Checkout My Map: Version Control for Fleetwide Visual Localisation

Matthew Gadd and Paul Newman

***Abstract*— This paper is about underpinning long-term operations of fleets of vehicles using visual localisation. In particular it examines ways in which vehicles, considered as independent agents, can share, update and leverage each others' visual experiences in a mutually beneficial way. We draw on our previous work in Experience-based Navigation (EBN) [1], in which a visual map supporting multiple representations of the same place is built, yielding real-time localisation capability for a solitary vehicle. We now consider how any number of such agents might operate in concert via data sharing policies that are germane to the shared task of lifelong localisation. We rapidly construct considerable maps by the conjoining of work distributed to asynchronous processes, and share expertise amongst the team by the selective dispensing of mission-specific map contents. We demonstrate and evaluate our system against 100km of data collected in North Oxford over a period of a month featuring diverse deviation in appearance due to atmospheric, lighting, and structural dynamics. We show that our framework is capable of creating maps in a fraction of the time required by single-agent EBN, with no significant loss in localisation robustness, and is able to furnish robots on real-world forays with maps which require much less storage.**

I. INTRODUCTION

This paper considers important aspects of the data management and data exploitation that arise when we consider the long-term operation of fleets of autonomous vehicles using computer vision. Such fleets will need to deal with marked variation of scene appearance and structure over time. While scene change is often experienced at the level of individual agents, there is a need and great boon in intelligently sharing individual experiences across the collaborating team to improve fleetwide localisation robustness and to avoid the ceaseless accumulation of redundant data.

Seamlessly dealing with stark scene change (i.e not getting lost) is difficult, and we have previously addressed it with a map of "experiences", where an experience is a single representation of the environment (or part of it) under particular conditions, which augments the map at runtime.

While this approach has been shown to provide significant robustness to appearance change, it can prove computationally demanding, as environments which are variegated in their appearance may need many overlapping experiences to capture the full spectrum of change. Then, as experience density increases, the machine which the robot is equipped with must (on average) do more work to obtain a successful localisation. This will be particularly aggravated by the totality of experiences over the lifetime operation of a fleet of vehicles, where the question of sharing independently gathered experience data has not been adequately addressed, most simply manifest in the lack of an intuitive guide for the optimum consolidation of all sensory input available. Clearly, some management policies are required.

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

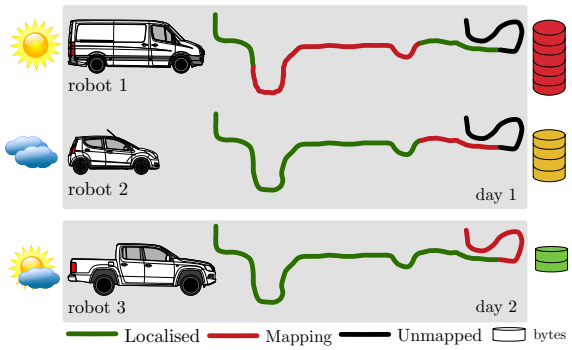


Fig. 1. Our centralised versioning framework facilitates building, managing, and sharing experience maps in a team of robots which are working towards the common goal of localising well in a changing environment. In this simple example, two robots embark on tentative forays into the world, and experience it as it appears to them: mapping (red) when there is an insufficient map representation, and localising (green) when possible. They hand off to a central manager equipped with the ability to rapidly incorporate their offerings, and the savvy to distribute some highly relevant portion of the result to a third robot, which can now localise better, and may experience new parts of the world (black). © Digital Designware / MR Clipart 2009.

This work presents a centralised framework for version control of experience maps, a motivating illustration for which we show in Figure 1, geared towards the lifetime operation of a team of robots working towards the common goal of localising well within a variable environment. We show how periodically (between forays) we can distribute the map construction (quickly assembled, approximately optimal, and of a fine quality) and deploy highly relevant portions of this privileged map to (perhaps resource-constrained) robots in the field, with no significant loss in representational power.

This paper proceeds by reviewing existing literature in Section II and discusses some preliminary concepts in Section III. A framework for the effective management of experience maps proffered by asynchronous processes is presented in Section IV, which we sometimes refer to informally as "dreaming" (in the sense of post-rationalisation of a day's experience). We show in Section V some choices for implementation, which are instantiations of generally useful management policies. Finally, we present our results in Section VI, demonstrating our competency in the lifelong localisation of fleets of resource-constrained robots.

II. RELATED WORK

Mapping and localisation in the space of appearance, particularly on the scale of data collected over the lifetime operation of fleets of robots, demand of us the provision of some robustness to unmodelled modes of appearance change (through ephemeral or enduring environmental, structural and viewpoint changes). Towards lifelong navigation of this kind, early exploitation of generative models describing the

co-occurrence of vocabulary elements are used in robust place recognition [2], and aggregation of scene variation [3]. More recent work has involved searching for consistent sequences of image matches [4], which Pepperell *et al.* [5] have extended to matching places from day to night.

In contrast, Experience-based Navigation (EBN) curates a database of distinct visual experiences which in tandem represent change in the world [6, 1, 7]. Until now, we have not included a framework for distributing experience of the world to asynchronous agents, or (conversely) the intelligent incorporation of sensory input from the fleet. The community is allied with version control systems (VCS), to manage file history [8]. We show that by applying similar concepts, we can build maps quicker than single-agent EBN (with no loss in map quality) and leverage experience within the fleet for more robust localisation of novice agents. In contrast to [9], we avoid globally consistent metric maps.

Yet, without intelligent predictions about which memories to next consider, an unmanageable list of experiences (to search and update) manifests itself, imposing unreasonable requirements on the disk space of field robots. Hence, much literature tries to prune such ever-growing representations to a manageable level, either by limiting spatial density [10], aggregation of cumulative visual experience through topics [11], or the inclusion of only landmarks that are most useful for place recognition [12]. We abhor the deletion of (perhaps ultimately useful) experience, and maintain a comprehensive map in a central repository, while selecting only highly relevant subsets of the map for deployment on robots in the field in order to work with the storage constraint.

Overcoming experience density by attempting matches in a small set of experiences commonly localised against together as in [7] may still result in longer average times to localise particularly when regaining localisation from a lost state (linear in the size of the map). Our selective distribution of smaller maps with more relevant content will alleviate this.

Multi-agent systems (MAS), sharing exteroceptive measurements, can be exploited to add redundancy and improve the comprehensiveness of world representation. Cooperative navigation based on the Extended Kalman Filtering with interleaved update algorithms [13] and central optimal estimation of all robot poses [14] both require robust and efficient communication of robot-to-robot pose information [15]. Our system mitigates such ruinous associations by expediting a search, first attempted more greedily in [1], of the best order in which to incorporate sensory contributions from each robot in order to initially capture more surprising and diverse information in the experience map.

Our framework will have ubiquitous applicability to systems localising in experience maps inspired by somewhat different sensor modalities, such as the LIDAR-based work of Maddern *et al.* [16], but is validated here by a clear improvement on single-agent EBN using stereo cameras.

III. PRELIMINARIES

We begin by establishing some prerequisite knowledge of the operation of single-agent EBN.

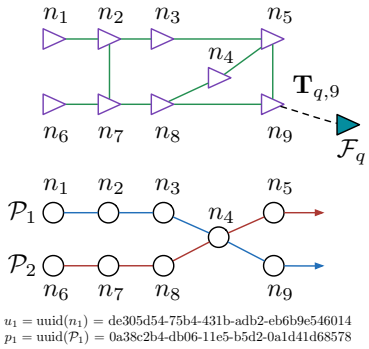


Fig. 2. Experience graphs are implemented as a database of nodes, n, connected by directed edges, which define topometric neighbourhoods linked by 6DoF relative pose, $\mathbf{T}_q$. An adjacent topology, "Path Memory" [7] records the history of localisation success, $\mathcal{P}$. Frames are uniquely identified across databases through the use of 128-bit UUIDs, u.

A. Stereo localisation in experience maps

As shown in Figure 2 the underlying representation of the world is a topometric experience graph $\mathcal{G}$, where nodes store 3D landmarks and edges store six degree of freedom (6DoF) relative poses, both acquired from a visual odometry (VO) [17] pipeline that is able to perform joint optimisation of local landmarks and frame-to-frame relative pose between stereoscopic frames. During fleetwide sharing of information, the estimation must be general to the intrinsic and extrinsic properties of incoming frames due to an exchange of frames between agent and server, and back to (a perhaps differently configured) agent. Localisation is achieved by breadth-first searches in a local graph neighbourhood, and is considered successful on obtaining a stereo match with sufficient inliers.

B. Relocalisation

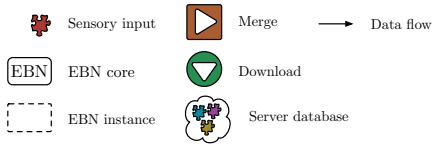
Resets and failures, however, are mitigated by a FAB-MAP search in appearance space [2]. EBN leverages an open-source FAB-MAP implementation [18]. This proves to be the bottleneck performance due to the looking up of a vocabulary match across all nodes in the database (linear in the database size). Deploying intelligently subsampled maps - as we do - alleviates this bottleneck.

C. Path memory

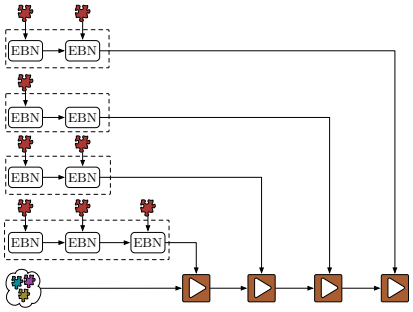
The history and connectedness of successful localisations form a topology atop the topometric poses called "Path Memory", a term introduced by Linegar *et al.* [7], which effectively relates nodes that represent a view of the environment under similar conditions. We will show that we can leverage this historical colocalisation of database content to select statistically relevant sequences of nodes to download.

D. Unique data identification

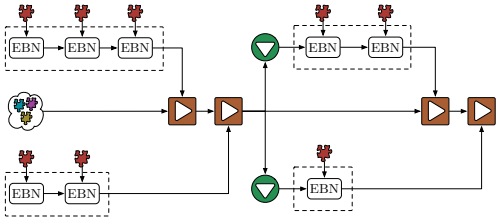
As exemplified in Figure 2, we use 128-bit universally unique identifiers (UUIDs) [19] to annotate each database node. This ensures the ability to track the lifetime usage of images used in experiences as they are passed around between robots and processes, and also ensures that no duplication of frames takes place, in the case that a robot



(a) Processing blocks. A single EBN core operates on a stereo log. Sequential processing of several logs to produce a map is indicated by an EBN instance. We indicate the incorporation of two such maps into a centralised repository as a merge, and the selective deployment of a single map's contents as a download.



(b) Splitting the processing of many logs amongst multiple agents allows us to build large EBN maps much faster than with single-agent EBN, with no loss in map quality.



(c) Sharing information amongst multiple robots between forays into the field allows for more robust localisation of agents. We can also alleviate the stress of strict on-board storage constraints by selectively deploying information most relative to a specific foray or mission.

Fig. 3. We illustrate in (c) our complete version control framework for fleetwide visual localisation, which allows agents to share expertise in a changing world to the rest of the team. This is achieved by periodically incorporating their sensory inputs, illustrated as a single step in (b), before being dealt with a subset of the privileged map most relevant to the next foray.

has downloaded a frame and tries to merge it back into the server-side map. The server knows not to duplicate the frame.

The paths shown in Figure 2 are likewise identified by UUIDs, which we state here not as an implementation detail but as an essential emphasis of the utility each robot makes of the experience map and how it may share such expertise with the team.

IV. SYSTEM OVERVIEW

Figure 3 illustrates two strategies for the management and integration of sensory data proffered asynchronously by multiple agents, which may represent:

- Parallel EBN instances on a server, having split all available logs into smaller groups.
- Several robots submitting to a central server while embarking upon regular forays into the world.

To this end, in addition to the single-agent EBN core, we require two generalised processing blocks:

- MERGE, which combines two EBN maps, detailed in Section V-A.
- DOWNLOAD, which selects some portion of an EBN map for deployment, some choices of implementation for which are given in Section V-B.

The frequency of interaction for MERGE and DOWNLOAD is determined by either scheduling limitations on the server, or mission goals directed at each agent.

Batch processing of data as in Figure 3(b) is advantageous in that all EBN instances are independent, and completed in parallel. MERGE takes as input a decimated frame-rate impression of live camera imagery in the form of information stored on database nodes, and as such is quicker than any EBN instance. We are thus able to build maps much faster than if we were to combine all logs sequentially, as in single-agent EBN. As the server is growing an ever more comprehensive representation of the environment, we expect to ignore redundant information from each EBN instance. Additionally, we can explore some portion of the full combinatorial space of the order in which logs are collected by attempting different organisations for groups (randomly or guided by design) to MERGE into the server.

Figure 3(c) provides an alternative scheme, which lends itself well to a team of robots periodically docking to the server in-between forays into the world. The team, via the server, can share expertise through a DOWNLOAD just prior to journeying out into the world again, which would improve localisation performance at each foray and mitigate the necessity to record unnecessary experiences. Indeed, we can reduce storage requirements on the resource-constrained team members by intelligently choosing which portions of the server database to DOWNLOAD.

V. DATA SHARING POLICIES

We turn our attention now to specific choices of implementation for MERGE and DOWNLOAD. The resemblance of the system to centralised software version control (for example SVN [20]) is clear. MERGE and DOWNLOAD are analogous to submitting a pull request and checking out a revision of code. However, we should stress that the schemes shown in Figure 3 are somewhat agnostic to the choice of implementation for each operation. Indeed, since we are "dreaming", it will prove beneficial to set up the server to make several attempts with different behaviour, so that the final maps are the best that we can achieve.

A. MERGE

In accordance with the description of our system in Section IV, the MERGE processing block incorporates a robot's map, $\mathcal{G}_{robot}$, into the server, $\mathcal{G}_{server}$.

$$\mathcal{G}_{server} \leftarrow \text{MERGE}(\mathcal{G}_{server}, \mathcal{G}_{robot}) \tag{1}$$

When iterating through the database content submitted by the agent, we require an elegant manner in which to mimic a stream of live data. It is thus important to discover chains or segments of frames that are connected linearly by egomotion

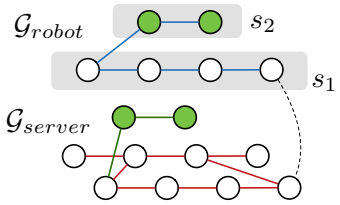


Fig. 4. In our versioning framework, robots hand off sensory information to the server in the form of an independently processed EBN graph (blue). Robots can send database contents asynchronously to the server, which will discover linear segments in the topology (s_1 and s_2). If these localise well (dotted line), they are omitted (s_1) from the privileged map (red), driven by the natural behaviour of EBN localisation. If the segment does not localise well (s_2), this information is regarded as required, and it is included within and connected to the server database (green).

in the $\mathcal{G}_{robot}$ database, which we detail in Algorithm 1 and illustrate in Figure 4.

Here we maintain a lookup between timestamps discovered on nodes (lines 1 to 2) at the termination of edges taken from $\mathcal{G}_{robot}$ and segments (lines 3 to 4) of such linearly connected timestamps (we discover small topometric submaps, lines 5 to 13).

Algorithm 1 The database nodes have been amended with motion $\mathbf{T}_k$ incremental between frames at times t_i. We maintain a correspondence, C, between linearly connected (at terminating timestamps T) segments S of such egomotion.

Require: $\mathbf{T}_k, C = \{T, S\}$
Ensure: $C = \{T, S\}$
1: $t_{k-1} \leftarrow$ source-timestamp$(\mathbf{T}_k)$ ▷ Correspond segment
2: $t_k \leftarrow$ target-timestamp$(\mathbf{T}_k)$
3: $s_{k-1} \leftarrow C(t_{k-1})$
4: $s_k \leftarrow C(t_k)$
5: **if** $s_{k-1} \wedge \neg s_k$ **then** ▷ Prepends segment
6: $C \leftarrow$ add-to-segment(t_k, s_{k-1})
7: **else if** $\neg s_{k-1} \wedge s_k$ **then** ▷ Appends segment
8: $C \leftarrow$ add-to-segment(t_{k-1}, s_k)
9: **else if** $s_{k-1} \wedge s_k$ **then** ▷ Connects segments
10: $C \leftarrow$ fuse-segments(s_{k-1}, s_k)
11: **else if** $\neg(s_{k-1} \wedge s_k)$ **then** ▷ Novel segment
12: $C \leftarrow$ new-segment(t_{k-1}, t_k)
13: **end if**

Merging $\mathcal{G}_{robot}$ and $\mathcal{G}_{server}$ is then achieved by querying the relevance of a discovered segment via:

- Initial seed estimate of location from FAB-MAP (see Section III-B).
- Breadth-first searches in modest graph neighbourhoods (see Section III-A).
- Stereo matching using VO.

Such that, while no data fusion of landmarks is taking place, the relevance of discovered segments is measured by the underlying localiser and as such the MERGE is not a simple union of the two databases. Also note that MERGE makes no attempt to incorporate a frame from $\mathcal{G}_{robot}$ that is already in $\mathcal{G}_{server}$, through a simple database query of the incoming node identifier, $u_k = \text{uuid}(k)$.

As the agent's database content is effectively decimating the live camera stream (EBN creation of database nodes is triggered by some spatial separation), MERGE is much quicker than EBN, allowing us to delegate map-building to several independent agents and obtain large maps quickly.

B. Download

Once more adhering to the schematic in Figure 3, we denote the operation of the DOWNLOAD processing block as some selective dispensing of the server to an agent.

$$\mathcal{G}_{robot} \leftarrow \text{DOWNLOAD}(\mathcal{G}_{server}) \tag{2}$$

A good downloading policy is characterised by:

- An intelligent selection of a mission-specific subset of the database content, without the exclusion of information which is essential to robust localisation.
- Ensuring that the experience map maintains integrity, which in a graph of nodes and edges corresponds to avoiding the creation of disconnected subgraphs.

We now present several candidate policies.

1) Whole-database download ("copy"): In the simplest case we faithfully copy each and every node in $\mathcal{G}_{server}$ to $\mathcal{G}_{robot}$ with no exclusions.

$$\mathcal{G}_{server} \leftarrow \text{copy}(\mathcal{G}_{robot}) = \mathcal{G}_{robot} \tag{3}$$

We are now facilitating a sharing of expertise from agent to agent between their forays into the world. However, the $\mathcal{G}_{server}$ may grow briskly in size beyond the storage capabilities of a resource-constrained robot in the field, and we now present two candidate policies for mitigating this.

2) Selection near the foray time of day ("tod"): A simple strategy for selecting the most relevant portions of $\mathcal{G}_{server}$ involves excluding any database content outside of a temporal window, $T_w > 0$, around the time at which the mission commences, $0 < t_{foray} < 1440$ [min] (24 hour modulus).

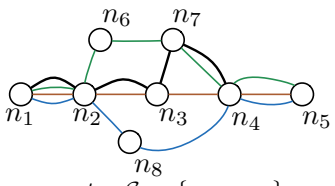
$$\forall n_{server} \in \mathcal{G}_{server}$$
$$n_{robot} = \begin{cases} n_{server}, & \text{if } |t_{foray} - \text{t}(n_{server})| \leq T_w \\ \emptyset, & \text{otherwise} \end{cases} \tag{4}$$

In this, we are making the assumption that appearance change is driven solely by the passage of time. This may account fairly well for illumination effects, but will fail on longer timescales in the face of seasonal and structural variation. This approach also relies on wide scope in the data available: experiences at all points during the day to cover the space of possible mission times.

3) Discovering the most statistically relevant paths ("paths"): We can promote this naïve selection by time alone by further considering patterns within the use that the localiser makes of the map, and infer the set of nodes most likely to be localised against together.

As discussed in Section III-A, the server database $\mathcal{G}_{server}$ is equipped with a record of successful localisations in the form of path memory, a topology connecting nodes $n_i \in \mathcal{G}_{server}$ with paths $\mathcal{P}_j \in \mathcal{P}$.

In Figure 5, we illustrate a simple example of a DOWNLOAD strategy which discovers the most statistically relevant paths for downloading, as a prior on the similarity in the appearance of the environment that we might expect.



$t_1 : \mathcal{G} \leftarrow \{n_1 \dots n_5\}$
$t_2 : \mathcal{G} \leftarrow \{\mathcal{G}, n_6, n_7\}$
$t_3 : \mathcal{G} \leftarrow \{\mathcal{G}, n_8\}$

$\mathcal{P}_1$ $\mathcal{P}_3$
$\mathcal{P}_2$ $\mathcal{P}_4$

M

	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{P}_4$
n_1	1	1	1	1
n_2	1	1	1	1
n_3	1	0	0	1
n_4	1	1	1	1
n_5	1	1	1	0
n_6	0	1	0	0
n_7	0	1	0	1
n_8	0	0	1	0

PCM

	t_1	t_2	t_3
$\mathcal{P}_1$	5	0	0
$\mathcal{P}_2$	4	2	0
$\mathcal{P}_3$	4	0	1
$\mathcal{P}_4$	4	1	0

PWM

	t_1	t_2	t_3
$\mathcal{P}_1$	0.29	0	0
$\mathcal{P}_2$	0.24	0.67	0
$\mathcal{P}_3$	0.24	0	1
$\mathcal{P}_4$	0.24	0.33	0

Fig. 5. Discovering co-occurrence in path memory. During normal mapping activity (nodes added on 3 separate occasions), the EBN core records the use it makes of database nodes in the form of some identified paths through the topology. We can use a *path weight matrix* to recover the sequences of nodes most statistically relevant to a particular mission time.

Just prior to DOWNLOAD, the server has been populated by forays during operating periods of the agents driving around the world

$$t_k = \{t \mid (k-1)\Delta t \leq t \leq k\Delta t\} \tag{5}$$

Where in Figure 5 we have illustrated three such mapping activities. We follow a simple counting procedure to keep track of which paths are registered against all database nodes and populate a *path memory matrix*, M, as

$$M(n_i, \mathcal{P}_j) = \begin{cases} 1, & \text{if } n_i \in \mathcal{P}_j \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Algorithm 2 transforms M to encode the co-occurrence of paths by counting the total number of nodes (line 2) that each path touches within time periods (line 1) in the epoch. The transformation is inspired by representation of motifs in biological sequences (such as genetic sentences) [21], and we refer to it as the *path count matrix*, PCM (line 6).

We can inspect the statistical significance of a path at a certain time of day by normalising PCM to obtain a *path weight matrix*, PWM.

$$PWM(\mathcal{P}_i, k) = \frac{PCM(\mathcal{P}_i, k)}{\sum_{\mathcal{P}_i \in \mathcal{P}} PWM(\mathcal{P}_i, k)} \tag{7}$$

Thus, using Algorithm 3, a foray between times k_{start} to k_{end} (lines 1 to 6) can be furnished with a subset of $\mathcal{G}_{server}$ by choosing the most statistically relevant paths (line 10).

Selecting the most statistically relevant paths for download just prior to a mission in this way is intuitive in that paths tacitly represent the relationship between experiences captured in the map, where (for example) nodes storing sunny experience tend to be localised against during a single foray, and likewise for other broad environmental categories.

Algorithm 2 Populating the path count matrix. We find nodes, N_k, originating from imagery within discrete epochs. If these nodes have been used in localisation along a particular path, $\mathcal{P}_i$, we increment the usage of that path within the epoch.

Require: $K, \mathcal{G}_{server}, \mathcal{P}, D$
Ensure: PCM
1: **for** $k = 1 \dots K$ **do** ▷ Epochs
2: $N_k \leftarrow$ find-nodes$(\mathcal{G}_{server}, t_k)$ ▷ Epoch nodes
3: **for** $n_k \in N_k$ **do**
4: **for** $\mathcal{P}_i \in \mathcal{P}$ **do** ▷ Path memory
5: **if** $M(n_k, \mathcal{P}_i) = 1$ **then**
6: $PCM(\mathcal{P}_i, k) \leftarrow PCM(\mathcal{P}_i, k) + 1$
7: **end if**
8: **end for**
9: **end for**
10: **end for**

Algorithm 3 Choosing the best $p\%$ paths to download before a foray. We look for the most commonly used path UUIDs within the mission times, k_{start} and k_{end}. If data is limited, it is possible to apply some buffer at either end of the mission times, or simply use the earliest k_{min} and latest k_{end} data available. This may also prove useful when deploying the robot with no known end time (live deployment).

Require: $k_{start}, k_{end}, PWM, p$
Ensure: $\mathcal{P}_*$
1: **if** $k_{start} = \emptyset$ **or** $k_{start} < k_{min}$ **then** ▷ Mission start
2: $k_{start} \leftarrow k_{min}$
3: **end if**
4: **if** $k_{end} = \emptyset$ **or** $k_{start} > k_{max}$ **then** ▷ Mission end
5: $k_{end} \leftarrow k_{max}$
6: **end if**
7: $\mathcal{P}_* \leftarrow \emptyset$
8: **for** $k = k_{start} \dots k_{end}$ **do**
9: $\mathcal{P}_{k,*} \leftarrow$ best-p$(PWM(:, k))$ ▷ Top paths for epoch
10: $\mathcal{P}_* \leftarrow \{\mathcal{P}_*, \mathcal{P}_{k,*}\}$
11: **end for**

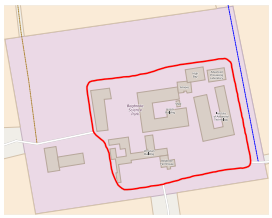
VI. EXPERIMENTS

We present tests of our system on a dataset collected in Summer 2013 by driving our Nissan Leaf *RobotCar* around Begbroke Science Park in North Oxford, as shown in Figure 6(a). The 700m loop was driven at regular intervals at all times of day and night over the period of a month, totalling 100km. It presents us with significant challenges in appearance change, as shown in Figure 6(b). The imagery is obtained from a *Point Grey Bumblebee2* stereo camera at resolution 384×512. We refer to this set of logs as *begbroke24hr*. We divide the full set of logs into groups representing input from a curated virtual team of 2 robots, as further explained in Section VI-A

A. Statistical validation

We present statistically robust experimental results by performing 5-fold cross-validation, involving:

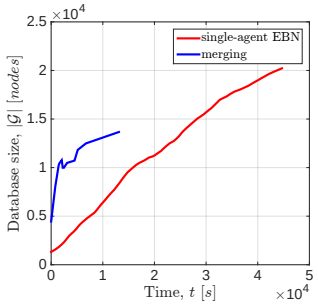
- Dividing *begbroke24hr* into 5 groups.



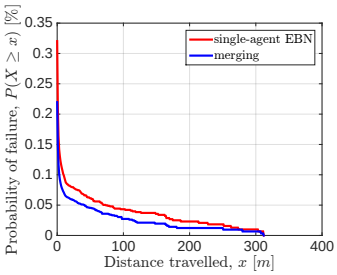
(a) The outer loop at Begbroke Science Park in North Oxford was driven at regular intervals between August 6^{th} and September 5^{th} (30 days) in 2013, totalling 100km of imagery, using a *Point Grey Bumblebee2* stereo camera. © OpenStreetMap contributors.

(b) The dataset presents challenging environmental change due to atmospheric (from glaring sun to gloomy overcast), lighting (at all times of day and night) and structural (fluctuating arrangements of traffic) effects.

Fig. 6. The *begbroke24hr* dataset and experimental setup.



(a) Evolving size of server database. Our merging strategy can build the map in 13120 seconds (3.6 hours), while single-agent EBN requires 44690 seconds (12.4 hours) - a savings of 29.36%. Furthermore the final merged database size is 13665 nodes, compared with 20222 built by traditional EBN (67.57%).



(b) The merging strategy does not compromise on map quality, offering localisation robustness that is at least comparable to single-agent EBN.

Fig. 7. Investigating distributing the building of an EBN database from all logs in the cross-validation set (see Section VI-A) and merging intermediate results for faster map construction, with no loss in the reliability of localisation.

- Mapping with 4 of the groups (the training sets) either via single-agent EBN, or various instantiations of "dreaming" policies, as presented in Sections IV and V.
- Localising in this map using the 5^{th} group (test set).

We repeat this procedure 5 times (such that each group of logs is used as the held-out set once), aggregating the iterations to obtain the plots that we now present. This approach ensures that there is no biased selection of logs with related appearance that should localise well together.

B. Metrics for localisation robustness

In order to enumerate the advantage our system offers, we require a metric for localisation robustness, a candidate for which was first presented by McManus *et al.* [22]. We regard localisation efficacy as the distance the robot (after becoming lost) is likely to have to travel using dead-reckoning (VO in our case) before reacquiring localisation, which we explore as a cumulative distribution, $P(X \geq x)$. A good distribution of this kind will rapidly dwindle to zero from a moderate peak. Generally, one distribution is better than another when it lies below it on a visual plot, unless it has a much longer tail. To be clear, we understand our measure of success to be represented by a comparison to the localisation performance of EBN when presented sequentially with logs collected in their natural order (single-agent EBN).

C. Building good maps quickly

In Figure 7, in a similar spirit to [23], we compare the construction of maps using the merging system first shown in Figure 3(b) against single-agent EBN. It is clear from Figure 7(a) that the maps are built in a fraction of the traditional time, specifically 29.36%.

The final $\mathcal{G}_{server}$ node count is 67.572% of that generated by single-agent EBN. The asynchronous processes are individually experiencing the world, but when the system instigates a merge, we drive the incorporation of information by the ability of the localiser, and no redundant information is recorded where it is not needed.

This would be of little use if significant compromise was made to map quality. Happily, Figure 7(b) shows that this is not the case, as localisation robustness is commensurate. In fact, the test set spent 21.97% of its foray not localised in the map built with our merging strategy, compared with 32.03% in the map built with single-agent EBN. One would expect to localise at best as well as single-agent EBN, and no better. However, our non-traditional layout will avoid the runaway effect of one bad stereo match (corrupting the topometric accuracy in graph neighbourhoods) that processing logs one after another with single-agent EBN is susceptible to.

D. Leveraging visual experiences in a team

As a fleetwide experiment, a pair of robots (r1 and r2) each embark on two forays into the world. Between the forays (just after processing the 7^{th} succession of stereo input, Log 7), they hand off to the server, and download the resulting merged map.

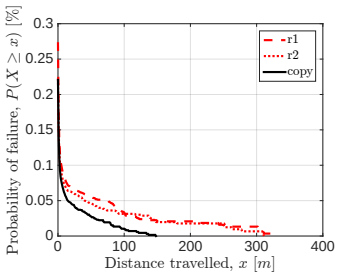


Fig. 8. Sharing and distributing maps between a pair of robots. The held-out set is able to localise more reliably in the shared map (copy) than in either of the per-robot maps that each robot would be limited to if lacking knowledge of each other (r1 and r2). As the number of robots in the experiment increases, the shared map quality should hug the vertical axis more closely.

1) Sharing expertise: In Figure 8 we show the advantages offered by our sharing scheme, limiting ourselves first to downloading by the simple copy strategy of Section V-B.1. It is strikingly evident on inspecting the distributions of localisation failure that map quality in the case of sharing expertise between these agents is better than if the robots had no knowledge of each other and had built EBN maps with only their own sensory input.

2) Distributing mission-specific maps: Figure 9 investigates the effect of excluding irrelevant server content as proposed in Sections V-B.2 and V-B.3.

Figure 9(a) illustrates how we mitigate possibly strict storage requirements. Selecting by time of day with a band of 3 hours or 180 minutes around the current mission time offers immediate respite, where each robot is required in between forays (around Log 7) to respectively store only 81.77% and 79.57% of the nodes it would be required to hold for a complete download (copy). Additional filtering by path, choosing the top 50%, reduces the storage requirement even further, as shown in Figure 9(a), where the savings becomes 74.56% and 66.57%, respectively.

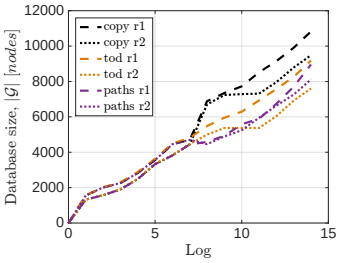
As a sanity check, we show in Figure 9(c) that we are not compromising the in-field localisation reliability of our robots, despite deploying subsampled maps. There is only a slight slump in localisation robustness towards the tails, which is worth noting as the trade-off between localisation and storage savings a system designer will have to make. We see comparable reliability in Figure 9(c) as we have designed policies which implement a canny selection of only content relevant to the current mission (excluded database content will not have aided localisation).

Finally, Figure 9(b) shows a comparison of the average time spent localising at points during each foray, where it is clear that subsampling the total map leads to improved performance and lenience on the processing power required, due to a kinder throughput to FAB-MAP relocalisation, the bottleneck in EBN performance.

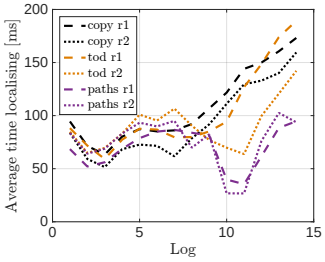
E. Building the best maps

Our framework allows for coarse but quick and useful combinatorial optimisation of the order in which we create EBN maps. We thus use this final experiment to ponder two important issues:

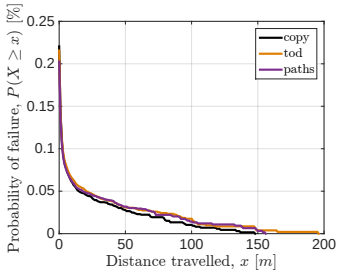
- What magnitude is the effect of the order in which logs are handed to EBN on the quality of representation?



(a) As expected, whole-database download imposes monotonous growth throughout the lifetime operation of each robot. Filtering by time of day and statistically relevant paths periodically (between forays, around Log 7) subsamples the amount of information required on each agent.



(b) Distributing smaller maps to agents in the field improves the rate that the localiser can operate at.



(c) Trading off significant localisation robustness for resource usage can be moderated by the selection of database content most relevant for localisation.

Fig. 9. Investigating sharing and distributing maps between a pair of robots, where small portions of the server's comprehensive environment map are intelligently selected and then downloaded to address resource constraints.

- Are cumulative distributions of localisation failure, as we have used them, a good metric for map quality?

In Figure 10 we have combined 4 groups of 9 logs in every possible order, totalling $4! = 24$ candidate maps. We show the quality of localising a test set in the resulting maps. It is clear that the designer is in possession of a best quality map for the choosing. This is attributed to a idealised addition of the most surprising experiences in the map, which immediately makes it more representative and better for localising. The design choice depends on the distance robots can reasonably travel on dead-reckoning alone, which could be dictated by safety, quality of egomotion estimation available, or frequency of localisation capable with the available processing power.

It can be shown that the time savings we enjoy in exploring the combination of k groups of stereo logs is

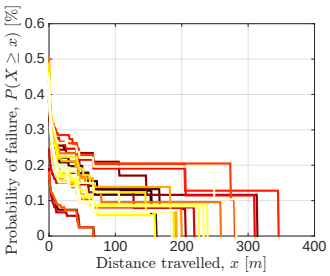


Fig. 10. Our versioning framework, due to both the rapidity of merging compared to single-agent EBN and the copying of independent intermediate results not available in single-agent EBN, allows us to quickly (in 22.44% of the time) explore a combinatorial space for the best order in which to build maps. Each curve shown here is a cumulative failure distribution which we use to judge the quality of the map built by that particular combination of sensory input.

$$C_{savings} = \frac{kT_G + k!(k-1)T_M + k(k!-1)T_C}{(k!(k-1)+1)T_G + (k!-1)T_C} \quad (8)$$

Where in our case there are $k = 4$ groups and $T_G = 10823$[s] is the time for processing each group, $T_M = 1829$[s] is the time to merge one map into another, and $T_C = 27$[s] is the time to copy some intermediate results for reuse (all sampled from jobs submitted to our server). The advantage of distributing EBN processing to asynchronous processes becomes even more clear via this calculation, as the copying of intermediate results for single-agent EBN is limited to only the first group (subsequent groups depend on the map built by the first group). Under our framework, all intermediate (group) results can be copied, and we are rapidly searching for the best map in $C_{savings} = 22.44\%$ of the traditional time.

VII. CONCLUSION

We have presented a technique for gathering, processing, distributing, and managing of experiences amongst asynchronous processes, agents, or robots. We show how to use a centralised version control style framework to assimilate multiple experience maps, and issue only highly relevant maps in between forays, through a medium that facilitates sharing of expertise. The system is evaluated in a robust 5-fold validation process across approximately 140 traverses of a 700m route, totalling 100km of driving in North Oxford. We are thus able to produce maps in a fraction of the time required by single-agent EBN and deploy fractionally sized maps with no significant loss in localisation robustness, as well as leverage expertise in a team with heterogeneous localisation capability. This system is thus particularly well-suited to teams of miscellaneous, resource-constrained robots engaging in regular missions into a changing world.

VIII. ACKNOWLEDGEMENTS

Matthew Gadd is supported by the FirstRand Laurie Dippenaar, Oppenheimer Memorial Trust, and Keble Ian Palmer scholarships. Paul Newman is supported by EPSRC Leadership Fellowship Grant EP/J012017/1 and EPSRC Programme Grant EP/M019918/1. Additionally, the authors acknowledge the support of this work by the European Community's Seventh Framework Programme under grant agreement FP7-610603 (EUROPA2). The authors thank Chris Linegar for his support in interfacing with core EBN software, as well as Winston Churchill for his valuable suggestions.

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. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[3] 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.

[4] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1643–1649.

[5] E. Pepperell, P. Corke, and M. Milford, "Automatic image scaling for place recognition in changing environments," 2015.

[6] W. Churchill and P. Newman, "Continually improving large scale long term visual navigation of a vehicle in dynamic urban environments," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012, pp. 1371–1376.

[7] C. Linegar, W. Churchill, and P. Newman, "Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA2015)*, 2015.

[8] L. Wingerd, *Practical perforce*. " O'Reilly Media, Inc.", 2005.

[9] T. Cieslewski, S. Lynen, M. Dymczyk, S. Magnenat, and R. Siegwart, "Map api-scalable decentralized map building for robots," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6241–6247.

[10] 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.

[11] R. Paul, D. Rus, and P. Newman, "How was your day? online visual workspace summaries using incremental clustering in topic space," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4058–4065.

[12] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "the gist of maps - summarising experiences for life-long localisation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA2015)*, 2015.

[13] A. Bahr, M. R. Walter, and J. J. Leonard, "Consistent cooperative localization," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 3415–3422.

[14] T. Bailey, M. Bryson, H. Mu, J. Vial, L. McCalman, and H. Durrant-Whyte, "Decentralised cooperative localisation for heterogeneous teams of mobile robots," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2859–2865.

[15] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1093–1100.

[16] W. Maddern, G. Pascoe, and P. Newman, "Leveraging Experience for Large-Scale LIDAR Localisation in Changing Cities," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015.

[17] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. IEEE, 2004, pp. I–652.

[18] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, "Openfabmap: An open source toolbox for appearance-based loop closure detection," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4730–4735.

[19] P. J. Leach, M. Mealling, and R. Salz, "A universally unique identifier (uuid) urn namespace," 2005.

[20] W. Nagel, *Subversion Version Control: Using the Subversion Version Control System in Development Projects*. Prentice Hall PTR, 2005.

[21] G. D. Stormo, T. D. Schneider, L. Gold, and A. Ehrenfeucht, "Use of the perceptronalgorithm to distinguish translational initiation sites in e. coli," *Nucleic Acids Research*, vol. 10, no. 9, pp. 2997–3011, 1982.

[22] 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*. IEEE, 2014, pp. 901–906.

[23] R. Aragues, J. Cortes, and C. Sagues, "Distributed consensus on robot networks for dynamically merging feature-based maps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 840–854, 2012.