Letter

A Triangulation-Based Visual Localization for Field Robots

James Liang, Yuxing Wang, Yingjie Chen, *Member, IEEE*, Baijian Yang, *Senior Member, IEEE*, and Dongfang Liu

Dear Editor,

Visual localization relies on local features and searches a pre-stored GPS-tagged image database to retrieve the reference image with the highest similarity in feature spaces to predict the current location [1]–[3]. For the conventional methods [4]–[6], local features are generally explored by multiple-stage feature extraction which first detects and then describes key-point features [4], [7]. The multiple-stage feature extraction is redundantly implemented, which is not memory and run-time efficient. Its performance degrades with challenging conditions such as poor lighting and weather variations (as shown in Fig. 1(a)) because the multiple-stage design may lose information in the quantization step which produces inadequately key-point features for matching. Another critical issue for existing visual localization is that most of the conventional methods are one-directional-based approaches, which only use one-directional images (front images) to search and match GPS-tag references [4], [8]. With the increase of database size, one-directional inputs can be homogeneous which makes it difficult for the localization algorithms to work robustly (as shown in Fig. 1(b)).

To address aforementioned problems, we propose a novel visual localization method that uses triangulation (front, left, and right) to robustly perform localization for the robotic system (as shown in Fig. 1(c). For the local feature extraction, we use a one-stage approach: an efficient implementation that can simultaneously detect and describe the key-point features of the input images to establish pixel-level correspondences. Since the one-stage method couples the detector and the descriptor closely, we keep the feature information untouched without the quantization step which improves the feature representations. In addition, we implement a generalized minimum clique graphs (GMCP) approach for feature matching, to organically manage features from all directions and triangulate the location prediction. Since the left and right scenes change more drastically than the front scene when a field robot is in linear motion, adding left and right images for feature matching is more informative to differentiate similar location references.

Related work: A line of work has developed different strategies to improve localization accuracy. For instance, [9] use ground texture features to compute the mobile robot positioning; [6] and [10] attempt to extract denser local features for key-point matching. Alternatively, [11] and [12] employ different types of global features, such as color histogram, GIST, and GPS coordinates to predict the final inference. However, the above improvements gain from extra features come at the price of longer matching times and higher memory consumption, and the results are still sub-optimal [4], [8]. In contrast to the existing one-directional-based method, we use three-directional views to triangulate a location, which is arithmetically effective and systematically intuitive.

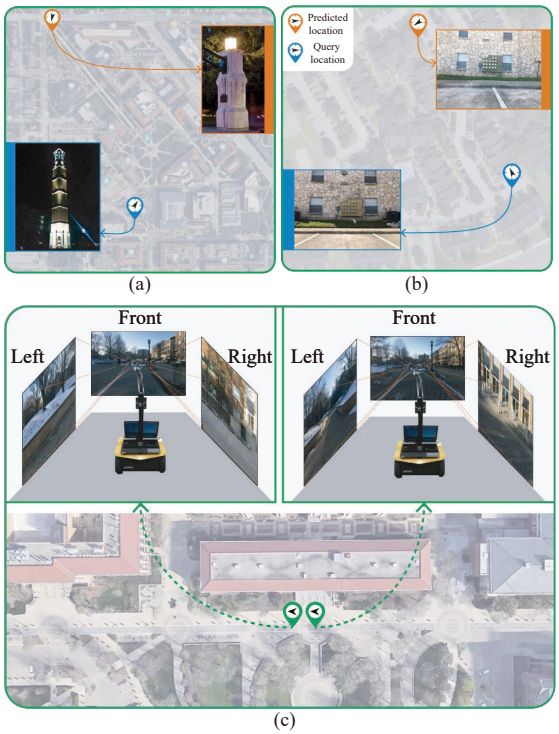

Fig. 1. (a) and (b) demonstrate the challenges to one-directional-based visual localization, such as lighting changes and similar appearances. The images for query (blue ones) and predicted locations (red ones) are shown for comparison. (c) illuminates our triangulation-based method. The images from the left and right cameras change quickly and they can be used to effectively identify different locations. We use camera inputs at t and $t+1$ for comparison.

Proposed approach: The working pipeline for our approach is demonstrated in Fig. 2. The top part of Fig. 2 shows the reference library construction pipeline. The first step is to collect GPS-tagged image data and store them as the reference library. We use the data from the reference library to train the location search engine. The bottom part of Fig. 2 shows the working procedures of the trained location search engine. Local features are first extracted from the query inputs. Then, the extracted features are compared with the reference library. The GPS-tag references with the closest distance to the query inputs are retrieved to predict the robot location.

Reference library construction: We collect image data under GPS-shadowed areas to build the reference library. Three high-resolution cameras are mounted in the front, left, and right directions of the field robot to record the scenes along the robot's trajectories. We slice images from the recorded video and label them with the corresponding GPS tags. All the GPS-tagged images are stored in the reference library.

Location search engine: The location search engine has two major working steps. First, query inputs are extracted for key-point features. Then, we use a modified GMCP [13] for feature matching to retrieve the most similar reference for localization prediction.

1) One-stage feature extraction: We couple detectors and descriptors closely to predict key-point image representation simultaneously. We first use a backbone CNN network $\mathcal{N}_{ext}$ to

Corresponding author: Dongfang Liu.

Citation: J. Liang, Y. X. Wang, Y. J. Chen, B. J. Yang, and D. F. Liu, "A triangulation-based visual localization for field robots," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 6, pp. 1083–1086, Jun. 2022.

J. Liang, Y. X. Wang, and D. F. Liu are with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: jcl3689@rit.edu; yw2009@rit.edu; dongfang.liu@rit.edu).

Y. J. Chen and B. J. Yang are with the Department of Computer and Information Technology, Purdue University, West Lafayette, IN 47907 USA (e-mail: victorchen@purdue.edu; byang@purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2022.105632

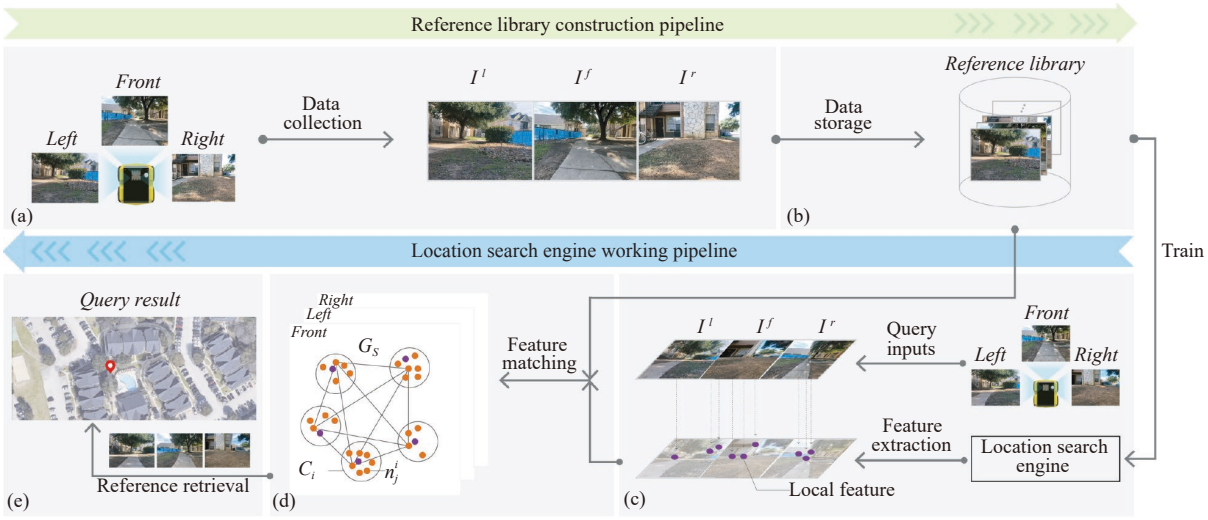

Fig. 2. The working pipeline for our approach. To construct the reference library, we first conduct (a) data collection, and then (b) data storage. After training, the location search engine can take the query image as input and perform (c) feature extraction, then (d) feature matching, and finally (e) reference retrieval.

process the input image set I in order to produce dense feature maps $f = \mathcal{N}_{ext}(I)$, $f \in \mathbb{R}^{h\times w\times n}$, where $h\times w\times n$ is the shape of feature map. Note I includes three separate images, I^f, I^l, I^r, which denote the images from the front, left, and right. The obtained feature maps are used by our feature descriptors and detectors.

The descriptor vectors q is decoded by the feature maps f

$$q_{ij} = f_{ij:},\ q_{ij} \in \mathbb{R}^n \tag{1}$$

where $i = 1,\ldots,h$ and $j = 1,\ldots,w$. We use the descriptor vectors to establish feature correspondences between image pairs to compute the Euclidean distance between image pairs.

Similarly, we use the feature maps f to produce a collection of detectors d in a similar fashion

$$d_k = f_{::k},\quad d_k \in \mathbb{R}^{h\times w} \tag{2}$$

where $k = 1,\ldots,n$. At a pixel (i, j), we perform a channel-level non-maximum suppression to obtain the detection scores $s_{ij} = \max_k d_{(ij)k}$. We then compute an image-level normalization for the detection score s_{ij}

$$s_{ij} = \frac{s_{ij}}{\sum_{i'=1}^{h}\sum_{j'=1}^{w} s_{i'j'}}. \tag{3}$$

2) Feature matching: We implement a GMCP-based feature matching technique which explores the k nearest neighbors and identifies the closest ones. We first define a graph $G = (\mathbf{N}, E, w)$, where $\mathbf{N}, E$, and w are a set of nodes, edges, and node costs respectively. We denote S to be the total number of local features extracted from the query input. The nodes $\mathbf{N}$ in G are a collection of all key-point local features from the reference library which are grouped into separate clusters C_i ($1 \le i \le S$). An individual node inside of a cluster represents the k nearest neighbors corresponding to a specific feature point for the query input. The ith query feature ($\equiv C$) can be defined as C_i and the mth candidate node inside of C_i can be defined as $n_m^i (\equiv \mathbf{N})$. All edges in G are connected if the nodes all come from different clusters. We can define the cost w for node distance as

$$w\left(n_m^i\right) = \| q^i - \phi\left(n_m^i\right) \| \tag{4}$$

where $\phi(\cdot)$ is an operator which returns the feature descriptor of a candidate node, q^i is the feature descriptor of the ith query point, and $\|\cdot\|$ computes the distance between the two features. The node distance cost indicates the similarity between the feature of the candidate node n_m^i and the corresponding query feature. A small value for the cost w demonstrates that the matching features have a high consistency and vice versa.

The feature matching task is to find the nearest neighbor for each query feature. We use a subgraph from G to explore the matching process that one node from each cluster ($\equiv$ candidate nearest neighbors set for one query feature) is selected. The subgraph $G_s = (\mathbf{N}_s, E_s, w_s)$ includes a set of nodes in the form of $\mathbf{N}_s = \{n_j^i \ldots\}(i = 1,\ldots,S)$ where n_j^i indicates the jth candidate node from cluster C_i (as shown in Fig. 2(e)). We can use $\mathbf{N}_s$ and (5) to express the solution for the cost of one query input

$$C(\mathbf{N}_s) = \frac{1}{2}\left(\sum_{i=1}^{S}\sum_{j=1,j\neq i}^{S}\left(w(\mathbf{N}s(i)) + w(\mathbf{N}_s(j))\right)\right) \tag{5}$$

where $C(\mathbf{N}_s)$ is the cost induced by all the nodes from $\mathbf{N}_s$ in the sub graph. $\mathbf{N}_s(i)$ denotes the ith node in $\mathbf{N}_s$. Accordingly, our feature matching is defined as

$$C_{\text{total}}\left(\mathbf{N}_{fs}, \mathbf{N}_{ls}, \mathbf{N}_{rs}\right) = \overbrace{\alpha(C(\mathbf{N}_{fs}))}^{\text{The front image cost}} + \overbrace{\beta(C(\mathbf{N}_{ls})) + \gamma(C(\mathbf{N}_{rs}))}^{\text{The left and right image cost}} \tag{6}$$

where $\mathbf{N}_{fs}, \mathbf{N}_{ls}$, and $\mathbf{N}_{rs}$ refer to the subgraphs for the front, left, and right images and α, β, and γ are adaptive weight (between 0 to 1) to represent the contributions of local features from each direction for total matching. A larger weight indicates a greater contribution from the corresponding direction to the overall cost and vice versa.

Adaptive weight selection: A pair of images with higher similarity in feature space indicates closer locations and vice versa. We, therefore, compute the similarity score between two sequential images (I_t and I_{t-1}) of each direction to organically select the adaptive weight to effectively differentiate locations. Concretely, we vectorize the feature maps f for two consecutive moments into feature vector $\hat{f}$. The similarity score is computed using the *cosine* distance

$$s_{t-1\to t} = \exp\left(-\frac{\hat{f_{t-1}}\cdot\hat{f_t}}{|\hat{f_{t-1}}||\hat{f_t}|}\right) \tag{7}$$

where $s_{t-1\to t}$ is the similarity score. Afterward, to select a specific adaptive weight, we turn the similarity score of each direction into his reciprocal and normalize the individual score with the sum of all directions. For instance, the front direction α can be obtained as

$$\alpha = \frac{s_f}{s_f + s_l + s_r} \tag{8}$$

where s_f, s_l, and s_r are similarity score of each direction calculated

by (7). By using this strategy, our method achieves further flexibility to more general settings.

Training objective: We define our training objective as the sum of the losses from feature extraction $\mathcal{L}_{ex}$ and feature matching $\mathcal{L}_{ma}$

$$\mathcal{L} = \mathcal{L}_{ex} + \mathcal{L}_{ma}. \tag{9}$$

For $\mathcal{L}_{ex}$, the loss function for feature extraction, we use the the triplet margin ranking loss, which trains our feature extraction in an unsupervised fashion. Our loss for feature extraction is given by

$$\mathcal{L}_{ex}\left(I_q, I_r^+, I_r^-\right) = \max\left(M + \mathcal{R}\left(I_q, I_r^+\right) - \mathcal{R}\left(I_q, I_r^-\right), 0\right) \tag{10}$$

where I_t, I_r^+, and I_r^- are image triplets including the query image, the positive reference (images from the same location), and the negative reference (images from a different location), respectively. Using this unsupervised training approach, our method can effectively learn image representation pertaining to which features should be suppressed or emphasized. For feature matching, we find a feasible solution for a clique of GMCP if G_s is complete. We retrieve the GPS-tag references for localization predictions. Our GPS-tag reference retrieval is meant to minimize the costs from nodes from each direction

$$\mathcal{L}_{ma}(\hat{\mathbf{N}}_s) = \operatorname{argmin} C_{\text{total}}(\mathbf{N}_{cs}, \mathbf{N}_{ls}, \mathbf{N}_{rs}) \tag{11}$$

where $\hat{\mathbf{N}}_s$ is the collection of nodes from the subgraphs of the front, left, and right images.

Experiments and evaluations: We purposefully select different university campus as well as some inner city areas under different weathers and seasons, where GPS signals are frequently denied due to the surrounding of dense buildings and vegetation. In order to have reliable GPS tags under GPS denied or partially shadowed areas, using Google location offers us a convenient access to the ground truth. More concretely, the GPS of each waypoint is manually selected and calibrated using Google Earth Map. A jackal robot platform is employed to collect the image data of each corresponding location. There are three Logitech C615 HD webcams mounted on the jackal robot that continuously video-record the scenes along the robot's trajectory. The jackal robot moves forward with a constant speed of 0.6 meters from one waypoint to another. After recording, we slice the video every second to obtain the reference images.

Implementation details: In the end, we obtain 146 828 images of 42 589 locations for the reference library. Our collected images generate ×4 triplets (402 628 triplets). We split the data, using 340 230 triplets for training and 62 398 for validation. Our training and evaluation are performed on a workstation with an Intel Core i7-7820X CPU and one NVIDIA GeForce GTX 1080Ti GPU. We use the collected dataset to construct the reference library and train the location search engine. For the one-stage feature extraction, we employ MobileNet [14] which is pretrained on ImageNet [15]. We remove the FC layers and only use convolution layers to initialize the feature extraction network $\mathcal{N}_{ext}$. In training, we use the margin M = 0.1 and perform 60 epochs with Adam [16], the learning rates of 10^{-3} and decay rate of 10^{-4} in every 6 epochs. In addition, we compare our method with three state-of-the-art methods, CRN [17], NetVLAD [18], and SARE [4], which are trained using both three and one-directional features for comparisons.

Evaluation metrics: Our work focuses on robotics operation so we evaluate our method based on the top one retrieval. We use the threshold of 7.8 meters for the true positive predictions. Namely, in our experiment, a predicted location is considered as a true positive if the top one retrieval from our reference library is located within 7.8 meters of the ground truth (GT) position. Average precision (AP) is reported for evaluation.

Field test results: We conduct field tests to examine the proposed method under three different conditions: 1) the daytime; 2) the nighttime; and 3) a snowy day. Table 1 shows the comparisons of our methods with state-of-the-art methods under different environmental conditions. For all methods using three-directional features, our methods outperform the compared methods by a significant margin. When compared to the methods using only one-directional features, our results exhibit better performance in accuracy, while achieving approximately 2× faster. Finally, we use a box plot to visualize the distance distribution to ground truths based on the predictions (see Fig. 3). Our triangulation-based method is more robust under different conditions compared to its counterparts.

Table 1. Performance Comparisons (3) and (1) Indicates Using Three- and One-Directional Features Respectively. * Indicates Using the Adaptive Weights Over the Average Weights

Method	Environmental conditions (AP (%))			Runtime (ms)
	Daytime	Nighttime	Snow	
Ours (3)*	**95.7**	**74.9**	**92.1**	50
Ours (3)	95.4	74.7	91.9	48
CRN (3)	75.1	66.6	73.2	89
NetVLAD (3)	72.0	61.3	68.2	83
SARE (3)	75.3	62.1	71.3	84
SIFT(3)	73.1	62.9	69.2	641
Ours (1)	51.3	30.2	49.7	45
CRN (1)	45.8	26.4	43.8	85
NetVLAD(1)	42.2	21.1	41.5	81
SARE (1)	50.1	29.8	48.9	82
SIFT (1)	43.7	25.4	44.3	610

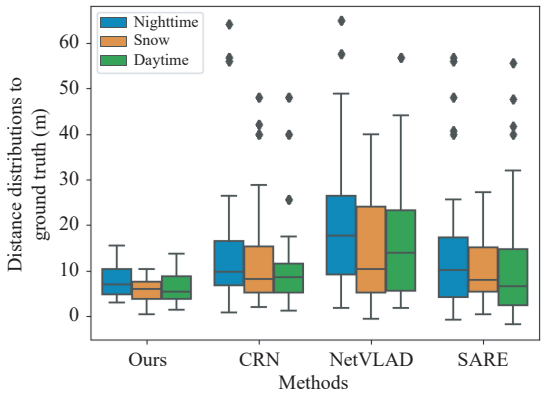

Fig. 3. Comparison with the state-of-the-art methods under different conditions. All methods use three-directional features.

Based on the results, we argue that our gains in accuracy stem from three sources. First, our method exploits triangulation-based features (three directional features) which effectively add the dimension of the image representations for location predictions. Second, our feature matching strategy can organically manage features from each direction in a flexible and general manner, rather than simply averaging feature contributions from each direction, which also brings improvement for our method (see Table 1).

Table 2 shows the comparison of our method with GPS signals in terms of AP and median error to the ground truth. Instead of using lighting and weather conditions, we categorize our data into two conditions: 1) open spaces, where only one side of sidewalks are closed off by buildings or trees; and 2) blocked spaces, where both sides of sidewalks are bordered by dense buildings or vegetation. The results indicate that our method outperforms GPS signals in AP and in average distance to ground truth under both conditions. Specifically, we observe that our method is more robust because the conditions of open spaces and blocked spaces have little impact on its performance, while GPS performs poorly in the blocked spaces where dense buildings or vegetation compromise its access. Our method performs consistently under either condition.

Table 2. Comparisons With GPS

Method		Ours	GPS
AP (%)	Open spaces	88.4	88.1
	Blocked spaces	86.3	68.9
Error to GT (m)	Open spaces	5.8	6.3
	Blocked spaces	6.1	12.8

Adaptive weight evaluation: We visualize the adaptive weight selection in order to achieve optimal performance (see Fig. 4). The three different color bars (blue, orange, and green) indicate the adaptive weight for the right, front, and left direction respectively, which also indirectly reflects the scenery change in each direction (a larger value means a bigger change). In the given example, the left scene has the fastest-moving scenes, while the left and the front have the approximately same moving pattern. These empirical observations corroborate the strategy of our adaptive weight selection that each weight is determined by the ratio of the scenery changes in (8).

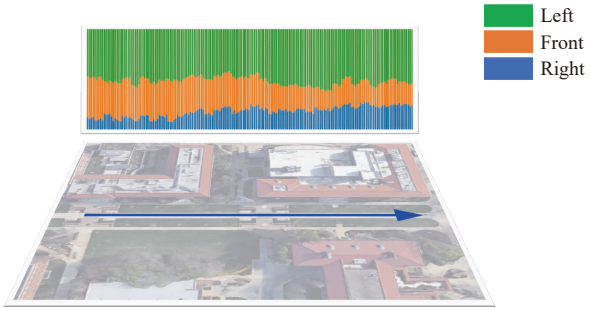

Fig. 4. Adaptive weight evaluation.

Conclusion: Localization under GPS shadowed areas is an important yet challenging task for field robot operation. In this study, we propose a novel visual localization method for field robots. Our method leverages triangulation views to accurately locate the robot in motion. We use one-stage feature extraction to effectively preserve local features for image representation and use a GMCP with flexible adaptive weights to manage features to triangulate the location prediction. The extensive experimental results indicate that our method is competitive with the existing state-of-the-art approaches and GPS.

References

[1] D. Liu, Y. Cui, X. Guo, W. Ding, B. Yang, and Y. Chen, "Visual localization for autonomous driving: Mapping the accurate location in the city maze," in *Proc. IEEE 25th Int. Conf. Pattern Recognition*, 2021, pp. 3170–3177.

[2] D. Liu, Y. Cui, Z. Cao, and Y. Chen, "Indoor navigation for mobile agents: A multimodal vision fusion model," in *Proc. IEEE Int. Joint Conf. Neural Networks*, 2020, pp. 1–8.

[3] H. Wang, W. Wang, T. Shu, W. Liang, and J. Shen, "Active visual information gathering for vision-language navigation," in *Proc. European Conf. Computer Vision*. Springer, 2020, pp. 307–322.

[4] L. Liu, H. Li, and Y. Dai, "Stochastic attraction-repulsion embedding for large scale image localization," in *Proc. IEEE Int. Conf. Computer Vision*, 2019, pp. 2570–2579.

[5] H. Wang, W. Wang, W. Liang, C. Xiong, and J. Shen, "Structured scene memory for vision-language navigation," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2021, pp. 8455–8464.

[6] H. Wang, W. Wang, X. Zhu, J. Dai, and L. Wang, "Collaborative visual navigation," arXiv preprint arXiv: 2107.01151, 2021.

[7] D. Liu, Y. Cui, L. Yan, C. Mousas, B. Yang, and Y. Chen, "DenserNet: Weakly supervised visual localization using multi-scale feature aggregation," in Proc. AAAI Conf. Artificial Intelligence , vol. 35, no. 7, pp. 6101–6109, May 2021.

[8] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 12716–12725.

[9] W. Wang, Y. Xu, J. Shen, and S.-C. Zhu, "Attentive fashion grammar network for fashion landmark detection and clothing category classification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 4271–4280.

[10] Q. Wang, Y. Fang, A. Ravula, F. Feng, X. Quan, and D. Liu, "WebFormer: The web-page transformer for structure information extraction," arXiv preprint arXiv: 2202.00217, 2022.

[11] A. R. Zamir and M. Shah, "Accurate image localization based on google maps street view," in *Proc. European Conf. Computer Vision*. Springer, 2010, pp. 255–268.

[12] Y. Song, J. Wen, D. Liu, and C. Yu, "Deep robotic grasping prediction with hierarchical RGB-D fusion," *Int. Journal of Control, Automation and Systems*, vol. 20, no. 1, pp. 243–254, 2022.

[13] M. Salarian, N. Iliev, A. E. Cetin, and R. Ansari, "Improved image-based localization using SFM and modified coordinate system transfer," *IEEE Trans. Multimedia*, vol. 20, no. 12, pp. 3298–3310, 2018.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv: 1704.04861, 2017.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv: 1412.6980, 2014.

[17] H. Jin Kim, E. Dunn, and J.-M. Frahm, "Learned contextual feature reweighting for image geo-localization," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 2136–2145.

[18] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.