Lightweight Ground Texture Localization

Aaron Wilhelm[1] and Nils Napp[1]

***Abstract*— We present a lightweight ground texture based localization algorithm (L-GROUT) that improves the state of the art in performance and can be run in real-time on single board computers without GPU acceleration. Such computers are ubiquitous on small indoor robots and thus this work enables high-precision, millimeter-level localization without instrumenting, marking, or modifying the environment. The key innovations are an improved database feature extraction algorithm, a dimensionality reduction method based on locality preserving projections (LPP) that can accommodate faster-to-compute binary features, and an improved spatial filtering step that better preserves performance when the databases are tuned for lightweight applications. We demonstrate the approach by running the whole system on a low-cost single board computer (Raspberry Pi 4) to produce global localization estimates at greater than 4Hz on an outdoor asphalt dataset.**

I. INTRODUCTION

The ability to localize is fundamentally important to mobile robots, and as such there is a vast body of literature on localization algorithms, sensors, and systems [1]. For many application scenarios off-the-shelf solutions exist and today, most research is focused on applications where these methods are difficult or impractical to apply. This work specifically focuses on low-cost, high-accuracy localization for ground robots based on ground texture using general-purpose cameras. The applications of such systems can be both in GPS denied outdoor environments (where GPS-RTK approaches are infeasible) or for non-instrumented indoor environments, such as navigation around hallways, warehouses, or airports where installing motion capture systems would be impractical or prohibitively expensive. In addition to low deployment overhead, ground texture localization has different failure modes than localization based on other sensors that make it particularly attractive for large-scale indoor localization in busy and dynamic environments. Since texture images are taken by a downward facing camera that can be under the robot, lighting can be carefully controlled and the camera's view is guaranteed to be unobscured even in crowded spaces. Laser range finders and/or sideways looking cameras, in contrast, are difficult to use in such busy spaces. Early SLAM systems attempted to overcome issues in dynamic indoor scenes by using upward facing cameras [2], but the larger distance to the surface results in significantly lower position resolution, and ceilings seem to have a significant amount of perceptual aliasing compared to texture based approaches [3].

Modern localization and SLAM systems typically use a combination of proprioreceptive and exteroreceptive sensors for better localization. These systems use sensor fusion, taking advantage of inexpensive MEMS-based accelerometers and gyroscopes for high-speed pose-estimates combined with exteroreceptive sensors to prevent drift and provide a transformation estimate relative to an external map. The texture localization methods presented here address the exteroreceptive portion of these systems and provide absolute global localization estimates at a rate of several Hz, which is sufficiently fast to correct the accumulated drift of IMU-based pose estimates.

Previous works on ground texture based localization focused on performance and consequently leveraged fast CPUs, GPU acceleration, large memory access, and high quality cameras [3], [4], [5], [6]. However, one of the most promising aspects of ground texture localization is that it can achieve high-precision localization with minimal infrastructure overhead and potentially provide robust, high-quality localization for inexpensive and thus resource constrained robots. To address this gap, we present a lightweight ground texture based localization algorithm (L-GROUT). L-GROUT makes several modifications to the state-of-the-art localization technique first presented in MicroGPS, a seminal work that paired image feature matching with a spatial voting system to globally localize [3]. Our adaptations not only improve global localization accuracy on several textures, but they also make the algorithm more lightweight and thus ready to deploy on resource constrained systems. To demonstrate this, we test our algorithm with decreasing feature dimension lengths, as is required for lightweight applications.

Another underexplored aspect of implementing ground texture localization algorithms on cheaper robots is the camera quality. As the camera only needs to see the ground, previous works releasing datasets have built an enclosed lighting system to help control exposure conditions [3], [4]. These controlled conditions in theory enable robot designers to select cheaper cameras, however, a main engineering trade-off still exists between image resolution (the greater the resolution the higher feature quality and hence accuracy) and recording frequency (higher frames per second result in less motion blur). Thus, to evaluate L-GROUT's performance with lower quality cameras, we show high robustness when testing our algorithm with decreasing image resolutions. Finally, to illustrate the high performance of L-GROUT under real-world constraints, we timed L-GROUT on a modern computer without GPU acceleration and a Raspberry Pi 4, a popular platform for educational and DIY robotics.

[1]Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853, USA. ajw344@cornell.edu, nnapp@cornell.edu

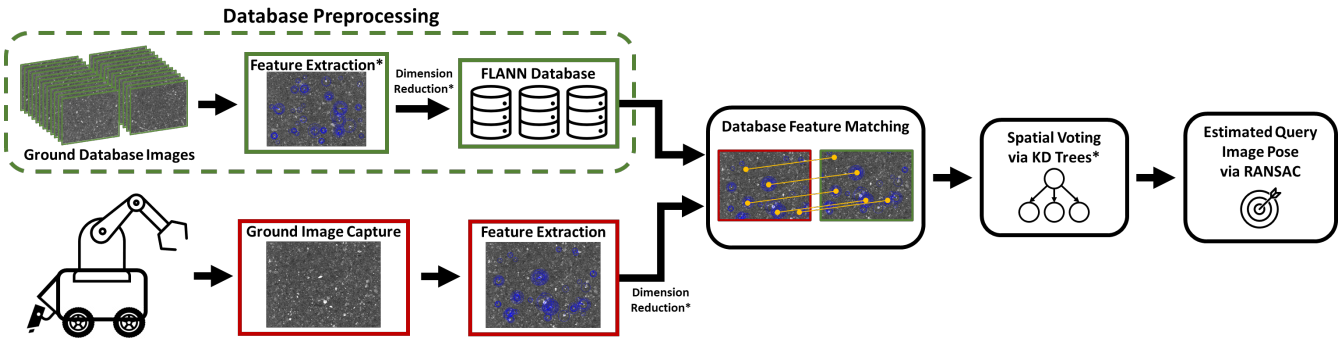

Fig. 1: The L-GROUT localization pipeline. Steps marked with an asterisk are changes made to the original pipeline introduced by MicroGPS [3]. First an offline map is constructed by extracting keypoints from database images and their feature vectors are stored in a set of FLANN databases [7]. At query time, keypoints are extracted from the query image and their features are used to query the database for the approximate nearest matches. Each database match votes for the estimated origin of the query image and the points in the highest density cluster (as determined by a radius search with a KD Tree) are selected as likely true positives. Finally, RANSAC is performed on the filtered set of query keypoints and their database matches to estimate the pose of the query image.

II. RELATED WORKS

Early work on using ground patterns to localize robots used dots on a linoleum floor and then applied ideas from star tracking to uniquely identify location-based patterns [8]. However, this approach only works on floor types that have randomly spaced, clearly detectable and identifiable features of the right density and it is not clear how this translates to more generic textures. Another stream of work for camera based localization and place recognition tries to learn features from data to create a visual vocabulary based on bag of words (BoW) representations of particular images and create topological maps over large areas [9]. These types of general-purpose place recognition methods are often used as building blocks in SLAM systems to perform place recognition and loop closure [10], [11]. The bag of words approach is used in StreetMap, one of the first papers for ground-based texture localization [5]. In their mapping pipeline, Chen et al. use a BoW built from ground image features to retrieve similar images from a map and localize the robot from the relative transformation. Although out-of-the-box application of BoW performs well on smaller ground texture datasets, subsequent work showed that this technique does not scale well to larger map areas [4].

In MicroGPS, Zhang et al. demonstrate an algorithm for global localization using approximate nearest neighbor (ANN) feature matching and a spatial voting system [3]. First, for each database image SIFT features [12] are extracted using SIFTGPU [13]. Of all the features extracted, 50 random features per image are selected and added to the database. Exploiting the fact that image features will be the same scale since camera distance from the ground is constant, Zhang et al. divide the main database into ten roughly equal sized ANN search structures using the Fast Library for Approximate Nearest Neighbors (FLANN) [7]. For speed up, the feature dimensions are reduced via Principal Component Analysis (PCA) that is derived from a training set. At query time, all SIFT features are extracted from the image and for each feature the ANN from the database is found. Next, a voting system is implemented where each database keypoint votes for what its estimation of the query image origin. Votes are binned via a fine resolution grid and RANSAC [14] is run on the feature matches to determine the estimated pose of the query image. Estimated poses are deemed accurate if they are within 4.8mm and 1.5 degrees of ground truth, a precise standard also adopted by later work [4], [15]. Along with their work, Zhang et al. released a ground texture localization dataset for use by future researchers. Until now, MicroGPS offers state-of-the-art accuracy for ground texture localization. While MicroGPS is able to exploit GPU acceleration for SIFT feature extraction, overall the algorithm is computationally expensive, making it impractical to apply on resource constrained hardware systems that do not have GPUs or high-performance CPUs.

Ground Texture Based Localization (GTBL) used a similar pipeline to MicroGPS but instead used LATCH binary features and introduced identity matching, where feature descriptors need to match exactly, with a database implementation [15]. By only considering nearby points in the database when a prior is available, GTBL was able to localize quicker than MicroGPS, but observed slower runtimes when no prior is available. Although GTBL had high accuracy than MicroGPS on smaller datasets, later testing by Schmid et al. showed that GTBL's accuracy did not scale well as the size of the dataset increased [4]. Deep Metric Learning for Ground Images attempts to simplify the localization problem by first using deep learning to perform similar image retrieval from the database [16]. Radhakrishnan et al. trained and deployed a Siamese CNN architecture to encode ground texture images. At localization time a query image is encoded by the network and the K most similar images in the database are found, their keypoints returned to be used for localization. When coupled with a localization algorithm

this technique has been shown to increase the localization accuracy [4], but the localization algorithm needs to be able to filter the database to keypoints that are only from nearby images. This means that the DML technique is incompatible with localization algorithms that do not keep the relationship between images and their keypoints in the database, such as MicroGPS [3], [16].

HD Ground released an extensive dataset that focused on four main textures (asphalt, carpet, cobblestone, laminate) and included additional textures and ground conditions such as wet and dirty [4]. Schmid et al. then tested Ranger, StreetMap, and GTBL on a variety of tests using the new dataset. Most recently SLAM was demonstrated for ground texture localization that uses local odometry information from the ground facing camera [6]. Loop closures are detected with a three part approach that checks for 1) VBoW similarity, 2) direct keypoint matches, and 3) uncertainty determined from the covariance matrix of the transformation calculated with the GTSAM library. While the algorihthm has been shown to perform well on the smaller MicroGPS dataset, it has yet to be explored how it performs on larger datasets such as HD Ground.

In contrast to these previous approaches, L-GROUT focuses on retaining the high accuracy of MicroGPS while improving upon speed and resource considerations. To accomplish this, our approach utilizes a modified database extraction method that incorporates highest response keypoints, Locality Preserving Projections (LPP) to accommodate faster-to-compute binary features, and a KD tree [17] for spatial filtering. We extensively test our algorithm under conditions that would be required for resource constrained platforms. Finally, we demonstrate the efficacy of our approach on the Raspberry Pi 4, a single board computer with limited memory and processing power.

III. SYSTEM

Figure 1 provides a system overview of L-GROUT. Of particular note are the three ways L-GROUT improves upon the original MicroGPS algorithm: database feature extraction, LPP dimension reduction, and KD tree voting for spatial filtering. Below we explain these changes and their significance in more detail.

A. Database Feature Extraction

When building their database, the authors of MicroGPS choose to randomly select 50 keypoints from the 1,000-2,000 keypoints extracted per database image [3]. They report that the high response features are just as likely to be due to noise, dust, etc. as they are to be prominent features. Although we concur that some of the most prominent features detected are not reliable, from our own experiments we have observed that some of the highest response keypoints are in fact high quality keypoints. We also observe a similar phenomenon with the largest keypoints in the image, which tend to be more reliable than randomly selected keypoints.

Therefore, we adopt a hybrid approach where we extract 1,000 keypoints, storing the 10 keypoints with the largest size and 40 other random keypoints. We choose to limit the number of extracted keypoints to 1,000 to keep our method scalable, especially when using keypoint extractors such as ORB (described below) that can have more than 20,000 keypoints per image. We keep a portion of the largest keypoints instead of the greatest intensity keypoints as the larger keypoints perform slightly better across the textures. Overall, this hybrid keypoint selection technique improves database feature matching robustness.

It should be noted that although one can increase localization accuracy if more keypoints are stored per database image, this also increases the memory cost of the database and the computation time of queries. The ideal number of keypoints depends upon the memory constraints, the target query speed, and the desired localization accuracy and is thus application dependent. To make a more relevant comparison to MicroGPS, for our experiments we have chosen to limit both algorithms to extracting 1,000 keypoints, saving 50 of those to the database per image.

B. Locality Preserving Projections

The authors of MicroGPS use PCA [18] to perform feature dimension reduction. However, variance is ill-defined for binary numbers, rendering PCA only suitable for feature vectors in the real number space.

For our algorithm, we instead adopt Locality Preserving Projections [19]. LPP is an unsupervised dimensionality reduction technique that linearly projects high dimensional features into a low dimensional subspace while trying to preserve neighborhood relationships. LPP constructs a weighted graph representing neighborhood information and from the graph's Laplacian matrix calculates a linear transformation to a lower dimensional subspace. Similarly to the work of [20], we simply use the ϵ-neighborhood procedure to generate the adjacency graph and keep the graph unweighted.

A significant advantage of LPP over PCA is that it is compatible with binary feature vectors. As many binary keypoint and feature detectors such as ORB [21] are considerably quicker than their real number counterparts, the use of LPP makes the feature extraction phase around twice as fast. Feature extraction time is a substantial part of the image processing and localization pipeline, so this results in a significant overall speedup.

C. KD Tree Voting for Spatial Filtering

Similar to MicroGPS, we adopt a voting mechanism to determine the spatial location of the query image. This stage is critical as we need to filter out a handful of true positive database matches from just under a thousand false positive matches. While MicroGPS presents an effective binning method for spatial filtering to find the true positives, below we present an updated method that improves accuracy at a similar computation cost.

First, we identify that although efficient, the binning approach of MicroGPS can lead to missed matches. In the case where the true image origin lies on the edge or corner of a bin, the votes can be divided across several bins. This

not only causes the true peak to be diluted, meaning that other bins of noisy votes are more likely to be selected, but it also means that fewer true positive database matches are selected for the RANSAC phase. To address this, we instead use a KD tree [17] approach with an additional filtering step based upon the estimated orientation of the query image.

A KD tree is constructed from the keypoint (x,y) coordinates of all of the estimated query image origins calculated from each of the potential database matches. As the number of points is fairly small and are two dimensional, the computation time to construct and search the KD tree is minimal. Then a radius search around each point is conducted, with the search radius equal to a translation error (in our experiments we chose 20mm). The members of the highest density cluster, i.e. the neighbors of the point with the highest neighbor count, are considered correct matches since they all voted for a similar estimated location. However, as an additional filtering step, we only count points as neighbors if the estimated pose of the query image is also within an orientation threshold, meaning that the two estimated poses match both in translation and orientation. From there RANSAC is conducted on the selected points to determine the final estimated image transformation and hence localize the robots. The KD Tree and the additional orientation check on the estimated help contribute to higher localization accuracy.

IV. EXPERIMENTS

A. Setup

We tested our algorithm using the four main textures (asphalt, carpet, cobblestone, laminate) from the HD Ground dataset. HD Ground provides training patches for the four main textures and these were combined to train texture-specific PCA and LPP vectors. The bin size for voting for MicroGPS and the vote search radius for L-GROUT were selected to achieve optimal performance.

To get a more fair accuracy comparison, we make one minor modification to the feature extraction process for MicroGPS. In the original paper, all features that were extracted from test images are used to query the database. With larger image resolutions this can result in more than 8,000 features per test image, which raises the localization accuracy but at a high computation and memory cost. For a more direct comparison to our algorithm, we limit both algorithms to exctracting 1,000 keypoints, saving 50 of those to the database during the mapping phase and at query time using only the top 1,000 keypoints for localization. This number was chosen since the authors report 1,000-2,000 extracted features per image when originally testing and optimizing their algorithm.

In our testing below we follow the accuracy guidelines first established in MicroGPS and later used by GTBL and HD Ground [3], [4], [15]. That is, a localization attempt is considered accurate if the estimated pose is within 4.8mm of translation and 1.5 degrees of rotation of the ground truth pose. We use the ground truths provided in the HD Ground dataset.

B. Global Localization Accuracy vs Time

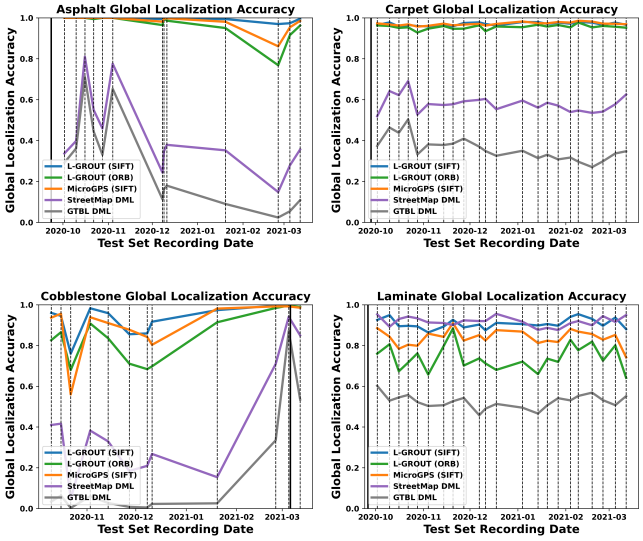

Fig. 2: The performance of ground texture based global localization algorithms over time on the four main textures from the HD Ground dataset [4]. The methods include L-GROUT with SIFT and ORB features, StreetMap Deep Metric Learning (DML), Ground Texture Based Localization (GTBL) DML, and MicroGPS optimized for each texture [3], [5], [15], [16]. Each dashed vertical line marks the test set recording date and the bold vertical line marks the database recording date.

For the four main textures in the HD Ground dataset we graph the performance of our algorithm and MicroGPS across the test datasets taken at different times. We include both a high accuracy version of L-GROUT that uses SIFT features with PCA and a faster L-GROUT version that uses ORB features with LPP. Both L-GROUT versions and MicroGPS use 16 dimensional feature vectors. To help contextualize the results with other algorithms, we include the accuracies for StreetMap Deep Metric Learning (DML) and Ground Texture Based Learning (GTBL) DML as reported in the HD Ground paper [4]. Each algorithm is run on the optimal image resolution that yields the highest accuracy. L-GROUT and MicroGPS use 1000 features as reasoned above and StreetMap DML and GTBL DML use the number of features that give the highest success rate.

As can be seen in Figure 2, across most of the textures L-GROUT with SIFT features is the highest performing algorithm, with the only exception being the laminate texture where StreetMap DML performs slightly better. L-GROUT with ORB performs similarly to MicroGPS and L-GROUT with SIFT features for the carpet dataset, but has a slightly lower accuracy for the other datasets. This is primarily due to a lower reliability of ORB features on those surfaces. Since L-GROUT and MicroGPS overall perform substantially better than the other algorithms, we chose to focus only on these algorithms for our following experiments.

We also make the observation that texture type has a larger impact on localization accuracy than the size of the database floor area coverage, as our algorithm and MicroGPS perform worst on laminate even though laminate is the smallest sized dataset (16.18 m^2 for laminate versus 106.12 m^2, 90.15 m^2, 59.28 m^2 for asphalt, carpet, and cobblestone, respectively).

C. Feature Dimension Length vs Global Localization Accuracy

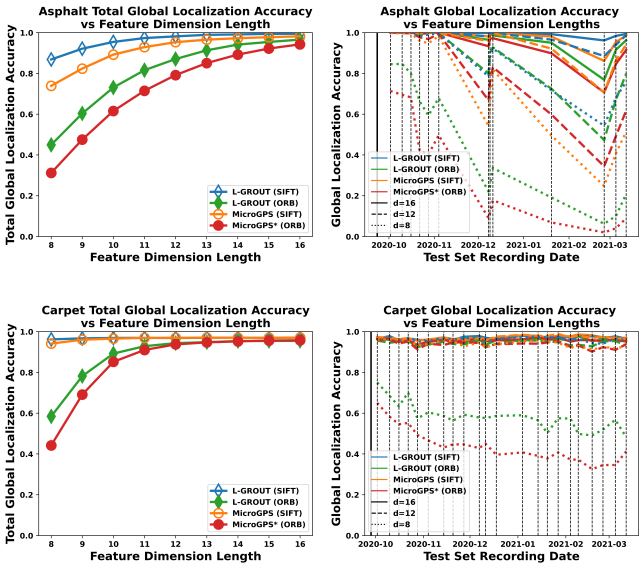

Fig. 3: The performance of L-GROUT and MicroGPS using various feature dimension lengths for both asphalt and carpet ground textures. For the left graphs, the accuracy is the global localization percentage of test images across all test sets for the respective texture. On the right, the global localization accuracy vs time for both textures was graphed for 16, 12, and 8 feature dimension lengths. *Note that MicroGPS does not originally support binary features, but for comparison LPP was used for dimension reduction to make MicroGPS compatible with binary features.

We next wanted to explore how MicroGPS and our algorithm perform on memory constrained systems. For both MicroGPS and our pipeline, the main memory cost is from storing the keypoint features and the FLANN databases. As the extracted feature dimensions decrease in size, so does the memory requirement of the algorithms. With this in mind, we tested both algorithms across a variety of feature dimension lengths with both SIFT and ORB features. Although MicroGPS cannot directly use binary features since it uses PCA, we chose to slightly modify their pipeline to use LPP and thus ORB features. This allows for a better comparison of the L-GROUT and MicroGPS to show how our novel database feature extraction approach and KD Tree voting technique contribute to the overall robustness of L-GROUT. For each algorithm, when using SIFT features we used PCA and adjusted the number of principal components of the feature vectors that we kept. Similarly, when employing ORB features we applied LPP and varied the number of projections that we preserved. Each data point represents the respective algorithm's localization accuracy across all recorded regular test datasets for that texture.

Figure 3 shows our results. Overall, the localization accuracy of both algorithms decrease as the number of feature dimensions decrease. However, across all feature dimension lengths our algorithm performs better than MicroGPS for the same feature extraction method. This indicates that our algorithm would be preferable on memory-constrained systems where one would have to decrease feature dimension length.

D. Image Resolution vs Global Localization Accuracy

Next, to explore how a cheaper camera might impact the algorithms' performance, we determined the effect of image resolution on the algorithms. Once again we tested both MicroGPS and our algorithm with both SIFT and ORB features, using PCA for SIFT features and LPP for ORB features. Similar to before, we modified the MicroGPS pipeline to use LPP to give insight on how the database feature extraction and voting procedures affect the robustness of MicroGPS and L-GROUT.

As can be seen in Figure 4, for each image resolution L-GROUT has a higher localization accuracy than MicroGPS with the same feature type. In general, as the image resolution decreases the localization accuracy does as well. However, L-GROUT is able to maintain a higher image resolution versus accuracy tradeoff than MicroGPS. This indicates that on lower quality cameras our algorithm would be a better choice.

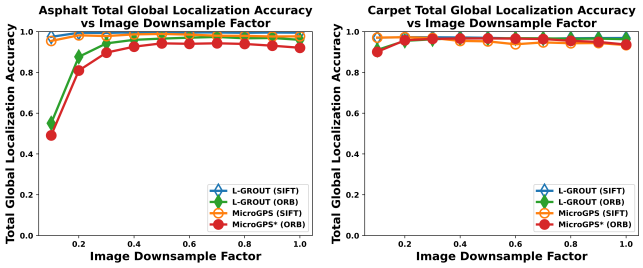

Fig. 4: The performance of our algorithm and MicroGPS across various image resolutions. As the image resolution decreases, the accuracy of both methods decrease as expected. However, our algorithm maintains a higher accuracy overall, suggesting that if a cheaper camera is used our algorithm would be a better choice than MicroGPS. *Note that MicroGPS does not originally support binary features, but for comparison LPP was used for dimension reduction to make MicroGPS compatible with binary features.

E. Time Comparison

Finally, we present a timing comparison between MicroGPS and L-GROUT with both SIFT and ORB features in Table I and Table II. To demonstrate how the algorithms

TABLE I: The timing characteristics of MicroGPS versus L-GROUT with SIFT features and with ORB features on a modern computer without GPU acceleration. The mean and standard deviation of the algorithms' timings for 100 randomly selected asphalt images are recorded, with the quickest mean execution times bolded. Note that overhead sections common with all algorithms are not included in the table.

Algorithm	Feature Extraction Time (ms)	DB Query Time (ms)	Voting Time (ms)	RANSAC Localization (ms)	Total (ms)
MicroGPS	58.5 ± 4.3	$\mathbf{3.5 \pm 0.2}$	$\mathbf{0.8 \pm 0.0}$	4.1 ± 0.4	85.6 ± 4.2
L-GROUT (SIFT)	56.7 ± 2.1	3.6 ± 0.1	3.1 ± 0.4	4.3 ± 0.4	86.7 ± 2.1
L-GROUT (ORB)	$\mathbf{13.4 \pm 1.5}$	4.1 ± 0.1	3.0 ± 0.2	$\mathbf{4.0 \pm 0.7}$	$\mathbf{43.6 \pm 2.3}$

TABLE II: The timing characteristics of MicroGPS versus L-GROUT with SIFT features and with ORB features on a Raspberry Pi 4. The mean and standard deviation of the algorithms' timings for 100 randomly selected asphalt images are recorded, with the quickest mean execution times bolded. Note that overhead sections common with all algorithms are not included in the table.

Algorithm	Feature Extraction Time (ms)	DB Query Time (ms)	Voting Time (ms)	RANSAC Localization (ms)	Total (ms)
MicroGPS	348.5 ± 10.8	56.3 ± 6.8	$\mathbf{2.5 \pm 0.1}$	15.6 ± 1.4	491.3 ± 12.6
L-GROUT (SIFT)	349.3 ± 5.7	$\mathbf{55.7 \pm 9.3}$	13.6 ± 1.8	16.4 ± 1.4	503.2 ± 10.9
L-GROUT (ORB)	$\mathbf{71.5 \pm 3.5}$	59.8 ± 6.7	12.4 ± 1.4	$\mathbf{14.9 \pm 2.8}$	$\mathbf{226.9 \pm 8.2}$

behave on platforms with different hardware resources, we timed the algorithms on a modern computer without GPU acceleration (Intel Core i7-10700 CPU @ 2.90GHz, 16 GB RAM) and a Raspberry Pi 4 (Broadcom BCM2711B0 CPU @ 1.5GHz, 4 GB RAM). Although the Raspberry Pi 4 has a Broadcom VideoCore VI GPU, it is not compatible with Nvidia CUDA library and thus is incapable of running GPU speedup libraries like GPUSIFT. The Raspberry Pi 4 has limited computing and memory resources, especially when compared to previous testing systems that had Intel i7 processors, 64 GB of RAM, Nvidia GPUs, etc. [3], [4], [5], [6]. With this in mind, the Raspberry Pi 4 provides an interesting benchmark to demonstrate how L-GROUT performs versus MicroGPS on resource constrained systems that are unable to use GPU acceleration.

As MicroGPS is not immediately compatible with binary features such as ORB features, we have only tested it with SIFT features. We executed L-GROUT with both SIFT and ORB features to show how the feature type and the modifications of the L-GROUT algorithm affect the computation time. For the experiment, the algorithms' perforomance was evaluated on 100 random asphalt test images. We chose asphalt as it is the dataset with the largest area and very relevant to a wide range of applications for mobile robots. Each image was downsampled by a scaling factor of 0.5 as at this scaling factor the overall global localization accuracy does not yet deteriorate. To avoid interference from other processes running on the computers, the minimum run time for each image over 10 trials was recorded. Then, across the 100 images the overall mean and standard deviation were calculated for each section of the algorithms.

The timing characteristics on the modern computer without GPU acceleration are summarized in Table I. Overall, L-GROUT with ORB features takes 43.6 ms, about half of the execution time on average required by MicroGPS (85.6 ms) and L-GROUT with SIFT features (86.7 ms). This speed advantage is primarily due to the quicker feature extraction for ORB features vs SIFT features. L-GROUT with SIFT and MicroGPS have similar timing characteristics, which is impressive as L-GROUT with SIFT yields a higher localization accuracy as shown in the previous sections.

We also demonstrate the performance of each algorithm on a Raspberry Pi 4. As can be seen from Table II, L-GROUT with ORB features is the fastest algorithm. Even with the limited resources of the Raspberry Pi 4, on average it takes L-GROUT with ORB 226.9 ms to localize a query image. This is more than twice as fast as the average timings of MicroGPS and L-GROUT with SIFT, 491.3 ms and 503.2 ms respectively. Once again, L-GROUT with SIFT has similar timing characteristics to MicroGPS with the main difference between their timings arising from L-GROUT's KD Tree voting technique. As shown in previous sections, the KD Tree voting enables a higher localization accuracy, which at this image scale only contributes to less than 2.5% additional computation time for the entire localization pipeline when compared to MicroGPS.

V. CONCLUSION

We present a novel algorithm for global localization from ground textures captured with a downward facing camera. When using SIFT features, our algorithm has state-of-the-art global localization accuracy and even when using faster-to-compute but lower performance ORB features, it outperforms several previous localization algorithms. We demonstrate that these accuracy improvements are especially prominent as the feature dimensions and image resolutions are decreased, making our algorithm a superior choice for resource constrained platforms such as the Raspberry Pi. We plan to add a mapping component and release this system as part of an easy to deploy open-source localization system. The current SIFT/ORB features perform very well on low-pile carpet and asphalt, but we plan to include other indoor ground textures and potentially optimize keypoint detectors and descriptors for use in the L-GROUT system.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] S.-Y. Hwang and J.-B. Song, "Monocular vision-based slam in indoor environment using corner, lamp, and door features from upward-looking camera," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 10, pp. 4804–4812, 2011.

[3] L. Zhang, A. Finkelstein, and S. Rusinkiewicz, "High-precision localization using ground texture," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6381–6387, IEEE, 2019.

[4] J. F. Schmid, S. F. Simon, R. Radhakrishnan, S. Frintrop, and R. Mester, "Hd ground-a database for ground texture based localization," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7628–7634, IEEE, 2022.

[5] X. Chen, A. S. Vempati, and P. Beardsley, "Streetmap-mapping and localization on ground planes using a downward facing camera," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1672–1679, IEEE, 2018.

[6] K. M. Hart, B. Englot, R. P. O'Shea, J. D. Kelly, and D. Martinez, "Monocular simultaneous localization and mapping using ground textures," *arXiv preprint arXiv:2303.05946*, 2023.

[7] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.," *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[8] Y. Fukase, H. Kanamori, and S. Kimura, "Self-localization system for robots using random dot floor patterns," in *Proceedings of the 30th International Symposium on Automation and Robotics in Construction and Mining (ISARC 2013): Building the Future in Automation and Robotics* (F. Hassani, O. Moselhi, and C. Haas, eds.), (Montreal, Canada), pp. 304–312, International Association for Automation and Robotics in Construction (IAARC), August 2013.

[9] M. Cummins and P. Newman, "FAB-MAP: Appearance-Based Place Recognition and Mapping using a Learned Visual Vocabulary Model," in *27th Intl Conf. on Machine Learning (ICML2010)*, 2010.

[10] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, "Openfabmap: An open source toolbox for appearance-based loop closure detection," in *2012 IEEE International Conference on Robotics and Automation*, pp. 4730–4735, 2012.

[11] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, pp. 1188–1197, October 2012.

[12] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[13] C. Wu, "Siftgpu: A gpu implementation of sift," 2007.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[15] J. F. Schmid, S. F. Simon, and R. Mester, "Ground texture based localization using compact binary descriptors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1315–1321, IEEE, 2020.

[16] R. Radhakrishnan, J. F. Schmid, R. Scholz, and L. Schmidt-Thieme, "Deep metric learning for ground images," *arXiv preprint arXiv:2109.01569*, 2021.

[17] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[18] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[19] X. He and P. Niyogi, "Locality preserving projections," *Advances in neural information processing systems*, vol. 16, 2003.

[20] B. Fan, Q. Kong, B. Zhang, H. Liu, C. Pan, and J. Lu, "Efficient nearest neighbor search in high dimensional hamming space," *Pattern Recognition*, vol. 99, p. 107082, 2020.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, pp. 2564–2571, Ieee, 2011.