IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Vision-based Topological Localization for MAVs

Simone Felicioni[1], Biagio Maria Rizzo[1], Claudio Tortorici[2] and Gabriele Costante[1]

***Abstract*—Vision-based topological localization is recently emerging as a promising alternative to metric pose estimation techniques in robotic navigation systems. Contrarily to the latter, which suffer from a quick degradation of their performance under non-ideal conditions (*e.g.,* scenes with poor illumination and low amount of textures), topological localization trades off precise metric positioning with a more robust and higher-level location representation. State-of-the-art works in this direction, however, often neglect the spatiotemporal relationships between poses that are naturally induced by robotic navigation. Furthermore, these techniques are nearly unexplored for autonomous flying platforms. Inspired by these considerations, in this work, we propose a vision-based topological localization approach designed for Micro Aerial Vehicles (MAVs) applications. Our strategy exploits the framework of graph recurrent neural networks to model the spatial and temporal dependencies and estimate the location of the robot with respect to a topological graph representing the environment. We demonstrate with experiments on different sets of scenarios, including scenes that considerably differ from those used in the training phase, that our approach is able to outperform state-of-the-art place recognition baselines.**

I. INTRODUCTION

Autonomous Micro Aerial Vehicles (MAVs) are becoming increasingly pervasive in a wide range of applications including surveillance, commercial delivery, and search and rescue. Enabling them with autonomous navigation capabilities requires different modules, among which localization plays a central role in the operation of the overall system. Vision-based localization, in particular, has attracted the interest of many research studies. The most popular solutions proposed in the last decades for this problem are grounded on the Visual Odometry (VO) and the Simultaneous Localization and Mapping (SLAM) frameworks [1], [2]. Their main product is a pose (and a map for SLAM systems) expressing the location of the robot relative to a metric reference system. Estimating precise metric transformations between frames is, however, often hindered by many factors. The performance of VO and SLAM approaches tend to degrade under non-ideal conditions, *i.e.,* texture-poor scenes, low amount of illumination, and fast motion dynamics, the latter being typical in MAV applications. In addition, their success is tightly linked to the accuracy of camera calibration procedures.

Most of the challenges arise from the requirement of precise metric positioning, which, in many applications, might be an unnecessary over-complication. If we shift the paradigm and take inspiration from humans and animals, we notice that they do not rely on metric positioning systems. Instead, they take advantage of semantic information and high-level relationship between objects, scenes, and shapes to localize in the environment [3]. Following these intuitions, visual place recognition and topological localization solutions have recently emerged as alternative paradigms to achieve robust localization even in non-ideal scenarios [4]–[7].

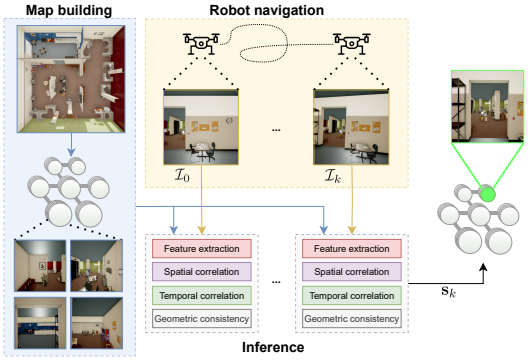


Fig. 1: Overview of the approach. The proposed strategy processes the images collected by the MAV and predicts the closest node in the topological graph that represents the environment.

Topological localization aims to estimate the position of the robot with respect to a semantic map similarly to place recognition and loop-closing strategies. The peculiarity of this paradigm lies behind the use of a topological map, *i.e.,* a sparse representation of the scene consisting of a set of nodes V and a set of edges E. A node $v \in V$ represents a salient location in the space, while an edge $e \in E$ is a connection between two positions due to, *e.g.,* mutual visibility or reachability.

A. Contributions

In this study, we focus on vision-based topological localization in indoor scenarios. The spatio-temporal relationships that are inherently induced during the robot navigation are considered only by few approaches in the literature, *i.e.,* [8]–[10], although only ground robot platforms are taken into account. MAV scenarios, instead, are significantly harder. An MAV is free to move in the 3D space, which increases the number of locations to be modeled in the semantic map and, thus, the chance of perceptual aliasing. In this work, we aim to take a further step in the research on vision-based topological localization for MAVs. In particular:

- We propose a novel Graph Recurrent Neural Network (GRNN) framework for topological localization in MAV contexts. Our algorithm processes a graph-based map and the sequence of images gathered by the MAV to estimate its position within the topological map.
- Our strategy models spatial relationships between adjacent locations and consider temporal dependencies of the im-

[1]Simone Felicioni, Biagio Maria Rizzo and Gabriele Costante are with Department of Engineering, University of Perugia, 06125 Perugia, Italy {simone.felicioni, biagiomaria.rizzo}@studenti.unipg.it, gabriele.costante@unipg.it

[2]Claudio Tortorici is with Autonomous Robotics Research Center, Technology Innovation Institute, United Arab Emirates claudio.tortorici@tii.ae

age sequence collected during the navigation to enforce consistency among consecutive predictions.
- To enhance the generalization capabilities of our approach, we develop a new procedural scripts to generate photorealistic environments and create a dataset composed of several scenarios with randomized characteristics.
- We evaluate our approach on environments significantly different from the training ones to further prove the generalization capabilities of our strategy and compare it against state-of-the-art baselines.

The rest of this work is organized as follows: Section II contains literature review; Section III describes the proposed approach; Section IV details the experimental setup analyzes the results; Section V discusses limitations and future development and Section VI draws the conclusions.

II. RELATED WORK

Vision-based robot localization solutions can be categorized into metric and semantic approaches, depending on the pose representation strategy they employ [6]. The former aims to compute the position and the orientation in the metric 6 degree-of-freedom space and has been extensively explored in the last decades. Besides the popular knowledge-based VO and Visual SLAM methodologies that exploit geometric relationships between images [11]–[13], also alternatives based on deep learning paradigms have emerged [14]–[16].

For those applications where the accuracy of metric approaches is insufficient or not required, semantic pose representations might provide a more robust and efficient alternative. They can be used to enhance SLAM or VO methods (*e.g.,* as loop closing modules [17], [18]), or act as stand-alone localization systems (such as in topological navigation or exploration tasks [8], [19], [20]). These approaches estimate the position by associating the observations with semantic entities that represent spatial locations. Those that rely on vision sensors for perception are, in general, referred to as Visual Place Recognition (VPR) methodologies.

VPR strategies [5]–[7] are designed to recognize places within a map based on visual, structural, and/or semantic cues. Their key modules are: i) place modeling, to map image data into a descriptor space; ii) place mapping, to build a representation of the world; and iii) belief generation, to match the current observation with a place within the map.

NetVLAD [21] is one of the most popular VPR solutions. It exploits an end-to-end trainable pooling layer based on a generalized VLAD (Vector of Locally Aggregated Descriptors) [22]. NetVLAD has been further improved by its more recent version, *i.e.,* Patch-NetVLAD [23]. It re-ranks the initial list of top-ranked candidate matches identified with the standard NetVLAD procedure by performing a geometric consistency check using a combination of local and global descriptors extracted from multi-level patches. Despite the impressive results achieved by [21] and [23], they exhibit two major limitations: i) spatial and temporal relationships, which might be present in many robot applications, are not taken into account and, thus, not exploited, and ii) matching descriptors might become computationally intensive as the number of visual cues increases [5]. Instead, [18] proposed a place recognition pipeline to identify loop closures under viewpoint variation during the robot navigation, specifically requiring a VIO method running in the background.

Spatial relationships are considered by topological localization (TL) approaches. TL [7], [24], [25], pioneered by Kuipers [26] in 1978, is a special case of VPR that does not model the environment as a database of uncorrelated entities, and, instead, assumes that topological relationships between locations exist. This assumption is structured through adjacency maps, where nodes and edges represent locations and their adjacency relationships, respectively. As an instance, in [4] localization is achieved with respect to a topological map encoding high-level textual information contained in the environment, *e.g.,* street signs and shop signage.

Among the possible data structures that can be used to encode topological maps, the graph representation is certainly one of the most adequate. To build models over graph-like inputs, Graph Neural Networks (GNNs) are recently gaining interest as they can naturally exploit spatial correlations between adjacent locations. [8] is among the first to formalize the robot navigation and localization tasks as a graph-based problem. The authors formulate the localization task as a GNN-based edge classification on the sub-graph cropped around the neighborhood of the previous predicted location. In [9], instead, a graph-based approach for ground robots localization is proposed. It exploits the objects in the scene and their relationship to improve robustness to perceptual aliasing. Nonetheless, a specific object detector is required, which could limit the possibility to deploy the approach in domains not considered when training the detector.

While modeling spatial relationships, the aforementioned image-to-image strategies do not take into account temporal dependencies. Contrarily, sequence-to-sequence VPR [27], [28] and loop closing approaches [29]–[31] consider the temporal evolution of the features to match database sequences with the query image stream. Instead, in topological localization, temporal dependencies can be considered only for the query image streams, as in [10], to enforce consistency between predictions of consecutive frames. More in detail, [10] presented an architecture for topological localization that models spatial and temporal correlation between observations and nodes. However, the authors consider only ground robot platforms and use the last layers of a deep feature extractor (*i.e.,* ResNet-18) to encode the observations, which has been shown to reduce the performance when compared to feature maps from intermediate layers [6]. Following this intuition, [32] redefines the message-passing procedure of GNN to accept feature maps from different layers of a deep network and improve the localization performance.

III. PROPOSED APPROACH

The objective of this work is to localize an MAV free to fly in the 3D space with respect to a topological map that describes the environment The topological map is structured as an undirect graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges, respectively. Each graph node $v_i \in \mathcal{V}$

represents a location in the environment and has associated a visual observation of the surrounding scene, captured by the robot front-looking camera. The observation is encoded by using *node features* extracted by a Convolutional Neural Network (CNN) on the images captured at that location during the graph construction phase (see Section III-A). An edge $e_{ij} \in \mathcal{E}$, instead, connects two nodes (v_i, v_j) if there exists an obstacle-free straight path between them.

At each timestep, our method processes the topological graph of the scene and the images captured by the MAV during the navigation, and predicts the closest location node in the graph. In particular, the features of the current MAV observation and the topological map are provided as inputs to a Graph Recurrent Neural Network (GRNN) model to achieve location (node) prediction. The model takes inspiration from [33] and [34] and exploit the message-passing paradigm to propagate information between the nodes of the graph to capture spatial relations, and through multiple timesteps to model temporal dependencies.

In the following, we first describe the initialization of the graph data structure, while in Section III-B we detail the MAV localization procedure. Finally, in Section III-C we provide details regarding model optimization.

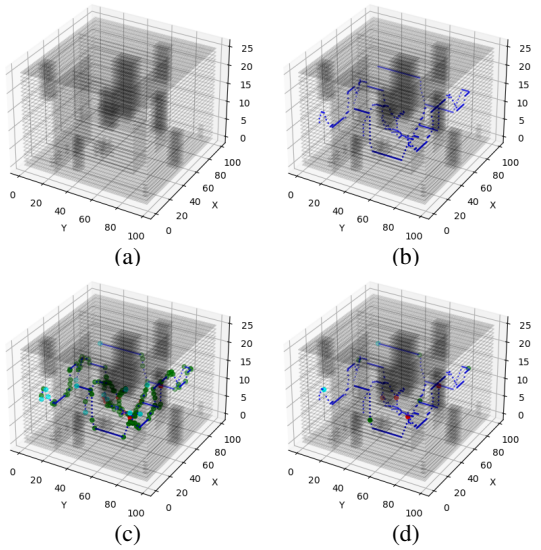


Fig. 2: 3D topological graph building process. 2a and 2b represent the voxel grid and the *skeletonization* of the environment, respectively. The following steps 2c-2d create the nodes and merge the closest ones to induce sparsity on the graph. A specific node color represents the type node according to its number of neighbors, *i.e.,* end-point (in light blue), corner (in green), or branch (in red).

A. Preliminary Graph Data Initialization

In the presented approach, the localization is performed on a pre-built graph representation of the environment. In this section, we describe the automatic graph construction procedure used to initialize the graph data.

We follow the intuitions of [35] to compute a one-voxel-thin skeleton diagram that acts as the graph representation of the environment. Starting from the 3D voxel grid (Figure 2a), we obtain the skeleton of the grid (Figure 2b) through a thinning algorithm [36], [37]. The skeleton is composed of the set of voxels that locally maximizes the distance from the obstacles. We classify each voxel according to its number of neighbors as either an *end*, *branch*, or *corner* node (see Figure 2c). The former has one neighbor and represents a boundary in the grid. A branch node, instead, has more than three neighbors, *i.e.,* it is the source of two or more branches. Finally, a corner node is a voxel that has two neighbors, and these three define a curve of $90°$, $135°$, $225°$, or $270°$ in the skeleton. Since, after this phase, the number of end, branch, and corner nodes might be significant, we induce sparsity by merging clusters of nodes (see Figure 2d). The output of this last phase is the graph structure of the environment.

Finally, we associate visual observations with each node of the graph. Due to viewpoint variations, *i.e.,* different parts of the scene might be observed from the same location depending on the heading of the front-looking camera, we consider two sets of nodes: *macro nodes* and *micro nodes*. Specifically, for each node v_i, which we refer to as macro node, we capture images by rotating the camera with respect to the yaw angle (*i.e.,* the heading) four times by 90 degrees. Each image is processed by the CNN extractor to compute the node features (see Section III-B1), which represent the encoding of the scene appearance captured from that orientation. This visual observation is referred to as *micro node* and the collection of the four micro nodes composes the macro node. We indicate the micro nodes with $v_{i,j}$, where the index $j = \{1, 2, 3, 4\}$ indicates the four camera orientations at the location represented by the macro node v_i. Thus, a graph contains M macro nodes, and $N = 4M$ micro nodes.

B. Localization procedure

Given a pre-built topological graph, we aim to localize the MAV during navigation by predicting the graph node closest to its actual position. The overall architecture is shown in Figure 3 and is composed of different modules:

Features extraction module: given the input image, computes the associated *global features*. This module is also used during the graph building phase to compute the *node features*;

Spatial correlation module: combines the global features from the current observation with those of graph nodes;

Temporal correlation module: propagates the spatial information through time to capture the temporal correlation that is naturally induced by the sequence of images;

Node prediction module: after spatial and temporal information are integrated into the graph, this module estimates for each node the probability of being the one that corresponds to the current robot location;

Geometric consistency module: weighs the scores from the node prediction module according to a geometric consistency check between the node images and the current observation.

In the following, we describe each module in detail.

1) Features Extraction Module: Using only features extracted from the last layers of deep networks often implies that important information of the input image is lost [6], as we also show in the ablation study presented in Section IV-D2 (see Table II). Hence, we propose to use feature maps from multiple levels to exploit different information. Similarly to

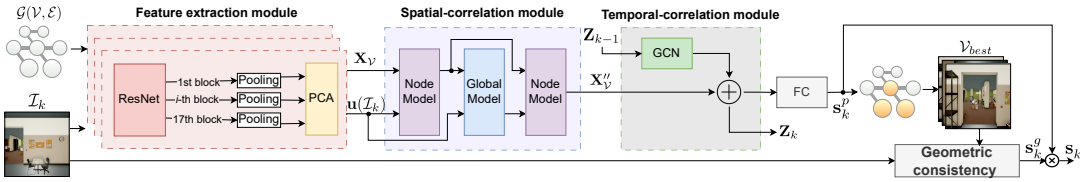


Fig. 3: Overview of the proposed architecture. The feature extraction module combines the outputs of different convolutional stages of ResNet-50 and processes it with PCA to obtain a 512-dimensional vector. This vector is combined with the features of the neighbor nodes by the spatial-correlation module. The temporal-correlation module, instead, propagates information from past graphs through time. Finally, the node scores computed by the FC layer are refined by a geometric consistency check that re-ranks the top k node candidates.

[38], we combine the output of the 17 convolutional blocks of ResNet-50 [39] to compute the image features. For each feature block, we apply a 2D average pooling to downsample the dimension into a fixed size (*i.e.,* (4, 4)). Feature maps are then flattened, concatenated, and Principal Component Analysis (PCA) is applied to obtain a 512-dimensional feature vector. This procedure is used to obtain both the *global features* $\mathbf{u}(\mathcal{I}_k)\in\mathbb{R}^{512}$ of the image $\mathcal{I}_k$ captured during the MAV navigation, and the micro *node features* $\mathbf{x}(v_{i,j})\in\mathbb{R}^{512}$, associated with the image of the micro node j (and macro node v_i) collected during the graph building phase. We indicate with $\mathbf{X}_\mathcal{V}\in\mathbb{R}^{M\times 4\times 512}$ the matrix containing all the micro node features of the graph, such that:

$$\mathbf{X}_\mathcal{V}[i,j,:]=\mathbf{x}(v_{i,j}),\ \forall i=1,\dots,M, j=1,\dots,4 \tag{1}$$

Algorithm 1: Spatial-correlation module

```
function SpatialCorrelationModule(X_V, E, u(I_k))
    X'_V = NodeModel(X_V, E, u(I_k))
    u' = GlobalModel(X'_V, u(I_k))
    X''_V = NodeModel(X'_V, E, u'(I_k))
    return X''_V
end function
```

Algorithm 2: Node Model

```
function NodeModel(X_V, E, u(I_k))
    for i∈{1...M}
        for j∈{1,2,3,4}
            N_i = X_V[z,:,:], ∀z : e_iz∈E
            X̃_V[i,j,:] ← Φ^v(X_V[i,j,:], N_i)
        end for
    end for
    ũ(I_k) ← ρ^{u→v}(u(I_k))
    X'_V → Φ^{v,u}(X_V, X̃_V, ũ(I_k))
    return X'_V
end function
```

Algorithm 3: Global Model

```
function GlobalModel(X'_V, u(I_k))
    X̂'_V ← ρ^{v→u}(X'_V)
    u'(I_k) ← Φ^u(X̂'_V, u(I_k))
    return u'(I_k)
end function
```

2) Spatial correlation: The global features of the current observation $\mathbf{u}(\mathcal{I}_k)$ and the node features $\mathbf{X}_\mathcal{V}$ are combined with a Graph Neural Network (GNN), taking advantage of the message-passing framework based on [33]. It propagates $\mathbf{u}(\mathcal{I}_k)$ through the graph and exploits the spatial relationships between graph nodes. The GNN is composed of a combination of computational blocks, as shown in Algorithm 1: the Node Model and Global Model. The former (see Algorithm 2) updates each node's features by combining them with those of its neighbors and the global features of the current observation. The Global Model (see Algorithm 3) integrates the updated node features with the global features.

The Node Model uses the following functions:

- $\Phi^v:\mathbb{R}^{512}\to\mathbb{R}^{256}$ implements the message-passing framework. First, the features of a micro node $\mathbf{X}_\mathcal{V}[i,j,:]$ and those of each of its neighbors $\mathcal{N}_i$ are separately fed into a Multi-Layer Perceptron (MLP) network to obtain the associated encodings. These are then averaged into a 256-element vector that is stored into $\tilde{\mathbf{X}}_\mathcal{V}[i,j,:]$.
- $\rho^{u\to v}:\mathbb{R}^{512}\to\mathbb{R}^{M\times 4\times 512}$ expands the global features $\mathbf{u}(\mathcal{I}_k)$ by replicating them $M\times 4$ times, *i.e.,* $\tilde{\mathbf{u}}(\mathcal{I}_k)\in\mathbb{R}^{M\times 4\times 512}$.
- $\Phi^{v,u}:\mathbb{R}^{M\times 4\times 1280}\to\mathbb{R}^{M\times 4\times 512}$ is an MLP network with the same structure as Φ^v that combines $\tilde{\mathbf{u}}(\mathcal{I}_k)$, the node features matrix $\mathbf{X}_\mathcal{V}$ before message passing, and the aggregated node features $\tilde{\mathbf{X}}_\mathcal{V}$. Specifically, the three matrices are first concatenated and then fed into the MLP to compute the updated node features $\mathbf{X}'_\mathcal{V}$.

The Global Model updates the global features as follows:

- $\rho^{v\to u}:\mathbb{R}^{M\times 4\times 512}\to\mathbb{R}^{512}$ averages $\mathbf{X}'_\mathcal{V}$ over the nodes to obtain $\hat{\mathbf{X}}'_\mathcal{V}\in\mathbb{R}^{512}$.
- $\Phi^u:\mathbb{R}^{1024}\to\mathbb{R}^{512}$ processes the concatenation of the global features $\mathbf{u}(\mathcal{I}_k)$ and $\hat{\mathbf{X}}'_\mathcal{V}$ and updates the global feature vector $\mathbf{u}'(\mathcal{I}_k)\in\mathbb{R}^{512}$. This function is implemented using the same MLP structure as in Φ^v and $\Phi^{v,u}$.

Finally, the updated node $\mathbf{X}'_\mathcal{V}$ and global $\mathbf{u}'(\mathcal{I}_k)$ features are further processed by the Node Model to obtain the final node features matrix $\mathbf{X}''_\mathcal{V}$. This is fed into the temporal-correlation module (see next Section) to combine it with information from past timesteps and compute the current graph state.

3) Temporal correlation: To model temporal dependencies between consecutive observations, a graph state is generated and propagated through time. The graph state at timestep k is represented by a node feature matrix $\mathbf{Z}_k\in\mathbb{R}^{M\times 4\times 512}$, computed similarly to [34], as follows:

$$\mathbf{Z}_k=\mathrm{ReLU}\left(\mathbf{X}''_\mathcal{V}+\mathrm{GCN}\left(\mathbf{Z}_{k-1},\mathcal{E}\right)\right), \tag{2}$$

where $\mathbf{X}''_\mathcal{V}$ is the node feature matrix computed by the spatial-correlation module at timestep k, and GCN is a Graph Convolutional Network [40] that selects the information to propagate through time from the previous graph state $\mathbf{Z}_{k-1}$ (see Section III-C2 for implementation details).

4) Node prediction: The graph state at timestep k is used to compute the score $s^p_k(v_{i,j})\in\mathbb{R}$, which encodes the probability that the robot location at timestep k is at the j-th micro node of the i-th macro node. The score vector, in particular, is computed as follows:

$$\mathbf{s}^p_k=\mathrm{Softmax}\left(\mathbf{W}\mathbf{Z}_k+\mathbf{b}\right), \tag{3}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and the bias.

5) Re-ranking via geometric consistency: The best location candidates are re-ranked with a geometric consistency check by following the procedure proposed in [23]. We extract patches at multiple scales and compute VLAD descriptors [22] on the images of the top-5 candidate micro nodes $\mathcal{V}_{best}$ (according to the scores in $\mathbf{s}_k^p$), and on the current observation. The VLAD descriptors are then compared to detect mutual matches between the candidate node images and $\mathcal{I}_k$. The matches are used to perform a geometric consistency check through a RANSAC-based scoring procedure (namely GC). Alternatively, a lighter and faster version (namely s-GC) computes the score directly on the matched feature pairs without requiring sampling. The consistency scores $\mathbf{s}_k^g = s_k^g(v_{i,j})$, $\forall v_{i,j} \in \mathcal{V}_{best}$ are then multiplied by the prediction probabilities $s_k^p(v_{i,j})$, and the nodes are re-ranked accordingly to compute the final score vector $\mathbf{s}_k$.

C. Model optimization

1) Data collection for model training: To train the location model we need scenes with varying characteristics and appearance, and with ground truth 3D occupancy maps for graph initialization. Collecting these data from real scenarios is impractical and burdensome, therefore, we decide to gather training samples by exploiting the Unreal Engine (UE4)[1] simulator to generate photorealistic environments. For this purpose, we develop a procedural script[2] to build indoor scenes with randomized characteristics. Specifically, each scenario is created by first randomizing the shape and the size of rooms and corridors, and then adding walls and floor textures, objects, and furniture with random layouts and characteristics. For each generated scene, the topological graph is first built as described in Section III-A. Afterward, the training set is generated by flying an MAV simulated with the AirSim plugin [41] in the scene, collecting 512×512 images from the front-looking camera and associating them with the nearest ground truth location node. To favor generalization over different viewpoints and positions, for each scene multiple random trajectories are sampled and executed. Each one is generated by sampling a starting and an end position, and using a 3D implementation of A* to obtain a collision-free path between those points (if it exists). For model optimization, we generate 400 indoor scenes and collect a total of 156820 images with ground truth node associations.

2) Implementation and Training details: As detailed in Section III-B1, both global and node features are extracted by a CNN with a ResNet-50 backbone pre-trained on the ImageNet dataset [42]. The features extracted from the ResNet are flattened and concatenated into a 242688 element vector, which is then processed by PCA to reduce the dimensionality to 512. PCA is fitted only on the node features of the training graphs. The MLPs that implement Φ^v and $\Phi^{v,u}$ of the Node Model include a hidden layer with 512 nodes followed by a ReLU activation and a dropout layer with dropout probability set to 0.2, and an output linear layer. The latter has 128 and 512 nodes in the case of Φ^v and $\Phi^{v,u}$, respectively. The same MLP structure is used for the function Φ^u of the Global Model, with both the hidden and output layer composed of 512 nodes. The GCN in the temporal correlation module includes a single layer that outputs a vector with 512 elements. The GRNN model (*i.e.,* the spatial correlation, temporal correlation, and classification modules) and the GC module (*i.e.,* the CNN that extracts the VLAD descriptor) are trained separately. The latter is trained as detailed in [23], while the GRNN model is optimized by setting the batch size to 32 sequences, and the weight decay to 0.001. The learning rate is initially set to 0.0001 and decreased by a factor of 0.7 every 20 epochs. The training process takes approximately 8 hours and 80 epochs to converge on a workstation equipped with an NVIDIA RTX A6000 48GB.

[1]https://www.unrealengine.com/

[2]https://github.com/isarlab-department-engineering/UE4-environment-randomization(available upon acceptance of the paper)

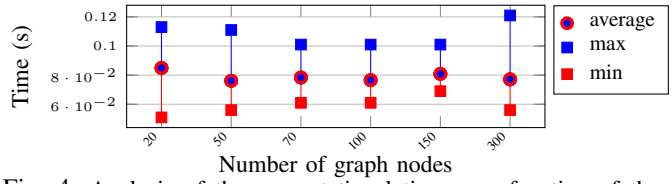
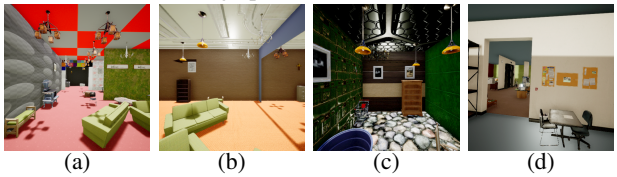


Fig. 4: Analysis of the computational time as a function of the number of nodes in the graph.



(a) (b) (c) (d)

Fig. 5: Examples of the evaluation environments: **Tr-T** set (5a); **Tr-A** set (5b); **Test** set (5c); **Test-Ph** set (5d).

3) Inference Phase: The node features $\mathbf{X}_\mathcal{V}$, their embeddings $\tilde{\mathbf{X}}_\mathcal{V}$, and the VLAD features of the graph nodes can be computed offline to reduce the inference time as they do not change. Therefore, during inference, only global features for the current observation $\mathbf{u}(\mathcal{I}_k)$ are computed. This step, which involves the spatial correlation, temporal correlation, and node prediction modules, takes approximately 0.08 ± 0.04 seconds. This computational time is nearly unaffected by the graph size, *i.e.,* our approach efficiently scales with the number of nodes. This can be observed in Figure 4, where we analyze the maximum inference time as a function of the graph size. Specifically, we considered six graphs, with an increasing number of nodes (from 20 to 300), and we perform nine runs for each set to compute average, minimum and maximum values (geometric check is not considered for this study since its execution time does not depend on the number of nodes in the graph). If also the GC is performed, the total time increases by 0.11 seconds (0.09 for the faster version) for each image candidate. Hence, if the top 5 candidates are considered for the GC, the total inference time is approximately 0.57 ± 0.08 seconds per prediction.

IV. EXPERIMENTS

A. Evaluation Environments

To assess the performance of our approach, we generate and execute additional trajectories different from those used

for training. To further evaluate the generalization capabilities, we also generate additional scenes with different characteristics with respect to the training environments. Specifically, we build four evaluation sets as follows:

Tr-T: A set of trajectories different from the training ones, but collected on the same scenes used during optimization (*i.e.,* by using the same topological graph). It contains 20 graphs and 8244 observations collected during the execution of different trajectories.

Tr-A: A set of trajectories collected on environments different from those used during training, although with a similar appearance. It contains 20 graphs and 5958 observations.

Test: A set of trajectories collected on scenes significantly different from the training ones (*i.e.,* different layouts, textures, furniture). It contains 20 graphs and 6286 observations.

Test-Ph: The last set contains 1 graph and 356 observations collected in a photorealistic office-like environment and is, therefore, the most challenging due to the significant domain shift with respect to the training scenes.

On average, the graphs of **Tr-T**, **Tr-A**, and **Test** have 70 micro nodes (± 10), while the graph of **Test-Ph** has 48 micro nodes. The topological maps for the scenes of **Tr-A**, **Test** and **Test-Ph** are built as detailed in Section III-A. Some examples of the scenes are shown in Figure 5.

B. Baselines

We consider different state-of-the-art approaches in the experimental study. All the baselines are trained on the same training set used to optimize our strategy (see Section III-C1):

GOLN [9]. This framework is based on the GNN proposed by [33] Differently from the original paper, we do not use the object-based map for the comparison since we do not assume the availability of a 3D object detector.

ST-GLN [10]. The approach based on a Graph Convolutional LSTM (GCLSTM) presented in [43]. For each frame, the framework returns the closest node in the graph.

NetVLAD [21]. This approach models the environment with a database of uncorrelated images, against which the current query image is matched to obtain the most similar ones. We train the model by using the training graph images as the database images, and the training observations collected by the MAV as the query images.

Patch-NetVLAD [23]. The improved version of [21] As for [21], we train the model by employing the training graph images as database images and training observations as query images. Since in our approach the GC is applied on the top 5 candidates, for comparison purposes we run Patch-NetVLAD by both considering all the node images and only the top-5 candidates in its GC step (namely Patch-NetVLAD-5). The lightweight version of Patch-NetVLAD is instead referred to as Patch-NetVLAD-s.

C. Evaluation metrics

At test time, our approach processes image observations and predicts the micro node in the topological graph encoding the robot location. To evaluate and compare its performance against the baselines, we introduce three metrics:

Macro Node Accuracy (MNA): the accuracy score computed at macro node level. In particular, the prediction is considered to be correct if any of the four *micro nodes* associated with the ground truth macro node is predicted.

One Hop Accuracy (OHA): the predicted node is considered correct if it corresponds to the ground truth macro node or one of its neighbors at distance one in the graph (*i.e.,* with an edge connecting them to the ground truth macro node).

Recall@N (R@N): a metric commonly used in VPR tasks [23] adapted to our setting. A prediction is considered correct if at least one of the top N predicted nodes is at most one hop distant from the ground-truth micro node in the graph map. It should be noticed that R@1 corresponds to OHA.

For each evaluation set, we first compute the MNA and OHA metrics on each environment separately and then their overall mean and standard deviation. For the R@N we show the comparative results between the proposed approach and the baselines obtained on the **Test-Ph** environment.

D. Results

1) Comparative results: In Table I we compare the Macro Node Accuracy (MNA) and the One Hop Accuracy (OHA) of the proposed approach with those of the baselines introduced in Section IV-B. For our approach, we reported the results obtained with both the RANSAC-based version (GC) and the faster version (s-GC) of the geometric consistency module described in Section III-B5. In addition, we report the results obtained without (w/o GC), where the predicted node is the one with the highest probability value returned by the GRNN.

Results show that the proposed strategy outperforms NetVLAD. As an instance, in the Test evaluation set MNA and OHA are respectively 12% and 11% higher for our approach without the geometric check than NetVLAD. If the consistency check is included, the improvement over this baseline is even more evident (15% higher on MNA and 16% on OHA). This proves that modeling spatiotemporal dependencies is beneficial for localization.

Our strategy outperforms also the other topological works, *i.e.,* GOLN [9] and ST-GLN [10], demonstrating its superiority in managing localization with MAV platforms. On the photorealistic set **Test-Ph**, the accuracy improvement with respect to these two baselines is up to 30% and 20% with respect to MNA and OHA, respectively.

Among the baselines, Patch-NetVLAD [23] is certainly the one achieving the best results, mainly thanks to its geometric consistency check. Reducing the number of candidates to be re-ranked does not have a significant impact on its performance. We can, indeed, observe a drop of only approximately 2% in the metrics when only 5 candidates are considered.

When compared with our version without the geometric consistency check, Patch-NetVLAD performance is on par and, in some cases, slightly better. However, when the geometric re-ranking is included, our approach outperforms Patch-NetVLAD in the most challenging scenarios, *i.e.,* the **Test** and the **Test-Ph** sets, which differ from the scenes used for training. In the latter set, the improvement is up to 5% with respect to both metrics. These results highlight

	Tr-T		Tr-A		Test		Test-Ph	
	MNA	OHA	MNA	OHA	MNA	OHA	MNA	OHA
GOLN [9]	0.64 ± 0.04	0.84 ± 0.04	0.49 ± 0.05	0.72 ± 0.07	0.38 ± 0.06	0.62 ± 0.08	0.43 ± 0.00	0.72 ± 0.00
ST-GLN [10]	0.72 ± 0.04	0.89 ± 0.03	0.64 ± 0.04	0.83 ± 0.05	0.51 ± 0.07	0.73 ± 0.09	0.46 ± 0.00	0.71 ± 0.00
NetVLAD [21]	0.64 ± 0.07	0.81 ± 0.08	0.66 ± 0.07	0.85 ± 0.05	0.53 ± 0.06	0.76 ± 0.05	0.59 ± 0.00	0.76 ± 0.00
Patch-NetVLAD-5 [23]	0.76 ± 0.05	0.88 ± 0.04	0.76 ± 0.06	0.90 ± 0.04	0.71 ± 0.06	0.88 ± 0.05	0.67 ± 0.00	0.85 ± 0.00
Patch-NetVLAD-s [23]	0.77 ± 0.04	0.89 ± 0.02	0.75 ± 0.06	0.90 ± 0.03	0.70 ± 0.05	0.87 ± 0.04	0.68 ± 0.00	0.87 ± 0.00
Patch-NetVLAD [23]	0.78 ± 0.04	0.89 ± 0.04	$\mathbf{0.77} \pm 0.06$	0.90 ± 0.04	0.77 ± 0.04	0.91 ± 0.03	0.69 ± 0.00	0.87 ± 0.00
Ours (w/o GC)	0.84 ± 0.03	0.96 ± 0.02	0.75 ± 0.05	0.93 ± 0.04	0.65 ± 0.07	0.87 ± 0.07	0.58 ± 0.00	0.85 ± 0.00
Ours (w/ s-GC)	0.85 ± 0.02	0.97 ± 0.01	$\mathbf{0.77} \pm 0.01$	$\mathbf{0.94} \pm 0.01$	0.71 ± 0.01	0.89 ± 0.02	0.73 ± 0.00	0.91 ± 0.00
Ours (w/ GC)	$\mathbf{0.86} \pm 0.03$	$\mathbf{0.97} \pm 0.02$	0.76 ± 0.05	$\mathbf{0.94} \pm 0.04$	$\mathbf{0.79} \pm 0.05$	$\mathbf{0.94} \pm 0.04$	$\mathbf{0.74} \pm 0.00$	$\mathbf{0.92} \pm 0.00$

TABLE I: The Table reports mean and standard deviation for MNA and OHA computed on the environments of each evaluation set.

MLF	TC	GC	Tr-T		Tr-A		Test		Test-Ph	
			MNA	OHA	MNA	OHA	MNA	OHA	MNA	OHA
-	-	-	0.64 ± 0.04	0.84 ± 0.04	0.49 ± 0.05	0.72 ± 0.07	0.38 ± 0.06	0.62 ± 0.08	0.43 ± 0.00	0.72 ± 0.00
-	✓	-	0.73 ± 0.05	0.92 ± 0.04	0.61 ± 0.05	0.88 ± 0.06	0.47 ± 0.07	0.80 ± 0.08	0.42 ± 0.00	0.81 ± 0.00
✓	-	-	0.80 ± 0.04	0.94 ± 0.02	0.64 ± 0.05	0.85 ± 0.04	0.51 ± 0.07	0.77 ± 0.06	0.47 ± 0.00	0.72 ± 0.00
✓	✓	-	0.84 ± 0.03	0.96 ± 0.02	0.75 ± 0.05	0.93 ± 0.04	0.65 ± 0.07	0.87 ± 0.07	0.58 ± 0.00	0.85 ± 0.00
✓	✓	✓	$\mathbf{0.86} \pm 0.03$	$\mathbf{0.97} \pm 0.02$	$\mathbf{0.76} \pm 0.05$	$\mathbf{0.94} \pm 0.04$	$\mathbf{0.79} \pm 0.05$	$\mathbf{0.94} \pm 0.04$	$\mathbf{0.74} \pm 0.00$	$\mathbf{0.92} \pm 0.00$

TABLE II: Ablation study on the main components of the proposed approach. Check marks (✓) indicate if a particular component has been enabled. For both the metrics, we report the mean and the standard deviation computed on the environments of each evaluation set.

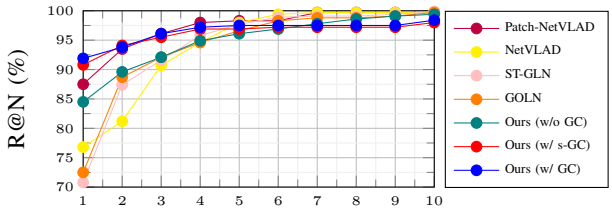


Fig. 6: Comparison of the R@N metric on the Test-Ph environment.

two important aspects. First, they show the generalization capabilities of the strategy we propose. Second, since our GC is similar to that of Patch-NetVLAD, this demonstrates that the node probability scores computed by our GRNN model are more accurate than those of the baseline.

Figure 6 shows the R@N performance of our approach compared to the state-of-the-art, considering the top N predicted nodes (from 1 to 10). Our approach (with GC included) outperforms the baselines for $N < 4$, proving that it requires few candidates to correctly predict the location.

2) Ablation study: We also perform an ablation study to show the impact of the components of the proposed framework on the performance. Specifically, we analyze the effect of the following components:

Multi-level Features (MLF): fusing multi-level information during the feature extraction phase (Section III-B1);

Temporal Correlation (TC): using the temporal correlation module (Section III-B3);

Geometric Consistency (GC): re-ranking the predictions through the geometric consistency check (Section III-B5).

The quantitative results are reported in Table II. As expected, a large performance drop is observed when the three components are absent. An accuracy gain (up to 12% and 18% on MNA and OHA, respectively) is experienced when the TC module is included. Modeling temporal dependencies is, indeed, crucial to improve the robustness to perceptual aliasing. This, in practice, allows the approach to avoid predicting nodes that exhibit a similar appearance to the current image but are inconsistent with past observations.

The improvement is, in some cases, even more significant when the MLF module is added. This proves that aggregating features from different layers of ResNet is beneficial for the localization task since a richer semantic representation of the scene is provided to the spatial and temporal modules. When both MLF and TC modules are enabled, the accuracy is further improved (up to 18% on MNA and 13% on OHA with respect to the versions that use them separately).

Results indicate that incorporating the Geometric Consistency (GC) module is particularly effective in environments that differ from the training data, such as Tr-A and Test. This is expected due to domain shifts and perceptual aliasing issues, which can cause the node with the highest estimated score to be incorrect. The GC module can re-rank nodes based on geometric consistency, filtering out spurious associations and improving performance.

E. Discussion and limitations

Backbones of GRNN and GC: the final setup proposed encompasses a combination of two different backbones for the features extraction in the GRNN network and that used to compute VLAD, respectively ResNet-50 and VGG-16. We also investigated the use of the same backbone (either ResNet or VGG) for both the feature computations to reduce the execution time during inference. However, this choice resulted in a performance drop and, therefore, we opted for the first strategy (ResNet+VGG), as it maximizes the trade-off between accuracy and time.

Deployment on real MAVs: reducing the computational cost is a major aspect to deploy our approach on a real platform. Low-level optimizations (*e.g.,* TensorRT) and distillation approaches to reduce the execution time of the deep networks without affecting the localization accuracy can be considered. In addition, different configurations that combine onboard computations (*e.g.,* on Nvidia NX boards) and offboard ones on dedicated workstations will be investigated, also considering the roundtrip delay induced by the communication.

Generalization on real environments: an important challenge lies in the domain gap between simulated and real-world images. Besides fine-tuning with real-world images, we are considering different approaches to address this problem and maximize the generalization capabilities of our network.

One approach that is under investigation, is to take advantage of Generative Adversarial Networks (GANs) to reduce the gap between the two domains.

Pre-built graph: Our approach performs localization on a graph that is pre-built offline and does not change during the inference phase. This assumption could be removed by extending our solution with online graph building strategies that build and update the topological map as the robot navigates in the scene.

V. CONCLUSIONS

In this work, we propose a vision-based topological approach to localize autonomous MAVs in a pre-built topological map. Spatiotemporal dependencies between location and consecutive observations are modeled by a graph recurrent neural network, and its estimates are refined by a consistency module that re-ranks the top node candidates according to a geometric check. Experiments show that our strategy outperforms other topological localization and visual place recognition baselines, and is able to generalize on environments that differ from those used during the training phase.

REFERENCES

[1] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *ICRA*, 2018.

[2] M. Legittimo, S. Felicioni, F. Bagni, A. Tagliavini, A. Dionigi, F. Gatti, M. Verucchi, G. Costante, and M. Bertogna, "A benchmark analysis of data-driven and geometric approaches for robot ego-motion estimation," *Journal of Field Robotics*, pp. 1–29, 2023.

[3] E. C. Tolman, "Cognitive maps in rats and men," in *Psychological Review*, vol. 55(4), 1948, pp. 770–778.

[4] Z. Hong, Y. Petillot, D. Lane, Y. Miao, and S. Wang, "TextPlace: Visual place recognition and topological localization through reading scene texts," in *ICCV*, 2019, pp. 2861–2870.

[5] X. Zhang, L. Wang, and Y. Su, "Visual place recognition: A survey from deep learning perspective," *Pattern Recognit.*, vol. 113, p. 107760, 2021.

[6] S. Garg, T. Fischer, and M. Milford, "Where is your place, visual place recognition?" in *IJCAI*, 2021.

[7] T. Barros, R. Pereira, L. Garrote, C. Premebida, and U. J. Nunes, "Place recognition survey: An update on deep learning approaches," *ArXiv*, vol. abs/2106.10458, 2021.

[8] K. Chen, J. P. de Vicente, G. Sepulveda, F. Xia, A. Soto, M. Vazquez, and S. Savarese, "A behavioral approach to visual navigation with graph localization networks," in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, 2019.

[9] S. Felicioni, M. Legittimo, M. L. Fravolini, and G. Costante, "GOLN: Graph object-based localization network," in *ICAR*, 2021, pp. 849–856.

[10] T. Niwa, S. Taguchi, and N. Hirose, "Spatio-temporal graph localization networks for image-based navigation," in *IROS*, 2022.

[11] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. PAMI*, vol. 40, no. 3, pp. 611–625, 2017.

[12] C. Campos, R. Elvira, J. J. Gómez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[13] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," *Proc. of the IEEE International Conference on Robotics and Automation*, 2020.

[14] G. Clement, M. A. Oisin, and J. B. Gabriel, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017.

[15] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017.

[16] G. Costante and M. Mancini, "Uncertainty estimation for data-driven visual odometry," *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1738–1757, 2020.

[17] K. Tsintotas, L. Bampis, and A. Gasteratos, "The revisiting problem in simultaneous localization and mapping: A survey on visual loop closure detection," in *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[18] F. Maffra, Z. Chen, and M. Chli, "Viewpoint-tolerant place recognition combining 2d and 3d information for uav navigation," in *ICRA*, 2018.

[19] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *CVPR*, 2020.

[20] R. R. Wiyatno, A. Xu, and L. Paull, "Lifelong topological visual navigation," *IEEE Robotics and Automation Letters*, vol. 7, pp. 9271–9278, 2021.

[21] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *CVPR*, 2016.

[22] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010.

[23] S. Hausler, S. Garg, M. Xu, M. Milford, and T. Fischer, "Patch-NetVLAD: Multi-scale fusion of locally-global descriptors for place recognition," in *CVPR*, 2021.

[24] E. Garcia-Fidalgo and A. Ortiz, "Vision-based topological mapping and localization methods: A survey," *Robotics and Autonomous Systems*, vol. 64, pp. 1–20, 2015.

[25] I. Ulrich and I. Nourbakhsh, "Appearance-based place recognition for topological localization," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 1023–1029.

[26] B. Kuipers, "Modeling spatial knowledge," *Cognitive Science*, vol. 2, no. 2, pp. 129–153, 1978.

[27] R. Mereu, G. Trivigno, G. Berton, C. Masone, and B. Caputo, "Learning sequential descriptors for sequence-based visual place recognition," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 383–10 390, 2022.

[28] F. Lu, B. Chen, X.-D. Zhou, and D. Song, "Sta-vpr: Spatio-temporal alignment for visual place recognition," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4297–4304, 2021.

[29] K. Tsintotas, L. Bampis, and A. Gasteratos, "Assigning visual words to places for loop closure detection," 2018, pp. 5979–5985.

[30] ——, "Probabilistic appearance-based place recognition through bag of tracked words," vol. 4, no. 2, 2019, pp. 1737–1744.

[31] ——, "Modest-vocabulary loop-closure detection with incremental bag of tracked words," vol. 141, 2021, p. 103782.

[32] F. Xue, X. Wu, S. Cai, and J. Wang, "Learning multi-view camera relocalization with graph neural networks," in *CVPR*, 2020.

[33] P. Battaglia, J. B. C. Hamrick, V. Bapst, A. Sanchez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. J. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *arXiv*, 2018.

[34] L. Ruiz, F. Gama, and A. Ribeiro, "Gated graph recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6303–6318, 2020.

[35] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3D topological graphs for micro-aerial vehicle planning," *IROS*, 2018.

[36] T. Lee, R. Kashyap, and C. Chu, "Building skeleton models via 3-D medial surface axis thinning algorithms," *Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994.

[37] H. Homann, "Implementation of a 3D thinning algorithm," *The Insight Journal*, 2007.

[38] A. Bojko, R. Dupont, M. Tamaazousti, and H. Le Borgne, "Self-improving SLAM in dynamic environments: Learning when to mask," in *BMVC*, 2022.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[40] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-Supervised Learning With Graph Learning-Convolutional Networks," in *CVPR*, 2019.

[41] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[43] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured Sequence Modeling with Graph Convolutional Recurrent Networks," *ICONIP*, p. 362–373, 2018.