTOMATIC PATH FOLLOWING

Our system provides a simple, inexpensive solution to achieve fine absolute positioning, and mobile robots having such a requirement represent an ideal application. To demonstrate the practicality of this approach, we build a robot that is able to follow a designed path exactly without any initialization of the position. Our robot (shown in Figure 8a) has a differential drive composed of two 24 V DC geared motors with encoders for closed-loop control of the motors. Using the encoder readings, we implemented dead-reckoning odometry on board to track the position of the robot at reasonable accuracy within a short distance. The drift in odometry is corrected using Micro-GPS running on the on-board NVIDIA Jetson TX1 computer at  $\sim 4$ fps.

To test the repeatability of navigation using this strategy, we first manually drive the robot along a particular path (Figure 8b), and mark its final location on the ground using a piece of tape. The robot then goes back to its starting position and re-plays the same path, fully automatically. The sequences of the manual driving and automatic re-play are shown in the accompanying video; screen-shots from the video are compared in Figure 8c. As shown in Figure 8d, the robot ends up in almost exactly the same position after automatic path following as it did after the manual driving.

### VI. DISCUSSION

To accommodate larger working areas, we would need to increase the volume of the database, which could degenerate the robustness of our system due to noisier feature matching. Also, performing matching within a large database could raise the issue of efficiency. However, our system could work together with existing systems that provide coarse localization (e.g., GPS) to narrow down the working area. While we believe that our system is applicable to a wide range of floor and ground materials, localization is currently only possible in the presence of unique and stable textures. When a reliable ground surface is absent, leveraging subsurface features, such as what has been demonstrated in LGPR [46], could be a feasible solution.

## REFERENCES

- [1] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo Tourism: Exploring photo collections in 3D," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 835–846, Jul. 2006.
- [2] S. Bae, A. Agarwala, and F. Durand, "Computational re-photography," *ACM Transactions on Graphics*, vol. 29, no. 3, pp. 24:1–24:15, Jul. 2010.
- [3] K. Dana, B. van Ginneken, S. Nayar, and J. Koenderink, "Reflectance and texture of real-world surfaces," *ACM Transactions on Graphics*, vol. 18, no. 1, pp. 1–34, 1999.
- [4] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision (IJCV)*, vol. 43, no. 1, pp. 29–44, Jun. 2001.
- [5] D. Heeger and J. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. ACM SIGGRAPH*, 1995, pp. 229–238.
- [6] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *IEEE International Conference on Computer Vision (ICCV)*, 1999, pp. 1033–1038.
- [7] A. Kelly, B. Nagy, D. Stager, and R. Unnikrishnan, "An infrastructure-free automated guided vehicle based on computer vision," *IEEE Robotics & Automation Magazine*, vol. 14, no. 3, pp. 24–34, 2007.
- [8] 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, Sep. 2009.
- [9] K. Kozak and M. Alban, "Ranger: A ground-facing camera-based localization system for ground vehicles," in *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2016, pp. 170–178.
- [10] W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J. A. Halderman, and E. W. Felten, "Fingerprinting blank paper using commodity scanners," in *IEEE Symposium on Security and Privacy*, 2009, pp. 301–314.
- [11] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2D-to-3D matching," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 667–674.
- [12] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3D point clouds," in *European Conference on Computer Vision (ECCV)*, 2012, pp. 15–29.
- [13] R. Mur-Artal, J. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
- [14] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DOF camera relocalization," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2938–2946.
- [15] S. Ramalingam, S. Bouaziz, P. Sturm, and M. Brand, "Geolocalization using skylines from omni-images," in *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2009, pp. 23–30.
- [16] Y. Li, N. Snavely, and D. Huttenlocher, "Location recognition using prioritized feature matching," in *European Conference on Computer Vision (ECCV)*, 2010, pp. 791–804.
- [17] S. Cao and N. Snavely, "Minimal scene descriptions from structure from motion models," in *Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 461–468.
- [18] B. Zeisl, T. Sattler, and M. Pollefeys, "Camera pose voting for large-scale image-based localization," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2704–2712.
- [19] Y. Avrithis and G. Toliás, "Hough pyramid matching: Speeded-up geometry re-ranking for large scale image retrieval," *International Journal of Computer Vision (IJCV)*, vol. 107, no. 1, pp. 1–19, Mar. 2014.
- [20] X. Wu and K. Kashino, "Adaptive dither voting for robust spatial verification," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1877–1885.
- [21] J. L. Schönberger, T. Price, T. Sattler, J.-M. Frahm, and M. Pollefeys, "A vote-and-verify strategy for fast spatial verification in image retrieval," in *Asian Conference on Computer Vision (ACCV)*, 2016, pp. 321–337.
- [22] L. Svam, O. Enqvist, F. Kahl, and M. Oskarsson, "City-scale localization for cameras with known vertical direction," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 1455–1461, 2017.
- [23] G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, and M. Pollefeys, "Handling urban location recognition as a 2D homothetic problem," in *European Conference on Computer Vision (ECCV)*, 2010, pp. 266–279.
- [24] H. Lim, S. N. Sinha, M. F. Cohen, M. Uyttendaele, and H. J. Kim, "Real-time monocular image-based 6-DoF localization," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 476–492, Apr. 2015.
- [25] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt, "Scalable 6-DoF localization on mobile devices," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 268–283.
- [26] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof, "From structure-from-motion point clouds to fast location recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 2599–2606.
- [27] A. Wendel, A. Irschara, and H. Bischof, "Natural landmark-based monocular localization for MAVs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5792–5799.
- [28] M. Schönbein and A. Geiger, "Omnidirectional 3D reconstruction in augmented Manhattan worlds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 716–723.
- [29] C. Arth, M. Klopschitz, G. Reitmayr, and D. Schmalstieg, "Real-time self-localization from panoramic images on mobile devices," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 37–46.
- [30] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
- [31] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," <http://www.cs.unc.edu/~ccwu/siftgpu/>, 2007.
- [32] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.
- [33] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009, pp. 331–340.
- [34] L. Zhang and S. Rusinkiewicz, "Learning to detect features in texture images," in *Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6325–6333.
- [35] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 404–417.
- [36] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571.
- [37] Y. Tian, B. Fan, F. Wu *et al.*, "L2-Net: Deep learning of discriminative patch descriptor in Euclidean space," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [38] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," in *Advances in Neural Information Processing Systems*, 2017, pp. 4826–4837.
- [39] K. He, Y. Lu, and S. Sclaroff, "Local descriptors optimized for average precision," in *Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 596–605.
- [40] X. Zhang, X. Y. Felix, S. Kumar, and S.-F. Chang, "Learning spread-out local feature descriptors," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4605–4613.
- [41] Z. Luo, T. Shen, L. Zhou, S. Zhu, R. Zhang, Y. Yao, T. Fang, and L. Quan, "GeoDesc: Learning local descriptors by integrating geometry constraints," *European Conference on Computer Vision (ECCV)*, 2018.
- [42] M. Keller, Z. Chen, F. Maffra, P. Schmuck, and M. Chli, "Learning deep descriptors with scale-aware triplet networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [43] S. A. Winder and M. Brown, "Learning local image descriptors," in *Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.
- [44] C. Wu, "VisualSFM: A visual structure from motion system," <http://ccwu.me/vsfm/>, 2011.
- [45] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3057–3064.
- [46] M. Cornick, J. Koechling, B. Stanley, and B. Zhang, "Localizing ground penetrating radar: a step toward robust autonomous ground vehicle localization," *Journal of Field Robotics*, vol. 33, no. 1, pp. 82–102, 2016.