Efficient Localisation Using Images and OpenStreetMaps

Mengjie Zhou[1], Xieyuanli Chen[2*], Noe Samano[1*], Cyrill Stachniss[2], and Andrew Calway[1]

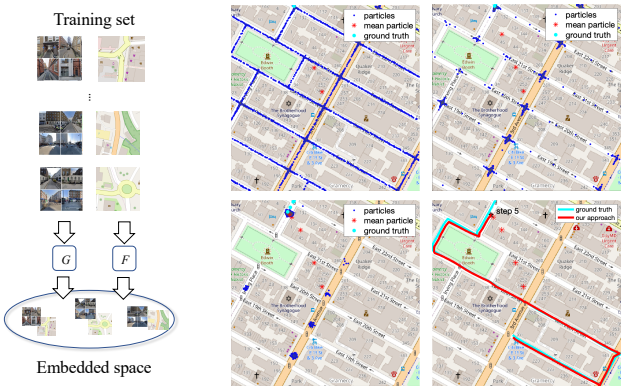
***Abstract*— The ability to localise is key for robot navigation. We describe an efficient method for vision-based localisation, which combines sequential Monte Carlo tracking with matching ground-level images to 2-D cartographic maps such as OpenStreetMaps. The matching is based on a learned embedded space representation linking images and map tiles, encoding the common semantic information present in both and providing potential for invariance to changing conditions. Moreover, the compactness of 2-D maps supports scalability. This contrasts with the majority of previous approaches based on matching with single-shot geo-referenced images or 3-D reconstructions. We present experiments using the StreetLearn and Oxford RobotCar datasets and demonstrate that the method is highly effective, giving high accuracy and fast convergence.**

I. INTRODUCTION

Autonomous localisation and pose tracking are fundamental capabilities for mobile robots. Multiple approaches exist using a variety of different sensors and there have been significant advances over recent years. This is especially true for vision-based techniques, which are now key to emerging applications such as self-driving vehicles and autonomous drones. The majority of approaches in this category match online images with either geo-referenced ground level images, point or dense 3-D reconstructions, or satellite imagery in the case of aerial applications.

However, maps based on images or 3-D reconstructions have two main drawbacks. First, they are difficult to scale up, requiring significant amounts of data storage even for relatively small areas. Second, they represent a single snapshot of the environment and thus online matching becomes problematic when dealing with changes in appearance and 3-D structure over short and long timescales and in the presence of dynamic objects. Although these challenges can be addressed to some extent, the inherent dynamic nature of natural environments suggests that more invariant map representations would be more effective.

One approach is to use semantic representations defined by objects and 'things', rather than storing structure and appearance. This can lead to more compact vector maps and hence increase the potential for scalability. Furthermore, semantic representations are likely to be more invariant to viewing conditions than those based on structure or appearance. One example, and the one that we adopt, is to use 2-D cartographic plan-view maps, such as OpenStreetMaps, which are ubiquitous and freely available in various forms for most of the planet. Defined by the spatial arrangement of semantic entities, such as roads, buildings, and rivers, etc, they provide compact and effective representations suitable for human wayfinding.



(a) (b)

Fig. 1: We use a learned embedded space representation (a) linking map tiles to ground level 4-images extracted from panoramas, which allows localisation using a particle filter (b).

We are motivated by previous work on using cartographic maps for localisation, such as [1] and [2], and in particular the recent approaches described by Panphattarasap and Calway [3] and Samano et al. [4]. These latter works linked semantic features in 2-D maps to ground level images, either via hard-wired features [3], or more effectively, by learning an embedded space [4]. Crucially, when that linkage is combined over routes, it proves highly effective for localisation. However, in both [3] and [4], localisation was implemented using a brute-force search approach, relying on pre-computed descriptors along all possible but constrained routes (excluding loops, for example) defined over discrete locations, making it impractical for many applications.

The main contribution of this work is to address this limitation by combining the embedded representation in [4] with a sequential Monte Carlo tracking framework, to provide an efficient method for localisation, as illustrated in Fig. 1. We use a particle filter implementation with 2-D location and yaw in the state and use comparisons between embedded features from ground level images and map-tiles as the likelihood. The tracking framework is similar in form to that used in [5], although that used an adaptation of the hard-wired features in [3] estimated using LiDAR depth measurements. Experiments on the StreetLearn [6] and Oxford RobotCar [7] datasets, using OpenStreetMap (OSM) [8], demonstrate that the approach is highly effective, giving fast convergence and

[1] Mengjie Zhou, Noe Samano and Andrew Calway are with the University of Bristol, UK. [mengjie.zhou, obed.samanoabonce, andrew.calway]@bristol.ac.uk.

[2] Xieyuanli Chen and Cyrill Stachniss are with the University of Bonn, Germany. [xieyuanli.chen, cyrill.stachniss]@igg.uni-bonn.de.

*Indicates equal contributions.

low error rates. Moreover, we also demonstrate successful localisation for far more complex routes than that presented in [4], and for urban areas significantly different from those used to train the learned representation.

II. RELATED WORK

There has been a large amount of previous work on autonomous localisation using a variety of different maps and sensors. That using vision falls broadly into two categories: indirect methods, which match online images with geo-referenced images [9], with resolution dependent of image density; and direct methods, typically based on dense 3-D models, as in [10], [11], [12], for example, which provide accurate full metric pose estimates. As noted above, these methods are difficult to scale and lack invariance to changing conditions, although work has been done to address the latter using appearance and semantic techniques [13].

A category of approaches which falls somewhere between are those developed for aerial applications, in which online images are aligned with dense satellite imagery, see for example [14] and more recently, [15] and [16]. These methods can provide high resolution localisation and yaw estimates and are related in some respects to our use of 2-D maps in using plan view reference data. Also related is work on learning embedded spaces to achieve ground-to-aerial cross-view matching [17], [18], [19], which have similarities with that used in [4]. However, being based on snapshot satellite data, all these approaches are still difficult to scale and are dependent on viewing conditions.

More directly related to our work are those approaches that use 2-D maps or similar representations. Much of this has been in the context of self-driving vehicles. For example, visual odometry is combined with road topology in [1], [2] to track map location, and extended by Ma et al. [20] to use semantic cues such as road junctions. Seff and Xiao [21] also used semantic features detected in images to correct GPS measurements based on classifiers trained on coupled ground-level images and map tiles and Pauls et al. [22] use semantic segmentation on a monocular camera to localise directly in an HD map. Semantic information from maps such as the location of buildings has also been used in direct methods for estimating metric pose [23], [24]. Other relevant approaches include aerial geolocation using GIS [25] and localisation based on floor plans [26].

Our approach is based on forming direct linkage between ground level imagery and 2-D maps, motivated by the work described in [3] and [4]. The former demonstrated that localisation could be achieved using images alone when coupled with semantic features from maps. They used minimal 4-bit Binary Semantic Descriptors (BSD) indicating the presence or not of junctions and gaps between buildings and showed that combining such features over routes gave reliable localisation. The same features were used by Yan et al. [5] for localisation based on LiDAR measurements and implemented within a sequential Monte Carlo framework similar to that used in this work.

Linking images to 2-D maps was generalised in [4] by learning an embedded space in which co-located images and map tiles are close and demonstrated significantly better performance than using BSDs, again by concatenating embedded descriptors over routes. The method gave improved generalisation, relying less on the presence of specific semantic features. Vojir et al. [27] coupled similar semantic features to BSDs with single image detection of both buildings and depth to generate an embedded space of descriptors that avoided the need for co-located image-map tile pairs as required in [4]. However, the reliance on specific semantic features means that the approach lacks the generality of the latter and limits its application to urban environments with sufficient building presence.

III. METHODOLOGY

Our approach aims at image-based localisation on OpenStreetMap data in an efficient and scale-able way. The key idea is to exploit the neural network proposed in [4], which embeds images and map tiles into a low dimensional vector space, to build an observation model which is then integrated with a motion model to estimate the global geographic location and yaw. The integration between the observation and motion model is done using sequential Monte Carlo Localisation (MCL) in the form of a particle filter, giving highly efficient localisation. This resolves the limitations of the search method used in [3], [4]. In the next section, we provide an overview of the embedded space representation, followed by details of the particle filter implementation.

A. Embedding Images and Maps

To link images to maps and vice versa, deep learning is used to train a network to embed images and map tiles (regularly spaced blocks from an RGB rendered version of the map) into a shared low dimensional vector space where they can be compared using Euclidean distance. We refer to this as the embedding space (ES) and the vectors within as ES descriptors. Locations are represented as 4-images cropped from panoramas in the front, left, right and rear facing directions. In the map domain, places are represented as RGB map tiles centred in the location coordinates.

The architecture of the network has a Siamese-like form, with two independent sub-networks, one to process location images and one for map tiles. Each branch has a feature extractor and a projection module. The output of each branch is a 16-D vector normalized and scaled to live in a hyper-sphere with a radius equals to 32. These form the ES descriptors. The network was trained in an end-to-end way using panoramas from the StreetLearn dataset [6] and map tiles from OSM [8]. Weighted soft margin triplet loss [28] was used in the training with the aim of projecting corresponding image-map pairs near each other inside the embedding space.

To illustrate the performance of the network, Fig. 2 shows examples of query images and the top-5 retrieved map tiles from different datasets, where the correct map tile is outlined in green. Note that the model has learned to relate semantics of the two domains, e.g. buildings, parks and junctions.

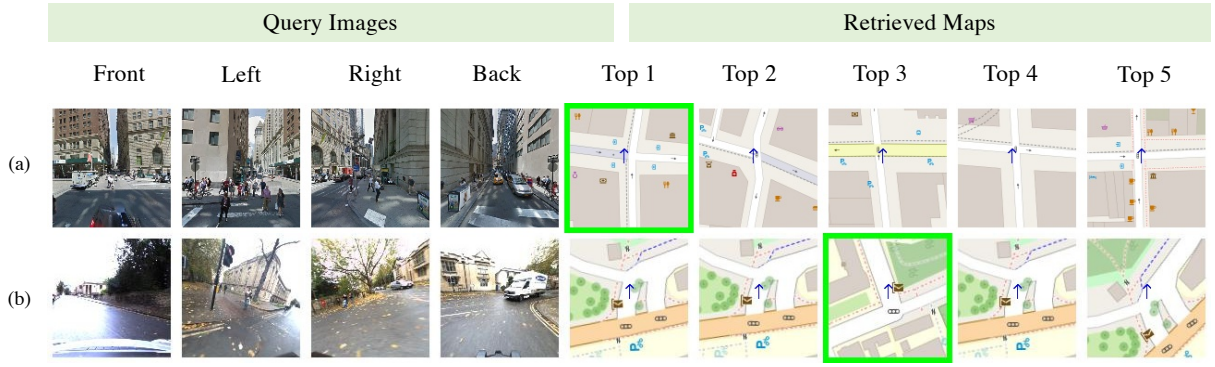


Fig. 2: Top-5 retrieved examples (right) given a query 4-image (left) from (a) StreetLearn dataset and (b) Oxford RobotCar dataset. The green frame encloses the true map location and the blue arrow in the center of the map tile represents the location and yaw.

Full details of the network structure, training procedure and model performance can be found in [4].

B. Monte Carlo Localisation

Monte Carlo Localisation (MCL) is commonly implemented using a particle filter [29]. We seek to estimate 2-D position and yaw, and thus each particle represents a hypothesis of the robot's 2D pose $\boldsymbol{x}_t = (x, y, \theta)_t$ at time t, i.e., 2-D position (x, y) and yaw θ. When the robot moves, the pose of each particle is updated based on a motion model with the control input $\boldsymbol{u}_t$. The expected observation from the predicted pose of each particle in the map is then compared to the actual observation $\boldsymbol{z}_t$ acquired by the robot to update the particle's weight based on an observation model.

Specifically, MCL via a particle filter realizes a recursive Bayesian filter estimating a probability density $p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t})$ over the pose $\boldsymbol{x}_t$ given all observations $\boldsymbol{z}_{1:t}$ and motion controls $\boldsymbol{u}_{1:t}$ up to time t. This posterior is updated as follows:

$$p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) = \eta \; p(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \cdot \int p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) \; p(\boldsymbol{x}_{t-1} \mid \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t-1}) \; d\boldsymbol{x}_{t-1}, \quad (1)$$

where η is a normalization constant, $p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ is the motion model, $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ is the observation model, and $p(\boldsymbol{x}_{t-1} \mid \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t-1})$ is the probability distribution for the prior state $\boldsymbol{x}_{t-1}$.

In our case, as illustrated in Fig. 3, the expected observation is the ES descriptor for the map tile at the particle position, with the tile oriented according to the yaw θ, and the actual observation is the ES descriptor for the panoramic 4-images captured at the current location. The Euclidean distance between the descriptors is then used to update the particle weights. We adopt low variance resampling in combination with adaptive resampling using the effective number of particles to reduce the risk of particle depletion [30], [31]. To do so, we use $N_{eff} = 1/\sum_{i=1}^{N} (w^i)^2$ to estimate how well the current particles set represents the true posterior, where the w^i are the particle weights. The resampling is performed only if N_{eff} drops below a given threshold (we used $3N/4$ in the experiments).

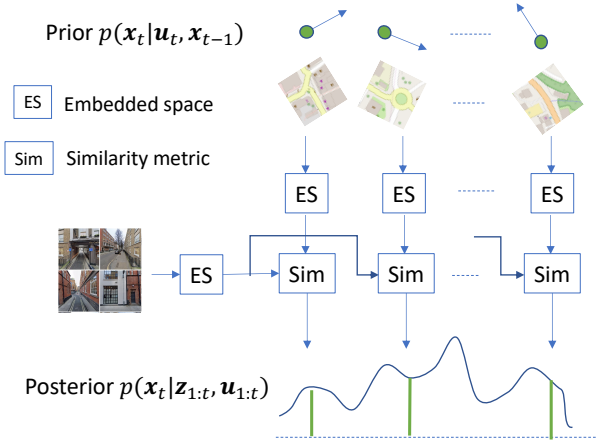


Fig. 3: We implement MCL using a particle filter in which embedded space (ES) descriptors for map tiles corresponding to expected observations (the prior $p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$) and panoramic 4-images are compared using Euclidean distance (Sim) and used to update the posterior.

In the following sections, we focus on the representation of the map we use and on the observation model for the filter, i.e., $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$. For the motion model, we use either noisy versions of the ground truth or visual odometry directly provided by the datasets (see Sec. IV).

C. Map Generation

To achieve localisation, a map representation of the environment is needed. Inspired by the grid representation used in [32], we quantise the OSM maps into a regular grid of locations and generate ES descriptors for map tiles in a set of different orientations around each location, where the different orientations will enable us to estimate yaw alongside 2-D position as described in Sec. III-D. This has the advantage of reducing the computational cost of implementing the filter, since particles are assigned to their nearest grid location and thus the observation update needs only be computed once for all particles associated with the same grid location. The choice of grid and orientation resolution therefore represents a trade-off between accuracy, storage and computational cost.

In the experiments, we used a regular grid with locations spaced at intervals of 1 m in each direction, map tiles corresponding to size 76×76 m^2 and orientations between 0 and 360 degree at intervals of 10 degrees. Note that we only need to store the set of 16-D ES descriptors for each grid location, providing a compact map representation. To further reduce map size, we restrict the map to locations nearby roads (within 5 m distance), on the assumption that we are localising a robot using the road network.

D. Observation model

When the robot senses the surrounding environment, it updates the weights of particles to gradually determine its position and yaw. Each particle i represents a pose hypothesis of the robot $\boldsymbol{x}_t^i = (x, y, \theta)_t^i$ at time t. We use the nearest map ES descriptor in position and orientation as the expected observation of the particle and compare it to the ES descriptor of the current query panoramic $\boldsymbol{z}_t$ generated by the embedded network (see Fig. 3). Let $\boldsymbol{f}_t^i$ and $\boldsymbol{g}_t$ denote the former and latter, respectively, the weight of the i-th particle, w_t^i, is then proportional to the likelihood $p\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t\right)$ derived from a Gaussian observation model, i.e.,

$$p\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t^i\right) \propto w_t^i = \exp\left(-\frac{d\left(\boldsymbol{f}_t^i, \boldsymbol{g}_t\right)^2}{2\sigma^2}\right), \tag{2}$$

where $d\left(,\right)$ denotes the Euclidean distance and σ^2 controls the sensitivity of the observation.

As previously noted, we assume that the robot is located on a road and only store map descriptors for locations on or nearby roads. We thus penalize particles that fall beyond a threshold distance from the nearest map location with an associated descriptor by assigning small weights to them. After updating the weights of all the particles according to the observation, we normalise them in the usual manner using $\hat{w}_t^i = w_t^i / max(w_t)$ and at the end of each iteration, the estimation of the current robot state $\tilde{\boldsymbol{x}}_t$ is calculated by the weighted mean of all the particles, i.e.,

$$\tilde{\boldsymbol{x}}_t = \frac{1}{S}\sum_{i=1}^{N} \hat{w}_t^i \, \boldsymbol{x}_t^i \tag{3}$$

where $S = \sum_{i=1}^{N} \hat{w}_t^i$ and N is the number of particles.

IV. EXPERIMENTS

We carried out localisation experiments to assess the performance of the proposed method using geo-referenced image datasets StreetLearn [6] and the Oxford RobotCar Dataset [7]. We used OSM data as the cartographic map reference and rendered RGB map tiles at required scales using Mapnik [33]. Embedded space descriptors for panoramic 4-images and map tiles were generated using the ES network trained on co-located StreetLearn-OSM pairs, as described in Section III-A and in [4].

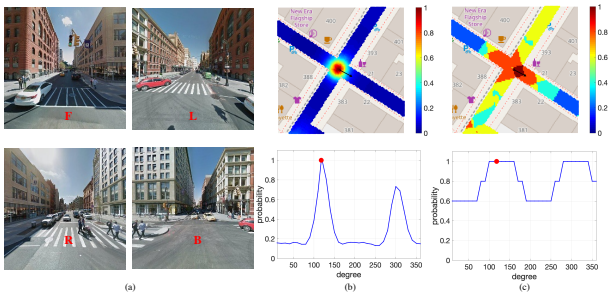


Fig. 4: The observation model with embedding space features (b) and binary semantic descriptors (c), illustrated by the location probability distribution (first row) and the yaw probability distribution (second row), given current panoramic 4-images (a), whose ground truth pose is shown in the map.

A. BSD Comparison

We also compared localisation performance when using the BSDs described in [3] within the observation model, instead of the ES descriptors. The former are 4-bit descriptors indicating the presence or not of junctions in the front and rear directions and gaps between buildings in the left and right directions. These were derived deterministically for each map tile from vectorised OSM data and using CNN classifiers trained on StreetLearn data [4] applied to the panoramic 4-images to detect junctions and building gaps in the different viewing directions.

Since these are binary descriptors indicating the presence or otherwise of features in the four directions, we base the support for particles on the Hamming distance between descriptors and use the following observation model

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t^i) \propto w_t^i = 1 - 0.2\; d_H(a_t^i, b_t) \tag{4}$$

where a_t^i is the BSD corresponding to the map tile of the ith particle and b_t is the BSD derived from the panoramic 4-image. All other elements in the filter were kept the same.

To illustrate the difference between the ES descriptors and the BSDs, Figs. 4b and 4c show observation model values for position and yaw samples for ES descriptors and BSDs, respectively, given the panoramic 4-image observation shown in Fig. 4a corresponding to the location at the centre of a crossroads. Note that for both position and yaw, the ES descriptors give significantly greater discrimination, which as we show next, leads to better localisation performance when incorporated within the filter.

B. StreetLearn Dataset Evaluation

The StreetLearn dataset [6] contains 113,767 panoramic images extracted from Google StreetView in the cities of New York (Manhattan) and Pittsburgh, U.S. Metadata is provided for all images, including geographic coordinates, neighbours and yaw. The average distance between each location in this dataset is around 10 m. Using breadth-first search and the same central panoramic image identifiers as used in [6], three testing subsets were generated, Union Square (US), WallStreet (WS), and Hudson River (HR), each with 5000 panoramic images, covering around 75.6

km, 73.1 km, and 69.3 km trajectories, respectively. The remaining locations in Manhattan, together with all locations from Pittsburgh, were used for training the embedded space network and the BSD classifiers [4].

In the first experiment, we generated 50 random routes in each of the three areas, each route consisted of 40 locations corresponding to approximately 400 m and constrained to contain no loops or direction reversals. For each route, we ran the particle filter on the panoramic 4-images along the route to get successive estimates of the location and yaw according to (3). We used the ground truth positions and yaw angles to simulate motion control $\boldsymbol{u}_t$ between locations and used a Gaussian motion model with mean $\boldsymbol{u}_t$ and standard deviations of 1 m in translation and 5 degrees in rotation. To initialise the filter, a particle was allocated to each central grid location of road in the map and its yaw set to align with the road direction (recall from Sec. III-C that map locations are restricted to those in the vicinity of roads). The initial number of particles for the three regions US, WS and HR was 75615, 73052 and 69308, respectively. To speed up the operation of the filter, once the number of grid locations containing particles reduces to 2% of the initial number of particles, we reduce the number of particles to 1000.

TABLE I: Comparison of location and yaw errors and convergence time using embedded space descriptors (ES) [4] and binary semantic descriptors (BSD) [3] for Union Square (US), Wall Street (WS) and Hudson River (HR) testing datasets.

	Obs	US	WS	HR
Location RMSE [m]	ES	4.35±1.69	4.36±1.28	3.65±1.14
	BSD	4.31±1.47	5.04±2.02	4.40±1.92
yaw RMSE [deg]	ES	6.75±0.78	6.84±0.80	6.45±0.82
	BSD	7.09±1.76	6.67±0.95	7.09±1.19
Convergence Time [steps]	ES	9.83±3.51	9.49±4.24	11.31±4.15
	BSD	23.38±9.65	19.88±7.73	22.13±9.46

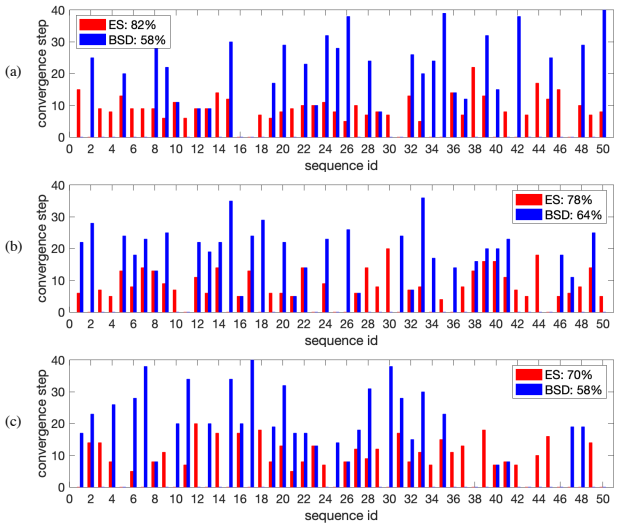


Fig. 5: Number of filter steps before convergence and mean success rates taken over 50 sequences from US(a), WS(b) and HR(c).

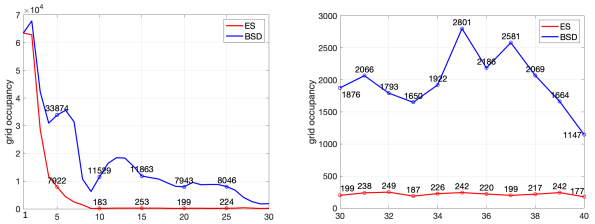
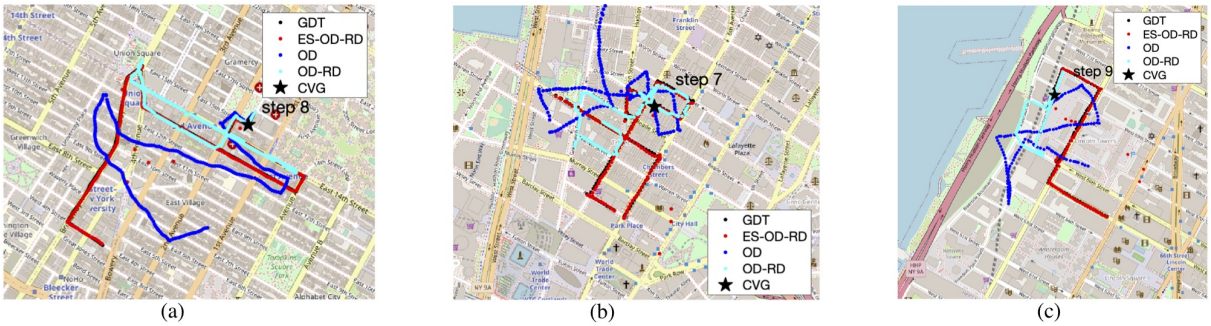


Fig. 6: Grid occupancy values as a function of filter step for (a) steps 1-30 and (b) steps 31-40, for one of the test routes in Union Square when using ES descriptors (red) and BSDs (blue) in the observation model.

Table I shows RMSE results for location and yaw and average convergence time when using the ES and BSD descriptors in the observation model. The location and yaw RMSE values were only computed once the filter was deemed to have successfully converged, which we defined to be once the absolute location error was below 10 m (difference between StreetLearn locations). These results demonstrate that the use of both types of descriptors leads to successful estimations of location and yaw, with RMSE values below 5 m and 10 degrees, respectively, and both approaches give similar accuracy. The key difference, however, is in the convergence time (expressed in terms of the number of steps along a route), which are significantly lower when using the ES descriptors. This reflects the greater discrimination and generalisation afforded by the embedded space descriptors, compared with the specific hard-wired features used in the BSD. This is further confirmed in Fig. 5 which shows the distribution of convergence time and average success rates for both methods and in Fig. 6, which shows the number of grid locations containing one or more particles as a function of the filter steps for one of the routes in US area. Note that as well as faster convergence, the proportion of cases with successful convergence is significantly greater using ES descriptors and that the grid occupancy is also significantly smaller, indicating tighter clustering of the particles.

Comparison with localisation results reported in [4] is not straightforward, given the significant difference in approach. The localisation approach taken in [4] is essentially a brute-force search method over all possible constrained routes defined over discrete locations, with each route being tested using all the ES descriptors along the route, and assuming that the yaw of each observation is known. Using the same StreetLearn testing subset as used here, localisation success rates of over 90% are reported once approximately 200 m of a route has been completed. However, the method becomes impractical for large areas and when arbitrary routes are allowed due to the number of routes that would need to be stored and tested. In contrast, although area size impacts on computational cost prior to convergence in our method, once convergence has started we make no assumptions about route complexity and we are not constrained to discrete locations, providing continuous estimates of both position and yaw. Moreover, we rely on the observations to focus the particles



(a) (b) (c)

Fig. 7: The trajectory of ground truth (GDT), odometry (OD), odometry with road restriction (OD-RD) and our approach (ES-OD-RD) converge at different step (CVG) for (a) Union Square (3 km, location RMSE = 3.9494, yaw RMSE = 8.5245) (b) Wall Street (2 km, location RMSE = 4.3949, yaw RMSE = 7.5571) and (c) Hudson River (1.5 km, reverse driving at 1 km, location RMSE = 5.0154, yaw RMSE = 7.8919). Noted: In order to show the route more clearly, we zoomed in on these areas.

on the most likely routes and the inherent 'memory' of observations within the filter, rather than a brute-force search over all possible routes. This yields a practical solution in contrast to that in [4]. Given this, the success rates of 82%, 77% and 70% reported in Fig. 5 represent a good outcome and demonstrate the potential of the approach.

C. Complex Routes and Different Cities

To investigate the performance when dealing with more complex routes, we generated 3 long routes within each of the areas US, WS and HR, of length 3 km, 2 km and 1.5 km, respectively. The WS route contained several loops and the HR route contained multiple random direction reversals. Localisation results in terms of position estimates are shown Fig. 7, with position and yaw RMSE values given in the caption. The ground truth is shown in black and the estimates are in red. Note that in all cases convergence is fast (the step at which successful convergence occurs is shown on each plot) and the estimates follow the ground truth accurately.

To illustrate the significance of using the ES descriptors, we also show the trajectories obtained when only using the noisy odometry derived from the simulated motion control, starting from the ground truth initial position, i.e. using only the prior without incorporating observations. We show the odometry trajectory with (cyan) and without (blue) restrictions to road locations. The deviation from the ground truth in both cases illustrates that the observations via the ES descriptors play a dominant role in maintaining accurate localisation. For a more detailed analysis of location and yaw accuracy, Fig. 8 shows the variation of absolute latitude, longitude and yaw error over the US test route following successful convergence at step 8.

Finally, we also tested the method on routes taken from the Oxford RobotCar dataset [7] to demonstrate the generalisation capability of the ES descriptors, which were trained on data from US cities in the StreetLearn dataset. The dataset provides sequences of images taken from 4 cameras mounted on a car pointing in the forward, rear, left and right directions, along with ground truth values for position and yaw. We used the 4 images to derive the ES descriptors at regular intervals over each route and visual odometry estimates also provided

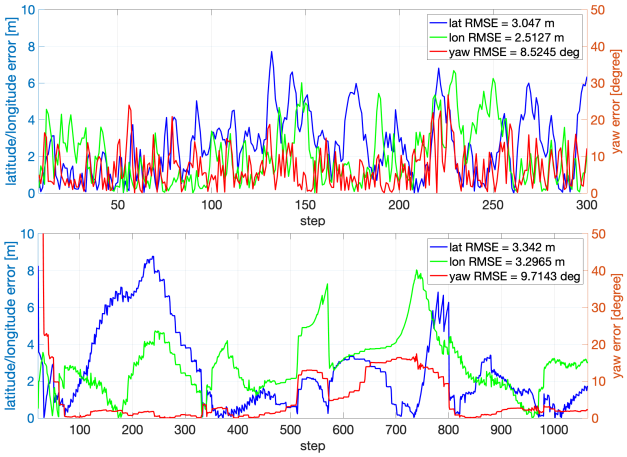
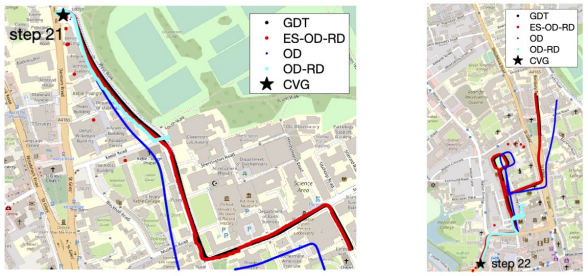


Fig. 8: The latitude, longitude and yaw error after convergence on sequences from Union Square (top) and Oxford (bottom) using our approach.

with the dataset as the motion control. Localisation results for the two sequences, which were of length 1 km and 1.4 km, are shown in Fig. 9, along with the ground truth and trajectories obtained using only the visual odometry, with and without constraining routes to roads. Our method provides good estimates of both position and yaw compared to the ground truth, as confirmed by the error plots in Fig. 8 for one of the sequences. Although at the beginning, where the location error decrease to 10 m, the yaw error is still slightly high. After several more steps, the error falls quickly below 20 degrees. These results are especially encouraging since the appearance of the Oxford locale is significantly different from that of data from New York and Pittsburgh in the StreetLearn dataset, which was used to train the embedded space, as can be seen from the example images in Fig 2.

V. Conclusions

We have presented a novel approach to vision based localisation in urban environments. The method combines a trained embedded space (ES) representation linking map tiles to ground level images from [4] with an efficient particle filter tracking framework. In addition to estimating 2-D



(a) (b)

Fig. 9: The trajectory of ground truth (GDT), odometry (OD), odometry with road restriction (OD-RD) and our approach (ES-OD-RD) converge at different step (CVG) for (a) Oxford-Area1 (1 km, location RMSE = 4.69, yaw RMSE = 9.71) (b) Oxford-Area2 (1.4 km, after 22 step: location RMSE = 8.85, yaw RMSE = 18.99; after 100 step: location RMSE = 5.60, yaw RMSE = 2.96).

position, the filter also provides estimates of yaw angle. The method provides an efficient implementation with potential to scale, in contrast to the brute-force search over discrete routes adopted in [4]. We compared performance between using ES descriptors and BSDs [3] in the observation model and demonstrated that the former provides superior convergence. We also showed that the method generalises well to environments with characteristics significantly different from those used to train the ES descriptors. Future work will focus on exploring in more in-depth the scalability of the method and its performance in comparison with previous appearance and 3-D structure approaches, as well as extending its generality, especially to non-urban areas.

REFERENCES

[1] M. A. Brubaker, A. Geiger, and R. Urtasun, "Map-based Probabilistic Visual Self-Localization," *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, vol. 38, no. 4, pp. 652 – 665, 2016.

[2] G. Floros, B. van der Zander, and B. Leibe, "Openstreetslam: Global vehicle localization using openstreetmaps." in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[3] P. Pilailuck and A. Calway, "Automated map reading: Image based localisation in 2-d maps using binary semantic descriptors," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[4] N. Samano, M. Zhou, and A. Calway, "You are here: Geolocation by embedding maps and images," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[5] F. Yan, O. Vysotska, and C. Stachniss, "Global Localization on OpenStreetMap Using 4-bit Semantic Descriptors," in *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2019.

[6] P. Mirowski, A. Banki-Horvath, K. Anderson, D. Teplyashin, K. Moritz, Hermann, M. Malinowski, M. Koichi, Grimes, K. Simonya, K. Kavukcuoglu, A. Zisserman, and R. Hadsell, "The streetlearn environment and dataset," *arXiv:1903.01292*, 2019.

[7] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[8] Open Street Map. [Online]. Available: https://www.openstreetmap.org

[9] S. Lowry, N. Sunderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, and M. Milford, "Visual place recognition: A survey," *IEEE Trans. on Robotics (TRO)*, vol. 32, pp. 1–19, 2016.

[10] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, vol. 39, no. 9, pp. 1744–1756, 2016.

[11] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson, "City-scale localization for cameras with known vertical direction," *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, vol. 39, no. 7, pp. 1455–1461, 2016.

[12] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss, "Range Image-based LiDAR Localization for Autonomous Vehicles," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[13] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla, "Benchmarking 6dof outdoor visual localization in changing conditions," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[14] D. Sim, R. Park, R. Kim, S. Lee, and I. Kim, "Integrated position estimation using aerial image sequences," *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, vol. 24, no. 1, pp. 1–18, 2002.

[15] A. Shetty and G. X. Gao, "Uav pose estimation using cross-view geolocalization with satellite imagery," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019, pp. 1827–1833.

[16] B. Patel, T. D. Barfoot, and A. P. Schoellig, "Visual localization with google earth images for robust global pose estimation of uavs," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020, pp. 6491–6497.

[17] S. Hu and G. Lee, "Image-based geo-localization using satellite imagery," *Intl. Journal of Computer Vision (IJCV)*, vol. 128, 06 2019.

[18] T. Lin, Y. Cui, S. Belongie, and J. Hays, "Learning deep representations for ground-to-aerial geolocalization," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] S. Workman, R. Souvenir, and N. Jacobs, "Wide-area image geolocalization with aerial reference imagery," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2015.

[20] W. Ma, S. Wang, M. Brubaker, S. Fidler, and R. Urtasun, "Find your way by observing the sun and other semantic cues," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[21] A. Seff and J. Xiao, "Learning from Maps: Visual Common Sense for Autonomous Driving," *arXiv:1611.08583*, 2016.

[22] J. Pauls, K. Petek, F. Poggenhans, and C. Stiller, "Monocular localization in hd maps by combining semantic segmentation and distance transform," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[23] T. Cham, A. Ciptadi, W. Tan, M. Pham, and L. Chia, "Estimating camera pose from a single urban ground-view omnidirectional image and a 2d building outline map," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[24] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit, "Instant outdoor localization and slam initialization from 2.5d maps," *IEEE Trans. on Visualisation and Computer Graphics*, vol. 21, no. 11, pp. 1309 – 1318, 2015.

[25] F. Lindsten, J. Callmer, H. Ohlsson, D. Törnqvist, T. B. Schön, and F. Gustafsson, "Geo-referencing for uav navigation using environmental classification," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010, pp. 1420–1425.

[26] R. Maffei, D. Pittol, M. Mantelli, E. Prestes, and M. Kolberg, "Global localization over 2d floor plans with free-space density based on depth information," 2020.

[27] T. Vojir, I. Budvytis, and R. Cipolla, "Efficient large-scale semantic visual localization in 2d maps," in *Proc. of the Asian Conf. on Computer Vision (ACCV)*, 2020.

[28] S.H., M. Feng, R. Nguyen, and G. Lee, "Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization."

[29] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

[30] C. Stachniss, G. Grisetti, N. Roy, and W. Burgard, "Analyzing gaussian proposal distributions for mapping with rao-blackwellized particle filters," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[31] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. on Robotics (TRO)*, vol. 23, no. 1, p. 34, 2007.

[32] X. Chen, T. Läbe, L. Nardi, J. Behley, and C. Stachniss, "Learning an Overlap-based Observation Model for 3D LiDAR Localization," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[33] Mapnik. [Online]. Available: https://mapnik.org