Work Smart, Not Hard: Recalling Relevant Experiences for Vast-Scale but Time-Constrained Localisation

Chris Linegar, Winston Churchill and Paul Newman

***Abstract*— This paper is about life-long vast-scale localisation in spite of changes in weather, lighting and scene structure. Building upon our previous work in Experience-based Navigation [1], we continually grow and curate a visual map of the world that explicitly supports multiple representations of the same place. We refer to these representations as *experiences*, where a single experience captures the appearance of an environment under certain conditions. Pedagogically, an experience can be thought of as a visual memory. By accumulating experiences we are able to handle cyclic appearance change (diurnal lighting, seasonal changes, and extreme weather conditions) and also adapt to slow structural change. This strategy, although elegant and effective, poses a new challenge: In a region with many stored representations – which one(s) should we try to localise against given finite computational resources?**

By learning from our previous use of the experience-map, we can make predictions about which memories we should consider next, conditioned on how the robot is currently localised in the experience-map. During localisation, we prioritise the loading of past experiences in order to minimise the expected computation required. We do this in a probabilistic way and show that this memory policy significantly improves localisation efficiency, enabling long-term autonomy on robots with limited computational resources. We demonstrate and evaluate our system over three challenging datasets, totalling 206km of outdoor travel. We demonstrate the system in a diverse range of lighting and weather conditions, scene clutter, camera occlusions, and permanent structural change in the environment.

Fig. 1. Appearance change poses a significant challenge to outdoor visual localisation. We grow and curate a map of visual memories which captures the full the spectrum of change in an environment. At run-time, the robot recalls a small number of these visual memories for localisation. We present a technique which enables the robot to predict which visual memory will be relevant to the live camera image, greatly increasing computational efficiency and enabling life-long navigation on robots with finite computational resources. Images here are sampled from the 206km dataset used to evaluate the system.

I. Introduction

This paper is about life-long, vast-scale localisation in challenging outdoor environments, where appearance change can be caused by changes in lighting, weather conditions and scene structure. It is also about intelligently managing a map of visual memories, and the prioritised recollection of past images to support time-constrained localisation.

We approach the localisation problem with a map of "experiences", where an experience is a single representation of the environment under particular conditions, much like a snapshot. Environments which exhibit multiple appearances may need many overlapping experiences to capture the full spectrum of change. While this approach has been shown to provide significant robustness to appearance change [1], it is computationally demanding. As experience density increases, the robot must do more work to obtain a successful localisation. This results in a navigation system which becomes less efficient, and poses a problem for resource-constrained systems. We find that robots with limited computational resources cannot keep up with the additional work load, resulting in localisation performance which degrades over time. This places an unacceptable limit on the robot's ability to navigate in a changing environment.

This paper presents a method for predicting which experiences the robot is most likely to localise against, given how the robot is currently localised in the experience-map. We allow the robot to learn from its past use of the experience-map in order to make these predictions, and present this in a probabilistic framework.

The robot's past use of the experience-map is captured in "path memory". We introduce this concept as a way to encode the robot's localisation history into the map representation. Path memory consists of a collection of paths, where each path links experiences used by the robot during localisation on a particular outing. These paths implicitly link relevant experiences together. For example, consider an experience-map containing sunny and rainy experiences. Without knowledge of the underlying causes of the appearance change (in this case weather), paths link the sunny experiences together, and the rainy experiences together. Path memory is used as training data to make predictions about which experiences will be relevant – this makes the learning

Authors are from the Mobile Robotics Group, University of Oxford, Oxford, England; {chrisl, winston, pnewman}@robots.ox.ac.uk

process unsupervised and one that continually improves.

We evaluate this system against three challenging outdoor datasets covering a total of 206km and show that this technique significantly increases efficiency. We demonstrate that this technique makes life-long navigation possible for resource-constrained robots.

This paper proceeds by reviewing related work in Section II and discusses some preliminary concepts in Section III. Section IV introduces path memory as a way to encode the robot's past use of the experience-map into the map representation. We show how this is used to intelligently retrieve relevant experiences from the experience-map in Section V. The datasets used to evaluate the technique are discussed in Section VI. We present our results in Section VII.

II. Related work

Experience-based Navigation (EBN) has been demonstrated as a robust method for localisation in challenging environments [1]. It was shown that over 53 traverses of a 700m loop in an outdoor environment, the number of experiences required to represent a dynamic environment tended towards a constant. To achieve on-line performance, processing was done on a high-end desktop computer with 2 Intel Xeon 2.93GHz CPUs, offering 16 available cores.

Much of the prior work on localisation in a map of experiences has focused on permanently deleting experiences to maintain on-line performance.

Milford and Wyeth's RatSLAM system [2] has a similar map of experiences to EBN. They prune their map of experiences to keep experience density below a given threshold, where they pick experiences at random for deletion. Results are presented for a two-week period in an office environment, where they point out that the technique works inefficiently with cyclic changes, such as appearance change between day and night. They suggest that the technique would not be suited to handling large-scale topological change.

Glover *et al.* combine the RatSLAM system with FAB-MAP [3] for appearance-based localisation across multiple times of the day in a set of outdoor datasets. They note a linear increase in the number of place-representations with time, and highlight the need for a map management or pruning algorithm for long-term autonomy.

Konolige, *et al.* developed a localisation system based on "views", which are similar to our experiences [4]. In [5], Konolige and Bowman propose a technique for limiting the number of views in an area to a chosen constant, while preserving view diversity and removing unmatched views.

Dayoub and Duckett [6] present a system based on short-term and long-term memory, where stable features in long-term memory represent the adaptive map of the environment, and are forgotten if used infrequently.

In many cases, experience density is the result of localisation failure caused by changes in lighting. McManus *et al.* make use of an illumination-invariant transformation [7] that operates on an RGB image, making subsequent localisation more robust to changes in lighting.

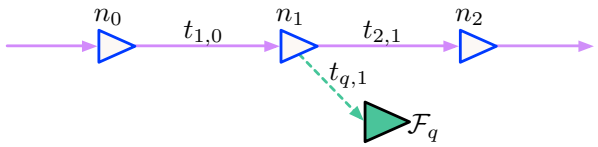


Fig. 2. This figure shows a simple experience graph $\mathcal{G}$, where the robot (with live stereo frame $\mathcal{F}_q$) has a position in the graph specified by a reference node $n_k = n_1$, and a six degree of freedom transformation representing the robot's position in the node's coordinate frame $t_{q,k} = t_{q,1}$.

Johns and Yang [8] assert that representing appearance change with multiple images of the environment is computationally inefficient. Instead, they present a system which operates on the co-occurrence of features at different times of the day.

Maddern *et al.* describe a technique for capping the computation time and storage requirements for a CAT-SLAM system [9]. They prune nodes from the trajectory based on a node's information content relative to its neighbours.

We approach the problem of computational efficiency in a new way. A deletion policy for outdoor environments would need to distinguish between experiences that are outdated, and those only temporarily not relevant. For example, do we want to delete all the map's rainy experiences simply because it has not rained for the past two weeks? But what if the experience density is still too high after pruning? To delete more experiences would reduce the diversity of our map representation. Instead, we present a technique for recalling experiences which are relevant to the live camera image. This allows us to maintain a much larger number of experiences in the map, while only localising in a small subset of these during localisation.

III. Preliminaries

Before discussing this paper's contribution we recap some Experience-based Navigation fundamentals.

A. Visual odometry

Our implementation makes extensive use of visual odometry [10][11][12]. Visual odometry is used to estimate the robot's trajectory through the world, and to estimate the transformation between the robot and an experience saved in memory. Our implementation of visual odometry operates on a stereo image pair to produce a stereo frame $\mathcal{F}_q$ containing a set of observed 3D landmarks, where each landmark is defined relative to the co-ordinate frame of camera. To estimate the robot's trajectory, visual odometry acts on the frame $\mathcal{F}_q$ and the previous frame $\mathcal{F}_{q-1}$. During localisation, $\mathcal{F}_q$ is matched against the 3D landmarks contained in one of the experiences in the map.

B. The experience graph

The experience-map is stored in a graph structure, referred to as the experience graph $\mathcal{G}$. The graph consists of edges and nodes, where nodes contain the set of 3D landmarks extracted by visual odometry, and edges contain six degree

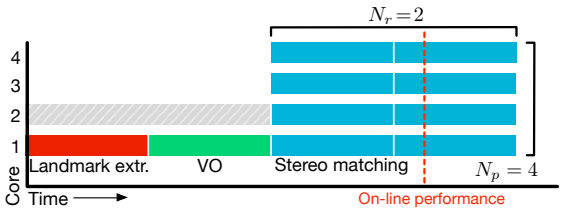


Fig. 3. This graph demonstrates a typical iteration during the localisation phase. A stereo image pair is obtained from the camera. Features are identified using a FAST corner detector [16] and BRIEF descriptor [17]. 3D landmarks are calculated in the stereo frame and saved to $\mathcal{F}_q$. Visual odometry operates on $\mathcal{F}_q$ and $\mathcal{F}_{q-1}$ to update the robot's position estimate in the map. Attempts are made to match $\mathcal{F}_q$ with candidate nodes $\boldsymbol{\mathcal{Y}}$ in the graph. This last step is performed in parallel, where we show the number of parallel processes $N_p = 4$ and the number of matching rounds $N_r = 2$. The requirement for on-line performance is shown as a dashed red line. As shown, with $N_r = 2$ the system would not be meeting its on-line performance criteria, and would have to reduce the number of localisation attempts by setting $N_r = 1$. This results in fewer localisation attempts. We note that the second core is used for the additional processing work during experience creation.

of freedom transformations between nodes. A simple experience graph $\mathcal{G}$ is shown in Figure 2.

The graph structure forms a topometric map [13][14], where over short distances a metric assumption holds, but over large distances we assume only topological connectivity. The robot's position in the map is specified by a reference node n_k (the node currently localised by the robot), and a transformation $t_{q,k}$. This is shown in Figure 2.

C. Localisation and the problem of experience density

Localisation is achieved by obtaining a stereo match between the live frame $\mathcal{F}_q$ and a node n_k in the experience graph $\mathcal{G}$. We distinguish between local and large-scale localisation [14], where large-scale loop closures are detected using FAB-MAP [15], and local localisation (the focus of this paper) is performed within a local metric neighbourhood of the robot's position estimate in the experience graph.

Localisation begins by querying the experience graph for all neighbouring nodes. We refer to these as the set of candidate nodes $\boldsymbol{\mathcal{Y}}$, and they represent the possible nodes in which the robot may localise. We refer to the i^{th} candidate node as ${}^{i}\boldsymbol{\mathcal{Y}}$. The number of candidate nodes $|\boldsymbol{\mathcal{Y}}|$ grows with the number of overlapping experiences in an area. Given finite computational resources, we may only be able to attempt to localise in a small number of these experiences before being forced to abort the localiser to maintain constant processing time.

To maximise our chances of obtaining a successful localisation with a limited number of localisation attempts, a ranking policy, Γ, orders the candidate nodes by their likelihood of obtaining a successful localisation with the live stereo image. We use a simple ranking policy $\Gamma(\text{distance})$ which ranks candidate nodes by their distance to the robot's estimated position. We found that ranking policies based on a simple image similarity metric did not accurately rank candidate nodes while meeting computational requirements.

Figure 3 illustrates the pipeline for localisation, showing the parameters which control the number of matches that can be performed. Localisation performance could be improved by increasing the number of CPU cores N_p, or by increasing the number of matching rounds N_r, but this would require more expensive hardware, or a reduction in on-line performance, both of which may not be feasible. Rather, this paper presents a technique for improving the ranking policy, Γ.

IV. Path Memory

In this section we present "path memory" as a way to encode the robot's past use of the experience graph. Path memory is a collection of paths, where an individual path records the robot's trajectory through the experience graph on a particular outing. A path implicitly links nodes that represent the environment under similar conditions. For example, consider the two paths in Figure 4. The red path might have been created on a sunny day and the blue path on a rainy day. The two paths link different nodes, since the weather conditions force the robot to localise in either sunny or rainy experiences. If the robot re-visits the area for a third time and starts to localise to nodes on the sunny path, we can infer that the robot will probably localise to other sunny nodes in the near future too. So, without knowing anything about what caused the appearance change in the environment, the robot can automatically learn which nodes are more likely to result in a successful localisation.

Previously we discussed metric edges containing transformations which gave the experience graph a relative structure. Here, we introduce non-metric edges which do not contribute to the relative structure. A path $\mathcal{P}_m$ of length k consists of a sequence of these non-metric edges, $\mathcal{P}_m = [e_0, e_1, ..., e_k]$, where an edge connects two nodes, n_s and n_t in the experience graph $\mathcal{G}$. Path creation is done incrementally at run-time, after the localiser has finished processing. If the localiser returns a successful localisation matching the live frame $\mathcal{F}_q$ to node n_k, which is different to the previously matched node n_{k-1}, the robot is said to have moved in experience space from n_{k-1} to n_k. This triggers the creation of a new edge belonging to path $\mathcal{P}_m$, between n_{k-1} and n_k.

We define path memory as a collection of these paths:

$$\boldsymbol{\mathcal{P}} = \left\{\mathcal{P}_1, ..., \mathcal{P}_{|\boldsymbol{\mathcal{P}}|}\right\}$$

where $|\boldsymbol{\mathcal{P}}|$ is the number of paths recorded, and a single path $\mathcal{P}_m$ represents the robot's trajectory through the experience graph on a particular outing.

In the following section, we show how this information is leveraged to intelligently select candidate nodes for localisation.

V. Learning to Select Relevant Experiences

We propose a probabilistic framework which enables the robot to predict candidate nodes that are likely to localise successfully. We generate a probability distribution over the set of candidate nodes $\boldsymbol{\mathcal{Y}}$ based on a set of conditionally independent observations. The system uses the robot's past localisation history as training data, and since this is stored

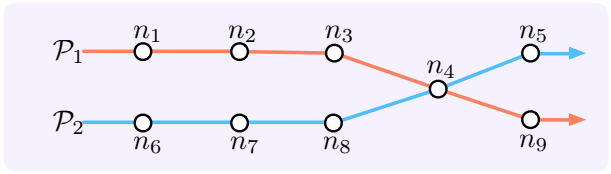
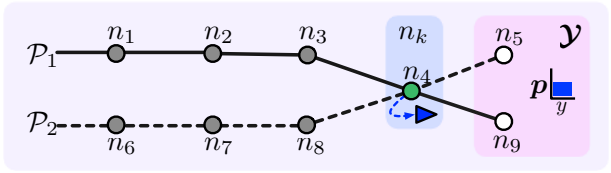
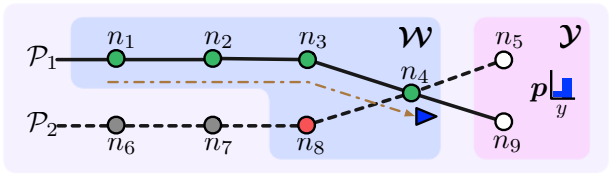


Fig. 4. Paths connect nodes in the experience graph. Here, two paths are shown, $\mathcal{P}_1$ (red) and $\mathcal{P}_2$ (blue). From this we can infer that nodes $\{n_1, n_2, n_3, n_4, n_9\}$ represent the environment under similar conditions (e.g. afternoon sunshine), and $\{n_6, n_7, n_8, n_4, n_5\}$ represent the environment under a different set of conditions (e.g. rain). Note that the robot may localise in some nodes under both conditions, as shown here by n_4.



(a) Prior



(b) Likelihood

Fig. 5. This diagram illustrates how path memory is leveraged to make predictions about which candidate nodes are most likely to result in a successful localisation. Nodes in the experience graph are labelled $n_1, ..., n_9$ and two paths from path memory are shown, $\mathcal{P}_1$ and $\mathcal{P}_2$. In this example, the set of candidate nodes are $\mathcal{Y} = \{n_5, n_9\}$. In Figure 5a, n_k is the node the robot is currently localised against ($n_k = n_4$). The prior distribution over the candidate nodes is calculated based on the number of paths connecting n_k and each node in $\mathcal{Y}$. In Figure 5b, the set of observation nodes $\mathcal{W} = \{n_1, n_2, n_3, n_4, n_8\}$ are the nodes which the robot has recently tried to localise against (the robot's trajectory is shown as a dot-dashed line). Green nodes denote successful localisation attempts, whereas red nodes denote failed localisation attempts. The corresponding observation vector is thus $\mathcal{Z} = [1, 1, 1, 1, 0]$. A probability distribution over the candidate nodes is generated as shown, which is combined with the prior to make predictions about which candidate node is most likely to result in a successful localisation.

implicitly in path memory $\mathcal{P}$, the learning process is unsupervised.

In addition to recording path memory $\mathcal{P}$, we also explicitly record a summary of the robot's recent localisation attempts on the live run. We record these nodes in $\mathcal{W}$ and refer to the j^{th} node in the set as ${}^j\mathcal{W}$. We define the term "recent" by the parameter T, where T is the number of preceding iterations of the localiser to remember (recall that the localiser can make several localisation attempts on one iteration). An example of these nodes is shown in Figure 5b, where $\mathcal{W} = \{n_1, n_2, n_3, n_4, n_8\}$.

Each localisation attempt with nodes in $\mathcal{W}$ will have resulted in either a successful or failed stereo match. In Figure 5b, successful localisation attempts are marked green, while failed localisation attempts are marked red. We record success or failure in the binary observation vector $\mathcal{Z} = [z_1, ..., z_{|\mathcal{W}|}]$. For each node in $\mathcal{W}$, there exists a corresponding bit in $\mathcal{Z}$ such that:

$$z_j = \begin{cases} 1 & \text{if localisation with node } {}^j\mathcal{W} \text{ succeeded} \\ 0 & \text{if localisation with node } {}^j\mathcal{W} \text{ failed} \end{cases}$$

In the example in Figure 5b, we see that $\mathcal{Z} = [1, 1, 1, 1, 0]$.

Lastly, we define the indicator variable $\boldsymbol{y} = (0, ..., 1, ..., 0)^T$ as a vector of length $|\mathcal{Y}|$. One element takes the value 1 and all remaining elements are 0. We define y as a discrete random variable describing which candidate node will localise successfully. We use $p(y = i)$ to refer to the probability of successfully localising the live camera frame $\mathcal{F}_q$ to the i^{th} candidate node ${}^i\mathcal{Y}$.

Using Bayes Theorem, we calculate the probability distribution over $\boldsymbol{y}$ as follows:

$$p(\boldsymbol{y}|\mathcal{Z}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{1}{\beta} \underbrace{p(\mathcal{Z}|\boldsymbol{y}, \boldsymbol{\theta})}_{\text{likelihood}} \underbrace{p(\boldsymbol{y}, \boldsymbol{\pi})}_{\text{prior}} \tag{1}$$

where $\boldsymbol{\theta}$ is a $|\mathcal{Y}| \times |\mathcal{W}|$ matrix describing $p(\mathcal{Z}|\boldsymbol{y})$, $\boldsymbol{\pi}$ is a $1 \times |\mathcal{Y}|$ vector describing $p(\boldsymbol{y})$, and β is a normalisation constant. Since we only need to rank the candidate nodes by their corresponding $p(y = i|\mathcal{Z}, \boldsymbol{\theta}, \boldsymbol{\pi})$, we do not explicitly calculate β.

We discuss the likelihood and prior terms separately, before presenting the system as a whole again.

A. The likelihood

Intuitively, we want to capture the following: in path memory, if many paths connect node ${}^j\mathcal{W}$ and node ${}^i\mathcal{Y}$, it means ${}^j\mathcal{W}$ and ${}^i\mathcal{Y}$ must represent the world under similar conditions (e.g. early morning sunshine). At run-time, if the robot localises in ${}^i\mathcal{W}$, path memory would suggest that we are also likely to localise in ${}^i\mathcal{Y}$. For example, in Figure 5b if we have localised to n_3, we also expect to localise to n_9 since it is connected by a path.

Recall that each binary element in $\mathcal{Z}$ represents an observation, and that each observation corresponds to a node in $\mathcal{W}$ which either succeeded ($z_i = 1$) or failed to localise ($z_i = 0$). We make the assumption that all localisation attempts in $\mathcal{Z}$ are conditionally independent, given that we know which candidate node ${}^i\mathcal{Y}$ will localise successfully. This is a simplification of reality, since the probability of localising to a node on a path is affected by the success or failure of all neighbouring localisation attempts.

We express the likelihood term for a particular candidate node ${}^i\mathcal{Y}$ as:

$$p(\mathcal{Z}|y = i) \propto \prod_{j=1}^{|\mathcal{W}|} p(z_j|y = i). \tag{2}$$

Previously, $\boldsymbol{\theta}$ was introduced as a $|\mathcal{Y}| \times |\mathcal{W}|$ matrix. We define a single element $\theta_{i,j}$ as:

$$\theta_{i,j} = p(z_j|y = i) \tag{3}$$

and refer to the i^{th} row in $\boldsymbol{\theta}$ as $\boldsymbol{\theta_i}$, which corresponds to $p(\boldsymbol{\mathcal{Z}}|y=i)$.

We treat each observation in z_j as a single Bernoulli experiment parametrised by $\theta_{i,j}$. We learn these parameters from path memory:

$$\theta_{i,j} \propto \underbrace{p(y=i|z_j)}_{\substack{\text{Binomial}\\\text{distribution}}} \underbrace{p(z_j)}_{\substack{\text{Beta}\\\text{distribution}}} \tag{4}$$

where $\boldsymbol{\theta_i}$ is normalised such that $\sum_{j=1}^{|\boldsymbol{\mathcal{W}}|} \theta_{i,j} = 1$. Since the Beta distribution is the conjugate prior to the Binomial distribution, the posterior is also a Beta distribution [18]. We calculate $\theta_{i,j}$ using the expectation of the resulting Beta distribution:

$$\theta_{i,j} = \frac{Z_{i,j} + \alpha_j}{\sum_{x=1}^{|\boldsymbol{\mathcal{W}}|} (Z_{i,x} + \alpha_x)} \tag{5}$$

where $Z_{i,j}$ is the number of times a path links ${}^j\boldsymbol{\mathcal{W}}$ and ${}^i\boldsymbol{\mathcal{Y}}$ in path memory, and the parameter α_j specifies the prior Beta distribution. We set $\alpha_j = 1$ to represent the probability that in the absence of path memory, all observations are equally likely. This can be thought of as adding "pseudocounts" to the results from the binomial experiment in order to prevent the "zero count problem" which can occur in sparse training sets [19].

The likelihood term can be thought of as the probability of the robot's live trajectory generating the observation vector $\boldsymbol{\mathcal{Z}}$, given that a particular candidate node localises successfully. In the example in Figure 5b, we would say the observation vector $\boldsymbol{\mathcal{Z}} = [1,1,1,1,0]$ is unlikely if n_5 were to localise successfully. However, the observation vector $\boldsymbol{\mathcal{Z}}$ would be likely if n_9 localised successfully, since this corresponds with the knowledge in path memory. Thus, we calculate the likelihood term as:

$$p(\boldsymbol{\mathcal{Z}}|y=i) \propto \prod_{j=1}^{n} \theta_{i,j}^{\mathbb{I}(z_j=1)} (1-\theta_{i,j})^{\mathbb{I}(z_j=0)} \tag{6}$$

where $\mathbb{I}(x=a)$ is an indicator function, such that:

$$\mathbb{I}(x=a) = \begin{cases} 1 & \text{if } x=a \\ 0 & \text{otherwise} \end{cases}$$

B. The prior

The prior models our initial belief in the probability distribution over $\boldsymbol{\mathcal{Y}}$, in the absence of the observation vector $\boldsymbol{\mathcal{Z}}$. The prior is calculated by querying path memory $\boldsymbol{\mathcal{P}}$ for the number of paths connecting the node n_k currently localised in, to each candidate node in $\boldsymbol{\mathcal{Y}}$. We bias the prior towards candidate nodes with many paths connecting n_k and ${}^i\boldsymbol{\mathcal{Y}}$. An example of this is shown in Figure 5a, where paths connect n_k and the candidate nodes $\boldsymbol{\mathcal{Y}} = \{n_5, n_9\}$.

We use the parameter vector $\boldsymbol{\pi}$ to model the probability distribution over $\boldsymbol{\mathcal{Y}}$, where π_i is the probability that candidate node ${}^i\boldsymbol{\mathcal{Y}}$ will localise successfully. We model $\boldsymbol{\pi}$ as:

$$\boldsymbol{\pi} = \underbrace{p(n_k|\boldsymbol{y})}_{\substack{\text{Multinomial}\\\text{distribution}}} \underbrace{p(\boldsymbol{y})}_{\substack{\text{Dirichlet}\\\text{distribution}}} \tag{7}$$

where $p(n_k|\boldsymbol{y})$ is a multinomial distribution and $p(\boldsymbol{y})$ is a Dirichlet distribution parametrised by the parameter vector $\boldsymbol{\gamma}$. Since the Dirichlet distribution is the conjugate prior to the multinomial distribution, the posterior distribution is also a Dirichlet distribution [18]. We calculate each element in $\boldsymbol{\pi}$ using the Dirichlet expectation for the corresponding i^{th} candidate node:

$$\pi_i = \frac{N_{i,k} + \gamma_i}{\sum_{x=1}^{|\boldsymbol{\mathcal{Y}}|} (N_x + \gamma_x)} \tag{8}$$

where $N_{i,k}$ is the number of paths in path memory connecting n_k and ${}^i\boldsymbol{\mathcal{Y}}$. We set $\gamma_i = 1$ to represent a uniform distribution over the candidate nodes in the absence of path memory.

C. Implementation

We began by introducing our system's output as a probability distribution over the set of candidate nodes $\boldsymbol{\mathcal{Y}}$ in Equation 1. We have shown that this can be achieved by simple event counting (Equations 5, 6 and 8), where events are stored implicitly in path memory. This enables the robot to learn from its past localisation history and make robust yet computationally inexpensive predictions.

We calculate the probability distribution over the candidate nodes $\boldsymbol{\mathcal{Y}}$ using the equations for the likelihood (Equation 6) and prior (Equation 8):

$$p(y=i|\boldsymbol{\mathcal{Z}}, \boldsymbol{\theta_i}, \boldsymbol{\pi}) \propto \pi_i \prod_{j=1}^{n} \theta_{i,j}^{\mathbb{I}(z_j=1)} (1-\theta_{i,j})^{\mathbb{I}(z_j=0)} \tag{9}$$

Finally, we rank the set of candidate nodes $\boldsymbol{\mathcal{Y}}$ by the probability distribution over $\boldsymbol{\mathcal{Y}}$ so that relevant nodes are prioritised over nodes unlikely to obtain a localisation.

VI. Evaluation data

We evaluate our system on three challenging outdoor datasets, covering a total of 206km.

A. Begbroke dataset

The Begbroke dataset (Figure 6) consists of 75km of driving around a 0.7km loop of the Begbroke Science Park in North Oxford. The dataset represents a 12-hour daylight period, with data captured between 7am and 7pm, over a period spanning two months. Data was captured using the Oxford University RobotCar.

B. Keble dataset

The Keble dataset (Figure 7) consists of 56km of data captured around a busy urban environment in central Oxford. Data was collected using a Bumblebee stereo camera attached to the handlebars of a bicycle. Data was collected around a 2.2km route with many intersections and loop closures, over a period of three months, between the hours

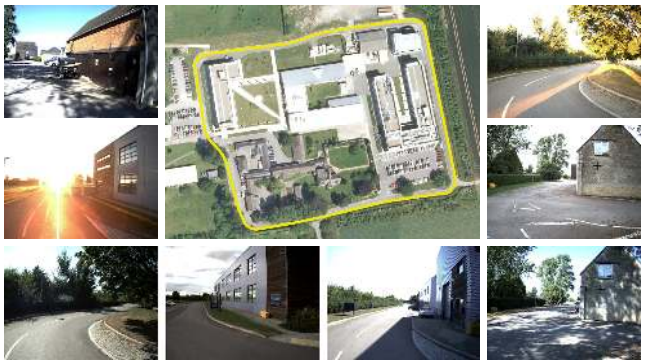


Fig. 6. The Begbroke datatset, consisting of a 0.7km loop with a total of 75km of data recorded.

Fig. 7. The Keble Dataset, consisting of a 2.2km route with a total of 56km of data recorded.

of 9am and 5pm. The dataset contains areas of harsh lighting conditions and changes in weather, occlusion of the camera by pedestrians and cars, and scene structure change.

C. Central Oxford dataset

The Central Oxford dataset (Figure 8) consists of 76km of data captured over a 7.6km route in busy central Oxford. Data was collected using one of the group's autonomous vehicles, the Bowler Wildcat. This dataset represents typical city driving, and includes loop closures, travel in both directions on the same road, occlusion of the camera as a result of other vehicles, and difficult lighting conditions. Data was captured between 9am and 7pm during summer.

VII. Results

In this section we compare the performance of the ranking policy $\Gamma(\text{path})$ with the baseline ranking policy $\Gamma(\text{distance})$ over three challenging outdoor datasets. The results presented here are obtained through 5-fold cross-validation. This is implemented as follows. Firstly, the datasets are divided into five groups. Four of the groups are used to create the experience graph, where new experiences are saved on localisation failure as discussed in [1]. The remaining group is used to localise in the experience graph, but the system is prevented from adding new experiences (it may only localise). This is repeated five times, so that each group of datasets is used for localisation exactly once. The localisation results

Fig. 8. The Central Oxford dataset, consisting of a 7.6km route with a total of 76km of data recorded.

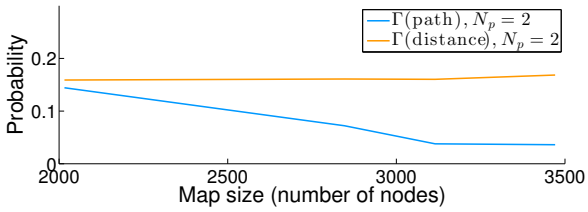


Fig. 9. The probability of travelling more than 10m without a successful localisation is shown as a function of map size (the number of nodes in the experience graph), where the length of the route is kept constant. Without an informed ranking policy, localisation failure increases with map size, whereas our proposed ranking policy $\Gamma(\text{path})$ results in an improvement by a factor of 4 over time.

are aggregated to produce the graphs here, and represent the probability of localisation failure on the robot's next (unseen) outing.

A. Life-long navigation

Figure 9 shows that using the baseline ranking policy $\Gamma(\text{distance})$ on a robot with limited CPU cores results in localisation performance that degrades as nodes are added to the experience graph. Localisation performance is measured as the probability of travelling more than 10m without a successful localisation. The reduction in performance is caused by high node density, where the baseline ranking policy $\Gamma(\text{distance})$ is not able to reliably prioritise nodes likely to localise successfully. Figure 10 plots the number of candidate nodes $|\mathcal{Y}|$ as a function of distance for one lap of the Begbroke Science Park. It also shows that only a small portion of candidate nodes can result in successful localisation. This is because nodes represent the environment's appearance under different conditions, so are by definition visually different to one another.

Figure 9 shows that the ranking policy $\Gamma(\text{path})$ provides a significant improvement in localisation performance, reducing the probability of failure by a factor of 4 as nodes are added to the map. This is because the additional nodes provide experience diversity (we can model the environment

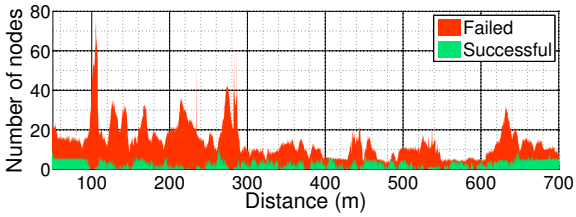
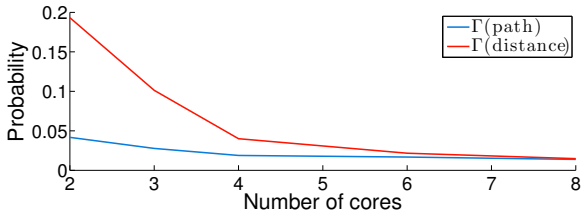
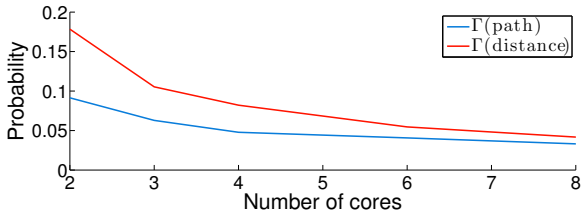


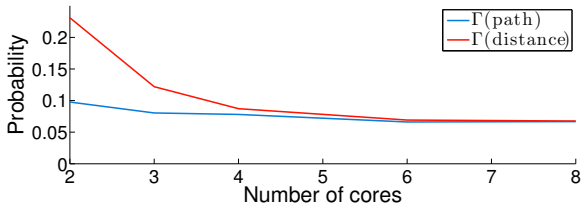
Fig. 10. The size of the candidate node set $|\mathcal{Y}|$ is plotted as a function of distance around one loop of the Begbroke Science Park. The red and green areas are the number of candidate nodes that would have resulted in failed and successful localisation attempts respectively, had the robot attempted to localise against every candidate node in $\mathcal{Y}$. This graph was calculated offline for demonstration purposes - at run-time, the robot can make only a finite number of localisation attempts and needs to predict which node is likely to result in localisation success.



(a) Begbroke dataset



(b) Keble dataset



(c) Central Oxford dataset

Fig. 11. The probability of travelling more than 10m without a successful localisation in the experience graph, as a function of the number of cores available during localisation. We see that in all three datasets the ranking policy leveraging path memory, $\Gamma(\text{path})$, is less likely to result in localisation failure compared with the ranking policy based on distance, $\Gamma(\text{distance})$.

over a wide spectrum of appearance change), which the ranking policy $\Gamma(\text{path})$ exploits by selecting experiences that are more likely to localise the live camera image successfully.

This is a significant result for robots with limited computational resources, as it demonstrates that $\Gamma(\text{path})$ enables long-term autonomy and life-long learning on these platforms.

B. Localisation performance

We present our key result in Figure 11, which compares the ranking policies $\Gamma(\text{path})$ and $\Gamma(\text{distance})$ for the Begbroke, Keble and Central Oxford datasets. The graphs show the probability of travelling further than 10m without a successful localisation in the experience graph as a function

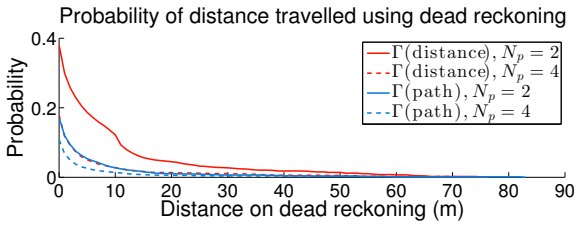


Fig. 12. Graph showing the probability of travelling a certain distance without successfully localising in the experience graph, using the Begbroke dataset. During this period, the robot proceeds in open-loop using visual odometry. The graph shows that using $\Gamma(\text{path})$ maintains performance using using half the number of cores when comparing $\Gamma(\text{path})$ with $N_p = 2$ and $\Gamma(\text{path})$ with $N_p = 4$.

of the number of CPU cores available during localisation. During periods of localisation failure, the robot must proceed in open-loop on visual odometry, which acculumulates drift in the position estimate. After travelling 10m without a successful localisation, the robot begins a re-localisation procedure using FAB-MAP [15].

From Figure 11, it is clear that for both ranking policies, localisation failures reduce as the number of cores are increased, converging on the best possible localisation performance given infinite resources. This is because a greater number of localisation attempts can be made in a single iteration, resulting in a statistically higher chance of choosing the right node. However, $\Gamma(\text{path})$ clearly outperforms $\Gamma(\text{distance})$ when the number of CPU cores is fixed to a small number. In terms of efficiency, $\Gamma(\text{path})$ results in approximately the same localisation performance as $\Gamma(\text{distance})$, but uses only half the number of CPU cores.

Of the three datasets, the Begbroke dataset (Figure 11a) showed the biggest improvement using $\Gamma(\text{path})$, reducing localisation performance by a factor of 5 for number of cores $N_p = 2$. We attribute this to the large amount of training data available. We also note that the Central Oxford dataset's performance with $\Gamma(\text{path})$ and $\Gamma(\text{distance})$ converges at approximately six CPU cores, fewer than the Begbroke and Keble datasets. This is because the Central Oxford dataset does not include as much training data (repeat traverses through the same area) as the Begbroke and Keble datasets, resulting in reduced experience density. Over time, experience density would certainly increase and require more CPU cores to obtain the optimal performance with $\Gamma(\text{distance})$.

Figure 12 uses the Begbroke dataset to present the probability of travelling a certain distance without a successful localisation. The graph shows that $\Gamma(\text{path})$ and number of CPU cores $N_p = 2$ provides nearly identical performance to $\Gamma(\text{distance})$ and $N_p = 4$, showing that the same performance is obtained while performing half the computation work.

We monitored the computational cost of implementing $\Gamma(\text{path})$ by observing the processing time on a 2.3GHz Intel Core i7 processor. We found that processing times never exceeded 0.5ms. This is a negligible portion of the processing time required to perform a single localisation attempt.

We note that while $\Gamma(\text{path})$ outperforms $\Gamma(\text{distance})$ in every test, $\Gamma(\text{distance})$ still performs reasonably well con-

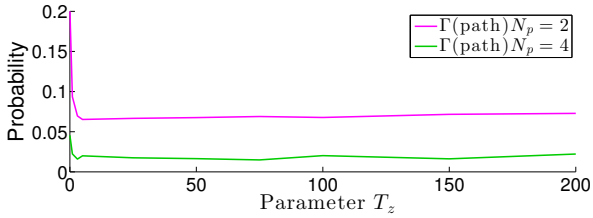


Fig. 13. Graph showing that localisation performance is not sensitive to T, for $T > 5$. This parameter controls the number of preceding localisation iterations to remember when generating the set of observation nodes $\boldsymbol{\mathcal{W}}$.

sidering it operates on very limited information. This is because the point features used by visual odometry during localisation have limited invariance to translation, so nodes that are closer to the robot are more likely to result in successful feature correspondences and consequently in successful localisation. However, this approach scales poorly with experience-density, requiring more CPU cores to process greater numbers of candidate nodes for reliable localisation.

C. The effect of changing T

Parameter T controls the number of preceding localisation iterations to remember when generating the summary of recent localisation attempts $\boldsymbol{\mathcal{W}}$ and corresponding observation vector $\boldsymbol{\mathcal{Z}}$ (see Section V). Figure 13 shows that between $T = 5$ and $T = 200$, localisation failure increases very slightly with T, a result of observations close to the robot being more relevant than those further away. However, this effect is minimal and for $5 > T > 200$ the localisation performance is not sensitive to the parameter T.

For $T = 0$ the performance decreases significantly. This is because the likelihood term (Section V-A) is not used when $T = 0$ and the candidate nodes are predicted solely using the prior distribution (Section V-B). This justifies the use of the likelihood term in Section V-A.

VIII. CONCLUSION

We have presented a technique for life-long, vast-scale localisation in challenging outdoor conditions. We have demonstrated how the prioritised recollection of relevant experiences is crucial for robots with finite resources and a limited amount of time for processing. We evaluate our system on three different datasets totalling 206km of outdoor travel. We show that an informed ranking policy that exploits knowledge of the robot's past use of the experience-map reduces localisation failure by as much as a factor of 5 for robots with a limit on the number of CPU cores and processing time for localisation. Even in the case of sparse training data, the system still outperforms the baseline ranking policy based on distance. From an efficiency point-of-view, we are able to maintain localisation performance while using half the number of CPU cores as previously.

The computational cost of implementing this system is minimal and the performance gains substantial. We show that by predicting relevant experiences, we are able to work smarter and not harder.

IX. ACKNOWLEDGMENTS

The authors wish to acknowledge the following funding sources. Chris Linegar is funded by the Rhodes Trust. Both Paul Newman and Winston Churchill are supported by EPSRC Leadership Fellowship Grant EP/J012017/1. Additionally, the authors acknowledge the support of this work by the European Community's Seventh Framework Programme under grant agreement FP7-610603 (EUROPA2).

REFERENCES

[1] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, 2013.

[2] M. Milford and G. Wyeth, "Persistent navigation and mapping using a biologically inspired slam system," *The International Journal of Robotics Research*, vol. 29, no. 9, pp. 1131–1153, 2010.

[3] A. J. Glover, W. P. Maddern, M. J. Milford, and G. F. Wyeth, "Fab-map+ ratslam: appearance-based slam for multiple times of day," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3507–3512.

[4] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based maps," *The International Journal of Robotics Research*, 2010.

[5] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1156–1163.

[6] F. Dayoub and T. Duckett, "An adaptive appearance-based map for long-term topological localization of mobile robots," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3364–3369.

[7] C. McManus, W. Churchill, W. Maddern, A. D. Stewart, and P. Newman, "Shady dealings: Robust, long-term visual localisation using illumination invariance," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014.

[8] E. Johns and G.-Z. Yang, "Feature co-occurrence maps: Appearance-based localisation throughout the day," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3212–3218.

[9] W. Maddern, M. Milford, and G. Wyeth, "Capping computation time and storage requirements for appearance-based localization with cat-slam," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 822–827.

[10] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[11] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[12] D. Scaramuzza and F. Fraundorfer, "Visual Odometry: Part I - The First 30 Years and Fundamentals," *IEEE Robotics Automation Magazine*, 2011.

[13] G. Sibley, C. Mei, I. Reid, and P. Newman, "Planes, trains and automobilesautonomy for the modern robot," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 285–292.

[14] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2352–2359.

[15] M. Cummins and P. Newman, "Appearance-only slam at large scale with fab-map 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[16] M. Trajković and M. Hedley, "Fast corner detection," *Image and vision computing*, vol. 16, no. 2, pp. 75–87, 1998.

[17] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1281–1298, 2012.

[18] C. M. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1.

[19] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.