LEARNING MAPS FOR OBJECT LOCALIZATION USING VISUAL-INERTIAL ODOMETRY

B. Zha[1,*], A. Yilmaz[1]

[1] Dept. of Civil, Environmental and Geodetic Engineering, The Ohio State University, 470 Hitchcok Hall, Columbus, Ohio, USA - (zha.44, yilmaz.15)@osu.edu

Commission I, WG I/9

KEY WORDS: Localization, Deep Learning, Motion Trajectory, Visual Inertial Odometry, OpenStreetMaps

ABSTRACT:

Objects follow designated path on maps, such as vehicles travelling on a road. This observation signifies topological representation of objects' motion on the map. Considering the position of object is unknown initially, as it traverses the map by moving and turning, the spatial uncertainty of its whereabouts reduces to a single location as the motion trajectory would fit only to a certain map trajectory. Inspired by this observation, we propose a novel end-to-end localization approach based on topological maps that exploits the object motion and learning the map using an recurrent neural network (RNN) model. The core of the proposed method is to learn potential motion patterns from the map and perform trajectory classification in the map's edge-space. Two different trajectory representations, namely angle representation and augmented angle representation (incorporates distance traversed) are considered and an RNN is trained from the map for each representation to compare their performances. The localization accuracy in the tested map for the angle and augmented angle representations are 90.43% and 96.22% respectively. The results from the actual visual-inertial odometry have shown that the proposed approach is able to learn the map and localize objects based on their motion.

1. INTRODUCTION

Without the loss of generality, object localization can be defined as finding *where an object is* on a map given sensory data. Object localization is important and necessary for not only core applications, like robotics and autonomous vehicles, but also tracking of pedestrians for surveillance or health reasons. Despite advances in the field, localization is still a challenging problem especially when Global Positioning System (GPS) data is contested, such as degraded or not available.

The most popular localization technology for the outdoor environment is GPS, which utilizes georeferenced satellite constellation to estimate the object's location (Hofmann-Wellenhof et al., 2012). However, the limitations are also notable: GPS is not accurate for especially civilian and consumer applications, and the signal may be unavailable or unreliable in serveral areas, such as underground, in tunnels, indoors and urban canyons. To resolve these limitations, researchers have proposed Indoor Positioning Systems (IPS) (Mautz, 2012) to localize objects using installed infrastructure, such as WiFi, Bluetooth, Ultra Wide Band (UWB), etc. Despite being a fast growing research area, IPS technology using external signals requires large investment and has high maintenance cost.

In addition to the GPS and IPS methods, other alternatives have also been developed in the past few years (Sattler et al., 2011, Gupta et al., 2016). One typical alternative approach is Simultaneous Localization and Mapping (SLAM) (Cadena et al., 2016), which constructs the map of an unknown environment while simultaneously keeping track of the object's location on the map that is being generated. Many research groups have developed SLAM variations using different sensors, such as laser-altimetry (Hess et al., 2016), vision-based (Mur-Artal et al., 2015) or their combinations (Qin et al., 2018). Despite the developments in the literature, there is still no viable solution for mapping and localization problem due to dead reckoning that causes constant *drift* and has a high computational costs.

A major category of work in literature is dedicated to the use of images which can be classified into: photogrammetric localization (Sattler et al., 2011) and image-matching based localization (Walch et al., 2017). Photogrammetric localization assumes the scene is represented by a texture mapped 3D model (potentially generated from structure-from-motion). The model is used to estimate the extrinsic camera parameters from the query image (position and orientation). Essentially, this method uses a photogrammetric pipeline to match 2D and 3D features to triangulate camera parameters. Despite their performance, photogrammetric pipeline is both computationally expensive – especially to perform matching 2D and 3D features when initial posi-tion is unknown– and generate and access texture mapped 3D point clouds. Image-matching based localization uses large-scale geotagged image collection which converts localization task to image retrieval task (Li et al., 2014). Given an input image, this method returns latitude and longitude of matched images which can be used for triangulating position of input image. Image-matching based approaches require large databases of images, and building, maintaining and matching against such a large image database can be very hard.

Recently, a number of studies started to use deep learning (DL) (Engel et al., 2019) or OpenStreetMap (OSM) (Brubaker et al., 2015) to deal with self-localization problem. These methods, however, either formulate localization as pose regression problem using DL or exploit an existing map to pre-build a probabilistic graph structure.

In this paper, we introduce a novel approach that introduces the concept of learning maps for object localization. Our approach

*Corresponding author

uses map transport layer, in particular the OSM, and object motion trajectory. Unlike SLAM methods that incrementally build and update the map, our approach utilizes an existing map in the form of a graph and estimate the object position swiftly and efficiently matching the motion trajectory to the learned map.

The remainder of the paper is organized as follows. In the Section 1.1, we introduce the motivation of the proposed method. Some related works are summarized in the Section 2. In Section 3, the localization methodology is discussed in detail. The experiments and results are provided in Section 4. Last but not least, we conclude the paper in the Section 5.

1.1 Motivation

Humans are exceptionally good at finding *where they are* based on observations and a simple mind-map (Sargolini et al., 2006). A typical approach humans use to achieve this goal is to match a landmark to known landmarks we learned when we were in the same location before. Moreover, other stimuli also helps us to achieve this goal, for instance a blind person can find his location based on the distance he traversed and corners he turned. Another example would be getting directions for an unknown location from others, e.g. "to get to the grocery, go straight until you see the hospital and turn left". Based on these observations, we hypothesize that one can 1) develop an algorithm to learn a map, and 2) based on the motion patterns localize the object. We illustrate this in the Figure 1, the left image is map with a transport layer containing as many trajectories as one can traverse (one sample is below the map) and the right image shows the only location that the object can be in from among many candidates. This localization treatment can be considered as graph isomorphism where a subgraph is matched to a graph. Another localization treatment, yet a simpler one, has been explored in the navigation literature (map-matching) where a noisy GPS input is matched to a road on a map. We should note however that both these methods have limitations and shortcomings. The graph isomorphism is an NP complete problem and the map matching only works when external GPS input is provided. In this paper, we take a different approach and exploit neural networks to learn the map by considering potential traverses one can make on the map.

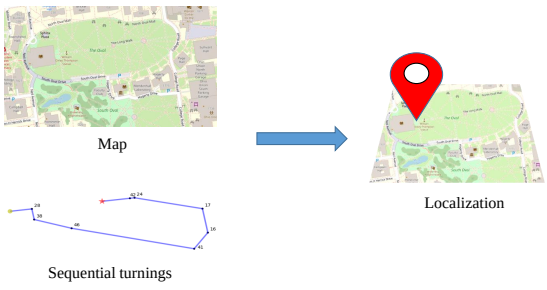


Figure 1. The concept of localization using a map and a trajectory. The motion generating the observed trajectory can only have one matching location if the shape of the trajectory is unique and matches the map road network. A trajectory can be unique if it is long enough (the map shown is from OSM).

2. RELATED WORK

Algorithmic approaches for localization have been long researched and is still open problem especially when there is lack of external information about the object's location. Below, we summarize several methods that are representative and are related to ours.

Probabilistic Localization Probabilistic localization is currently an active field of research. These methods are designed to estimate the position of an object with associated uncertainty. In an earlier work, (Fox et al., 1999) presented a Markovian method to model the posterior distribution of an object's position given its pose. Their method would work only in a small map and cannot be scaled to larger maps. For interested readers, a more comprehensive reference about probabilistic approaches is given in (Thrun et al., 2005). In contrast to the cited and many other methods in this category, our approach is scalable from small to large maps.

Visual Odometry & SLAM Visual Odometry (VO) (Scaramuzza, Fraundorfer, (Scaramuzza et al., 2011)) is a dead-reckoning approach where the position is incrementally estimated by using the object's past position and its relative motion generated from an image sequence. These methods have typically good performance on public datasets, such as KITTI; however, they suffer from *drift* problem which accumulates over time due to unknown scale unless a geo-tagged features are used to correct the drift. In order to mitigate the drift problem, most methods integrate other sensory data observed from, for instance, inertial sensors, magnetometers and range sensors (Li et al., 2014). We should note that VO based methods do not provide geolocation unless initial position and direction of motion are known. Different from VO, SLAM based methods jointly optimize relatively known landmark positions and objects position and may require loop closures during traverse to overcome *drift* problem (Cadena et al., 2016). In our approach, we use VO integrated with inertial sensor data to generate object trajectory and do not require initial position or heading direction. Using this as input, our approach predicts the location of the object on the map without loop closure requirements. Our approach can mitigate drift problems due to the trajectory representation used which will be discussed later.

Visual Localization & Deep Localization The visual localization problem is to estimate the six degrees of freedom (DoF) of an object by matching an image against a 3D scene model (Sattler et al., 2011). A variation to this approach uses landmark recognition (Zamir et al., 2014). The main step in these approaches is to efficiently represent a database and then match a query image to this database. Although these methods have good performances, building and maintaining a large georeferenced image database is neither trivial nor scalable. Furthermore, image matching is an expensive operation for real-time localization applications. Recently, deep learning has been applied in localization problem which is referred to as deep localization. (Chen et al., 2019) proposed a behavioral approach to localization and navigation using several deep neural networks. The behavior in their approach refers to actions taken at a node, such as turn left, turn right and go straight. (Amini et al., 2019) defined a variational network to conduct vehicle localization and navigation which integrated probabilistic uncertainty and deterministic control.

Map Matching Most methods, including the commercial solutions, use *map matching* to map an external position signal, such as GPS or UWB, to a road network (Newson, Krumm, 2009). The methods in this category are typically based on Hidden Markov Models (HMM) and are able to localize the object by matching the GPS trajectory to the road graph. They ex-

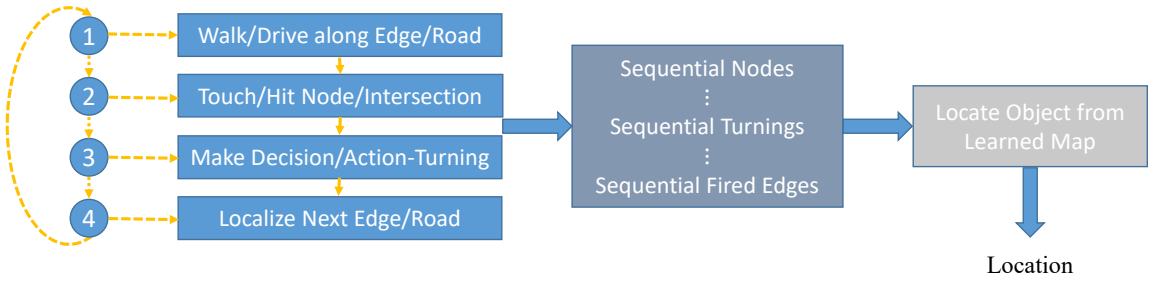


Figure 2. The pipeline for generating a sequence of motion features that is used in learning the map and retrieving the object location. The last box in the figure is the learned map using RNN which provides object location.

plicitly require external signal and cannot be scalable to cases when these signals are not available.

Others Our work is closely related to another body of work (Brubaker et al., 2015, Gupta et al., 2016, Zha et al., 2019) which use a map and a trajectory to perform the geolocalizatoin. (Brubaker et al., 2015) developed a probabilistic localization method using OSM and VO. (Gupta et al., 2016) presented an object localization approach based on stochastic trajectory matching using a brute-force location search which is time consuming in large maps. In (Wei et al., 2019), a sequence to sequence labeling method for trajectory matching using neural machine translation network is proposed. This approach, however, was shown to work well on synthetic scenarios where the input trajectory had no errors, while in reality all dead reckoning methods suffer from drift problems. In another paper, authors of (Zha et al., 2019) used similar ideas to (Wei et al., 2019) and showed their performance on synthetic trajectories. In this study, we propose an RNN based map learning approach that is resilient to errors in object trajectory that are generated from visual and inertial sensors.

3. RECURRENT NEURAL NETWORK LOCALIZATION

The proposed approach performs localization based on motion trajectory learned using an RNN. The pipeline for the proposed system consists of several key components, including map representation, trajectory generation, trajectory representation, neural network architecture and map learning. We represent the map as a graph with *nodes* and *edges*. In our application, the map is obtained from OSM. An object moving the scene is assumed to be either moving on an *edge* or has arrived to a *node*: this information is used to generate a sequence of motion features as shown in Figure 2. The nodes in this representation relate to making turns or moving a straight direction. The object location can be determined from this sequence by running it through the RNN which generates position candidates on the topological map. In the following discussion, we describe each of these components in more detail.

3.1 Problem Statement

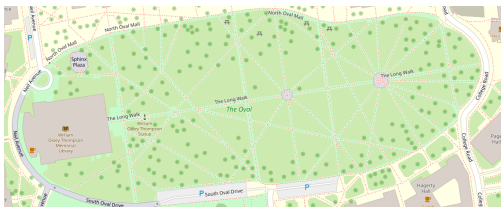
Let a map be represented as a graph G with nodes V and edges E, where the nodes and edges have unique IDs. Given an object trajectory −we use fusion of visual and inertial sensors to generate the trajectory−, a motion feature sequence can be generated by quantizing turning angles and the distances traversed. Using this sequence, the position of the object is defined as the edge connected to the last node visited. Therefore, the object localization problem becomes a variable-length sequential data classification problem:

$$f : X \rightarrow Y \tag{1}$$

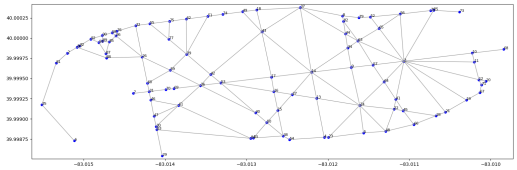
- Input feature sequence: $X = (\psi_1, \psi_2, \ldots, \psi_n)$, $\psi_i \in R$, $(i = 1, 2, \ldots, n)$, n is the length of sequence and ψ_i is i^{th} angle;
- Output position label: $Y = e_i$, $e_i \in E$, where $E = \{e_1, e_2, \ldots, e_k\}$ is the output label space, k is the number of edges in the topological map;
- Function: f is the trained RNN that learns the map.

3.2 OSM and Map Representation

OSM is a crowdsourced geographic database that provides free and editable map data under the Open Database License. The OSM data as shown in the Figure 3a consists of three basic elements: *nodes*, *ways* which we refer to as series of edges and *relations*. Each *node* contains its GPS coordinates (latitude and longitude) as meta data and a unique ID. The *way* is formatted as a collection of connected nodes forming edges and also has a unique ID. Relations are used to represent relationships between nodes and ways.



(a) OSM layers



(b) Topological road map

Figure 3. The OSM data and the graph based road representation used in our approach. Each node contains latitude and longitude meta data.

We convert the OSM data to an undirected graph as shown in the Figure 3b where the nodes copy the meta data from the OSM nodes and the edges define linear road segments. Using

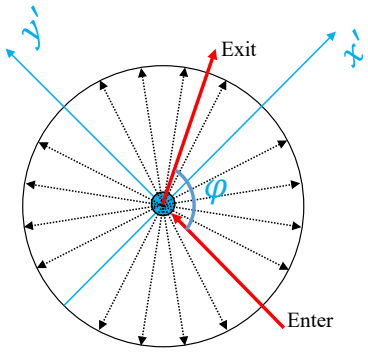


Figure 4. Local coordinate system for angle computation and quantization into discrete angle representation. The illustrated figure uses 20 bins.

the node position data, we assign each edge the length information which will be used to generate the augmented angle based trajectory representation as discussed in Section 3.3. We use this representation to generate a large number of training trajectories with varying lengths. Note that, the OSM data used in our paper contains the route type that specifies the a *way* as a "walkable" or a "drivable" way. Without the loss of generality, we only demonstrate our results on the "walkable" routes.

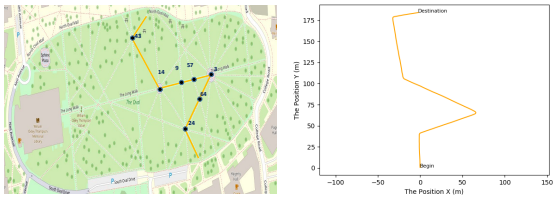
3.3 Trajectory Representation

The motion trajectory plays a critical role in our method. In this section, we will describe two sensor and scale independent trajectory representations that exploit turning angles: $\{\psi_1, \ldots, \psi_n\}$. We should note that the synthetic trajectories used for training the RNN and the real trajectory acquired from a mobile platform are not the same due to noisy data. Therefore, we also introduce an augmented representation in the form of an augmented angle sequence which resolves the problems related to noisy data. The augmented trajectory is generated by inserting virtual nodes in a uniform sample distance into a road segment between two nodes that have significant turns computed from the edges connected to them. These virtual nodes introduce additional turning angles, $\beta_i = 180°$. In this new formation, the augmented representation becomes $\{\psi_1, \beta_\mathbf{1}, \psi_2, \beta_\mathbf{2}, \beta_\mathbf{3}, \ldots, \psi_n\}$.

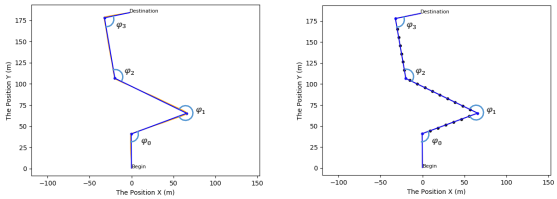
The turning angles are encoded in a local coordinate system centered at the node as shown in the Figure 4. In order to uniquely define the angles, a local coordinate system is generated at the entry direction of the node pointing in the turning direction. This representation guarantees turning $30°$ left from turning $30°$ right are unique. Given a sequence of traversed augmented nodes, $\{n_{i-1}, n_i, n_{i+1}\}$, the turning angle is computed by $\theta = \arccos \frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$ where $\mathbf{a} = n_i - n_{i-1}$ and $\mathbf{b} = n_{i+1} - n_i$. Note that, the basis vectors of the coordinate system are $(\mathbf{a}, \mathbf{c}) : \mathbf{a} \perp \mathbf{c}$. Using the basis vectors and projecting $\mathbf{b}$ onto the basis vectors:

$$\begin{aligned} proj_y &= \mathbf{b} \cdot \mathbf{a}, \\ proj_x &= \mathbf{b} \cdot \mathbf{c}, \end{aligned} \tag{2}$$

provide the sign of the turning angle. Using this formulation, a trajectory with n points results in $n-2$ turning angles $\{\psi_1, \ldots, \psi_{n-2}\}$ as shown in the left illustration in Figure 5a where seven angles are computed from a randomly generated trajectory with nine nodes. These turning angles are then quantized into 20 bins, such that the representation becomes



(a) Synthetic trajectory (for training) and real trajectory from VO (for testing).



(b) Respective trajectory representations for the trajectories above.

Figure 5. Illustrated trajectory representation, including synthetic trajectory and real trajectories and respective representations generated for the synthetic and real trajectory.

$\{\psi'_1, \ldots, \psi'_{n-2}\}$. Note that, the choice of 20 bins is dependent on the complexity of the road network and noise present in the motion trajectory.

The approach for generating the two representations described above works only with the map data because the nodes on the map are known; and the synthetic trajectories generated from the map can be used training the RNN. Real trajectory (right illustration in Figure 5a), however, does not contain nodes. In a real trajectory, we conjecture that the nodes occur at turning angles. In order to overcome this, we estimate large turning angles (see left illustration in Figure 5b) and use them as nodes for the given trajectory. The sequence of angles at these nodes can be used to represent the trajectory. For augmented angle representation of real trajectory, the same pipeline can still be applied due to the property of visual inertial odometry trajectory which contains absolute metric information.

3.4 RNN based Localization

Both proposed trajectory representations contain a discrete sequence that enable us to use a deep sequential model in the form of an RNN (Elman, 1990) as shown in Figure 6. RNN is composed of layers of neurons that have feedback loops and have the ability to process transient sequences. In our problem, this architecture models the dependencies between the map nodes, hence preserves the transition of state between consecutive time steps. In the most general form, the RNN is the function of time t that takes the current input x_t, the previous hidden state h_{t-1} and produces a new state through a non-linear activation function f and g:

$$\begin{aligned} h_t &= f(W \cdot x_t + U \cdot h_{t-1} + b_h), \\ y_t &= g(V \cdot h_t + b_y). \end{aligned} \tag{3}$$

where, U, V, W are weight matrices. h_t and y_t are hidden output and final output, and b_h and b_y are bias terms. RNN processes the input sequence one instance at a time and the output sequence generated at each time instant would depend on all previous inputs. This characteristics also has an important

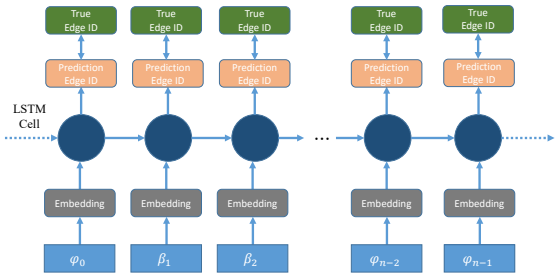


Figure 6. RNN with multiple outputs.

implication that is suitable for our object localization problem: RNN is capable of approximating arbitrary sizes of sequences (Hammer, 2000).

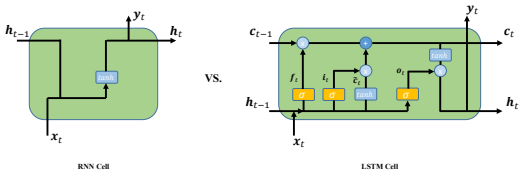


Figure 7. Structure of an RNN Cell and an LSTM Cell noting the differences between the two.

As illustrated in Figure 6, RNN network is not only cyclic but deep which makes training a difficult problem as it causes *gradient vanishing* and *gradient explosion*, even when special algorithms such as *backpropogation through time* (BPTT) are used (Werbos et al., 1990). One solution that overcomes these shortcomings is the Long Short-Term Memory (LSTM) network (Hochreiter, Schmidhuber, 1997) which uses internal cyclic mechanisms called gates that overcome gradient vanishing problem. These gates illustrated in Figure 7 include *forget gate* f_t, *input gate* i_t and *output gate* o_t by using

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ \tilde{c}_t &= tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot tanh(c_t). \end{aligned} \tag{4}$$

where, x_t is the input in time t. W and b represent weight matrices and bias terms for each gate. c_t and h_t are the LSTM cell output. $\odot$ denotes element-wise product. In this paper, due to their improved performance we adopt LSTM to learn maps and localize objects.

Trajectory localization problem can be thought of having three DoF (x, y, z). In our approach this complexity is reduced into one DoF due to the use of a sequence of turning angles at nodes. Each angle in the sequence generated by the preprocessing phase is fed into the LSTM network one by one as shown in Figure 6. An embedding layer is attached to input layer as a high-dimensional representation of discrete scalar input. The output in each time step generates the edge on which the object makes the last significant turn. We employ a *softmax* function:

$$P(Y = i|z) = softmax(z) = \frac{e^z}{\sum_{j=0}^{k} e^z} \tag{5}$$

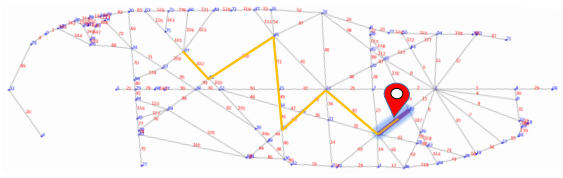


Figure 8. Localization example. The highlighted edge is the estimated position with the highest probability.

to calculate the probability of each output class which corresponds to unique edge id in a given map graph, where, z is the final linear output. $Y = i$ represents the edge ID which is equal to i. This equation shows the probability of the each edge with the final output. Figure 8 illustrates an example output sequence where the largest probability is observed at edge with ID 64 when the entire trajectory is used as input. This edge corresponds to the final estimated location.

4. EXPERIMENTS

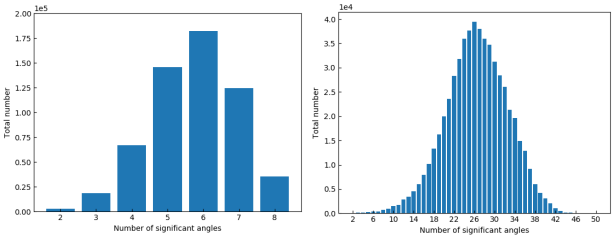
In this section, we evaluate the performance of our approach using real trajectory data generated by visual-inertial sensors on a mobile phone. First, we describe how the synthetic training set is generated and then present the details of the network training step.

4.1 Training Dataset

For supervised approaches, training dataset inarguably the most important piece in learning the parameters of a neural network. A large-scale, balanced and high-quality training data will provide a robust network that can generalize well to the testing data. In our problem, the input trajectory and output location labels of the training dataset need be generated simultaneously. For generating trajectories with location labels from the map that is represented as a graph, we use a modified depth-first search algorithm (Robert, 2002). Given a source node and target node, depth-first search generates all paths with no repeated edges. For limiting combinatorial explosion in dataset generation, we limit maximum number of nodes between source and destination to ten nodes. Given the node sequence, algorithm first generates trajectory and then the angle sequence computed using the trajectory. In Figure 9, we show the number of trajectories in the training set as a function of target node id first by filtering trajectories with small number of nodes and after inserting virtual nodes. We should note that before inserting virtual nodes the lengths of filtered trajectory are between 2 and 8 and after augmenting the trajectory the length can be up to 48. To balance the training set, we only select 34 output classes where each class contains 1000 trajectory samples in the training set as shown in the Figure 10. More detailed statistics of training set is given in the Table 1.

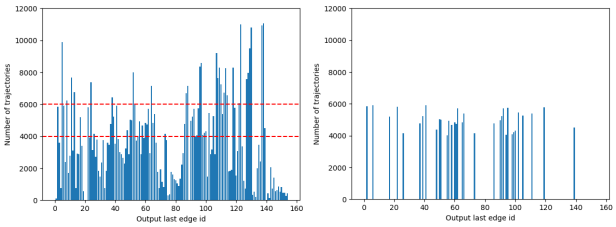
Data	Statistics
Topological Map	91 nodes, 155 edges
Trajectory Length	10 nodes
All Trajectories	576847
All Classes	153
Input Feature Space	20
Training Trajectories	34000
Training Output Classes	34

Table 1. The original dataset statistics.



(a) Filtered angle case (b) Augmented angle case

Figure 9. Number of angles in the original trajectory after filtering small angles and the augmented trajectory after inserting virtual nodes.



(a) Original dataset (b) Training dataset

Figure 10. Number of trajectories with different output edge id in original and training dataset.

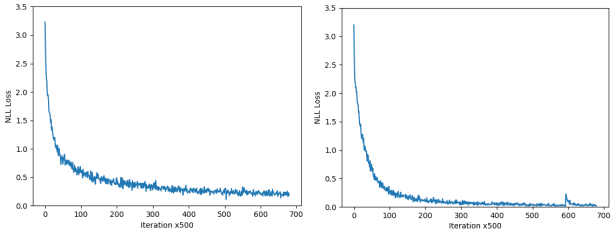
4.2 Training the Network

The training process for the proposed LSTM model using the synthetic data discussed above is performed on NVIDIA TITAN V GPU. Considering that learning map is a multi-class classification problem, the cross-entropy function is used as the objective function for measuring the training performance. The optimization algorithm is chosen as the stochastic gradient descent.

The hyperparameter settings are as follows: we used only one layer LSTM connecting a linear layer. The size of hidden state is set to 128 and learning factor is set to 0.01. The weights are initialized as zero. We used batch size of one trajectory, and the total number of iterations is set to 10 times the size of training data. At each iteration the batch is generated randomly. The benefit of using batch size of one is to eliminate padding operation and have the ability to learn from variable length trajectories. For comparing the different motion representations, namely angle and augmented angle, we generated two separate training sets (the augmented-angle sequence replaces the angle sequence and all else is the same) and trained two identical networks, one for each representation. The convergence of training phase for the angle representation are shown in the Figure 11 and Figure 12. The loss in the augmented angle representation, as shown in the Figure 11b, is decreasing faster and is closer to zero. Correspondingly, the confusion matrix in the second case shows that the augmented angle representation is more accurate. The classification accuracy for the two cases are given as 90.43% and 96.22% respectively.

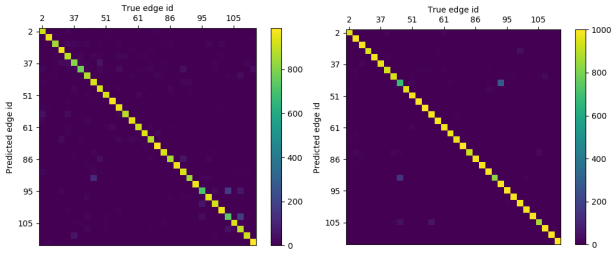
4.3 Testing Dataset

The validation of our approach is performed using real trajectories generated by visual inertial odometry (VIO) applied on data captured by a mobile phone. We should note that VIO performs dead-reckoning and does not only produce a trajectory with the right scale, in addition, the areas with relatively poor texture introduces more noise to the trajectory.



(a) Angle case (b) Augmented angle case

Figure 11. Number of angles in original and augmented angle representation.



(a) Filtered case (90.43%) (b) Augmented case (96.22%)

Figure 12. Confusion matrix

4.3.1 VIO Trajectory Generation Visual inertial odometry is the process of estimating the pose (position and orientation) of the object using both a camera and an inertial measurement unit (IMU). For data collection, we used the open-source MARS Logger (Hua et al., 2019) which is a mobile smartphone application developed for collecting synchronized video and IMU (3 acceleration and 3 angular velocity) data from mobile device. As the Figure 13 shows, an iPhone device is used for collecting the data. The sensor fusion is performed by open-source VINS-Mono approach (Qin et al., 2018) that provides high-accuracy visual-inertial odometry in the form of three rotation and three position data with absolute scale information. The starting position for all collected trajectories are set to zero. In Figure 14, five different trajectories with different number of significant turning angles ranging from two to six are collected by different people.

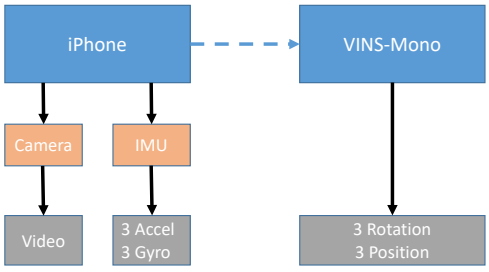


Figure 13. VIO data collection and process.

4.3.2 Turning Angle Detection The generated motion trajectory is simply a sequence of 3d points which is different from the synthetic trajectories (sequence of nodes) used in training. Therefore, using the proposed representations, we first identify the nodes where the object turns significantly. This will provide the angle representation. For augmented angle representation, the algorithm inserts auxiliary nodes at defined sampling distance on the edge between the two nodes with significant turns. In our approach, Douglas-Peucker algorithm

(Douglas, Peucker, 1973) is used to convert the curvilinear segments to line segments. This algorithm can fit piecewise lines to a curvilinear segment while minimizing the number of nodes and as a by product, detects significant turning points. The Figure 5b is the line segment simplification example from real VIO trajectory.

4.4 Localization Results and Analysis

We show five VIO trajectories in the Figure 14 as testing data. Most trajectories are successfully located on the map. In case (a) (b) and (e) (f), the results show the augmented angle representation is located, and angle only representation has failed. For those failed cases, there are two main reasons. One is related to the training dataset which does not have the noise present in these trajectories. The other reason is related to identifying the turning nodes from the noisy trajectory. In our experiments, despite limited test examples, we observe that the augmented trajectory representation obtained more robust localization performance than pure significant turning angle based on the training loss, confusion matrix and test results.

5. CONCLUSION

In this paper, we introduce a map learning approach and propose a motion based localization using recurrent neural network that can be employed in GPS contested environments and is independent of the sensors used to generate motion trajectories. The localization problem has been formulated as a sequence classification problem in a novel way that can easily be generalized to other topological maps. The training of the same network structure with two different representations is shown to provide different performances. The proposed method is validated with real visual-inertial odometry data that shows objects can be localized in a given map without assistance of external signals. In the future, we will explore encoding localization uncertainty and explore other map learning schemes using different neural network structures.

REFERENCES

Amini, A., Rosman, G., Karaman, S., Rus, D., 2019. Variational end-to-end navigation and localization. *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 8958–8964.

Brubaker, M. A., Geiger, A., Urtasun, R., 2015. Map-based probabilistic visual self-localization. *IEEE transactions on pattern analysis and machine intelligence*, 38(4), 652–665.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J. J., 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6), 1309–1332.

Chen, K., de Vicente, J. P., Sepulveda, G., Xia, F., Soto, A., Vázquez, M., Savarese, S., 2019. A behavioral approach to visual navigation with graph localization networks. *arXiv preprint arXiv:1903.00445*.

Douglas, D. H., Peucker, T. K., 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2), 112–122.

Elman, J. L., 1990. Finding structure in time. *Cognitive science*, 14(2), 179–211.

Engel, N., Hoermann, S., Horn, M., Belagiannis, V., Dietmayer, K., 2019. DeepLocalization: Landmark-based Self-Localization with Deep Neural Networks. *arXiv preprint arXiv:1904.09007*.

Fox, D., Burgard, W., Dellaert, F., Thrun, S., 1999. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349), 2–2.

Gupta, A., Chang, H., Yilmaz, A., 2016. GPS-denied geo-localisation using visual odometry. *ISPRS Annual Photogrammetry, Remote Sensing Spatial Information Science*, 263–270.

Hammer, B., 2000. On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1-4), 107–123.

Hess, W., Kohler, D., Rapp, H., Andor, D., 2016. Real-time loop closure in 2d lidar slam. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 1271–1278.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, 9(8), 1735–1780.

Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J., 2012. *Global positioning system: theory and practice*. Springer Science & Business Media.

Hua, J., Zhang, Y., Yilmaz, A., 2019. The mobile ar sensor logger for android and ios devices. *2019 IEEE SENSORS*, IEEE, 1–4.

Li, R., He, S., Skopljak, B., Meng, X., Tang, P., Yilmaz, A., Jiang, J., Oman, C. M., Banks, M., Kim, S., 2014. A multisensor integration approach toward astronaut navigation for landed lunar missions. *Journal of Field Robotics*, 31(2), 245–262.

Mautz, R., 2012. Indoor positioning technologies.

Mur-Artal, R., Montiel, J. M. M., Tardos, J. D., 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics*, 31(5), 1147–1163.

Newson, P., Krumm, J., 2009. Hidden markov map matching through noise and sparseness. *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, ACM, 336–343.

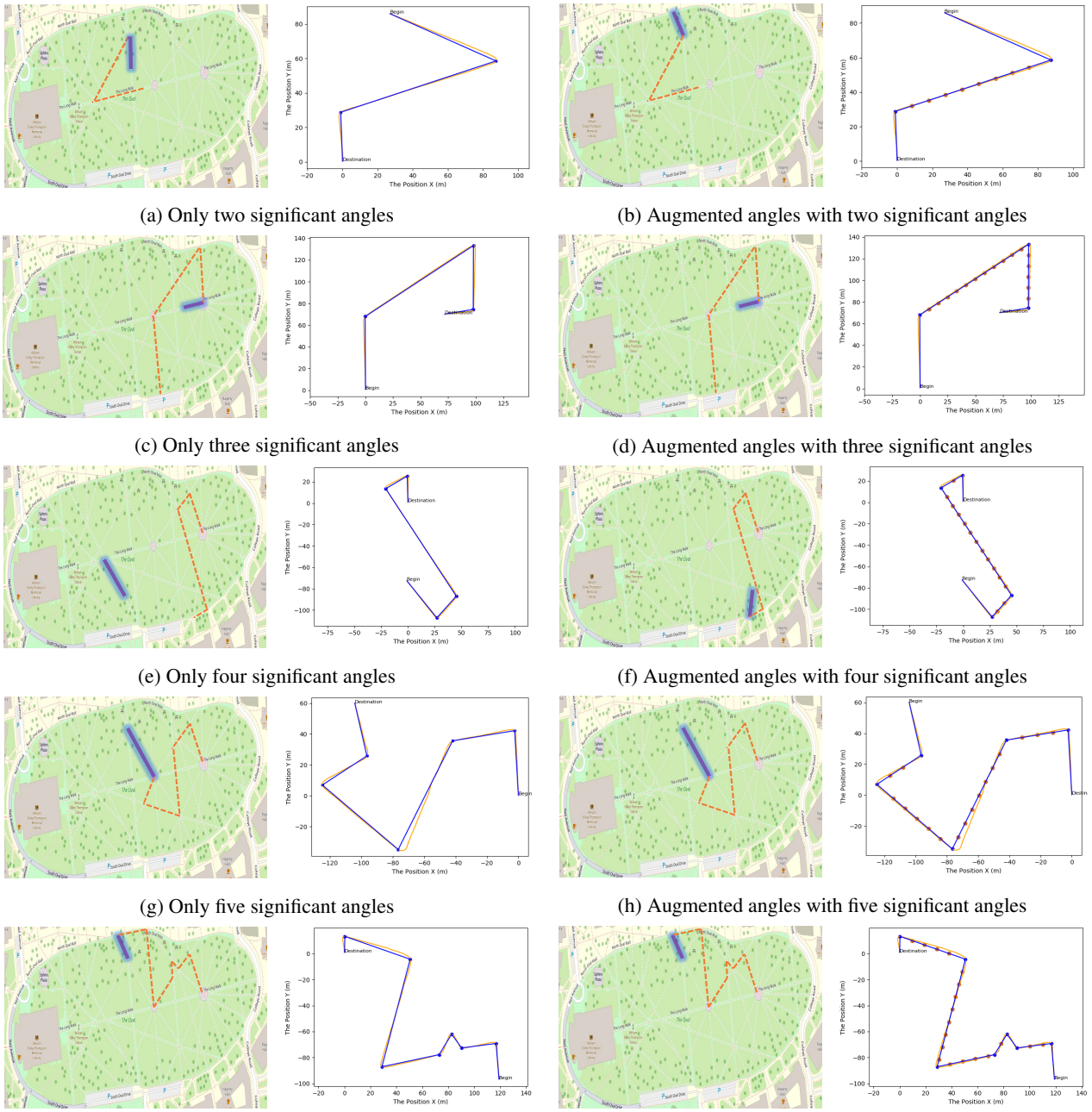
Qin, T., Li, P., Shen, S., 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4), 1004–1020.

Robert, S., 2002. Algorithms in c, part 5: Graph algorithms.

Sargolini, F., Fyhn, M., Hafting, T., McNaughton, B. L., Witter, M. P., Moser, M.-B., Moser, E. I., 2006. Conjunctive representation of position, direction, and velocity in entorhinal cortex. *Science*, 312(5774), 758–762.

Sattler, T., Leibe, B., Kobbelt, L., 2011. Fast image-based localization using direct 2d-to-3d matching. *2011 International Conference on Computer Vision*, IEEE, 667–674.

Scaramuzza, D., Fraundorfer, F., Someone, 2011. Visual odometry tutorial. *IEEE robotics & automation magazine*, 18(4), 80–92.



(a) Only two significant angles

(b) Augmented angles with two significant angles

(c) Only three significant angles

(d) Augmented angles with three significant angles

(e) Only four significant angles

(f) Augmented angles with four significant angles

(g) Only five significant angles

(h) Augmented angles with five significant angles

(i) Only seven significant angles

(j) Augmented angles with seven significant angles

Figure 14. Five different real trajectories localization example with only significant angle case (left side) and augmented case (right side). The number of significant turning angles from (a) to (e) are 2, 3, 4, 5, 7. The orange and blue line are raw VIO trajectory and line segment simplification result, respectively. The glory line in the map indicates location edge prediction

Thrun, S., Burgard, W., Fox, D., 2005. *Probabilistic robotics*. MIT press.

Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., Cremers, D., 2017. Image-based localization using lstms for structured feature correlation. *Proceedings of the IEEE International Conference on Computer Vision*, 627–637.

Wei, J., Koroglu, M. T., Zha, B., Yilmaz, A., 2019. Pedestrian localization on topological maps with neural machine translation network. *2019 IEEE SENSORS*, IEEE, 1–4.

Werbos, P. J. et al., 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550–1560.

Zamir, A. R., Shah, M., Someone, 2014. Image geo-localization based on multiplenearest neighbor feature matching usinggeneralized graphs. *IEEE transactions on pattern analysis and machine intelligence*, 36(8), 1546–1558.

Zha, B., Koroglu, M. T., Yilmaz, A., 2019. Trajectory mining for localization using recurrent neural network. *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 42–45.