Efficient Indoor Positioning with Visual Experiences via Lifelong Learning

Hongkai Wen*, Ronald Clark†, Sen Wang‡, Xiaoxuan Lu§, Bowen Du*, Wen Hu¶ and Niki Trigoni§
*Department of Computer Science, University of Warwick, UK
†Dyson Robotics Lab, Imperial College London, UK
‡Institute of Sensors, Signals and Systems, Heriot-Watt University, UK
§Department of Computer Science, University of Oxford, UK
¶School of Computer Science and Engineering, University of New South Wales, Australia

Abstract—Positioning with visual sensors in indoor environments has many advantages: it doesn't require infrastructure or accurate maps, and is more robust and accurate than other modalities such as WiFi. However, one of the biggest hurdles that prevents its practical application on mobile devices is the time-consuming visual processing pipeline. To overcome this problem, this paper proposes a novel lifelong learning approach to enable efficient and real-time visual positioning. We explore the fact that when following a previous visual experience for multiple times, one could gradually discover clues on how to traverse it with much less effort, e.g. which parts of the scene are more informative, and what kind of visual elements we should expect. Such second-order information is recorded as parameters, which provide key insights of the context and empower our system to dynamically optimise itself to stay localised with minimum cost. We implement the proposed approach on an array of mobile and wearable devices, and evaluate its performance in two indoor settings. Experimental results show our approach can reduce the visual processing time up to two orders of magnitude, while achieving sub-metre positioning accuracy.

Index Terms—Visual Positioning, Mobile and Wearable Devices, Lifelong Learning

1 Introduction

The majority of indoor positioning systems to date represent a person's location using precise coordinates in a 2D or 3D metric map, which has to be globally consistent. However, in many scenarios this could be an overkill: we don't really need global maps to find a particular shop in the mall, as long as someone, e.g. the shop owners, could guide or "teach" us step by step. Therefore, we envision that in the future locations should be merely *labels*, which are associated with objects, people, or other pieces of relevant information. In the same way as people exchanging mobile phone contacts, they can share locations, or to be more precise, the look and feel along the ways towards them, where others can ask their mobile phones or smart glasses to take them to "Jackie", "Terminal 1" or "Mona Lisa", by following navigation instructions extracted from previously constructed experiences.

Recently, this teach-repeat approach is gaining its popularity and has been implemented with various sensing modalities [1], [2], [3]. Comparing to the traditional solutions which seek to compute the global coordinates of the users [4], [5], [6], those teach-repeat systems require much less bootstrapping and training effort. For instance, the Escort system [1] navigates a user towards another by combining her previously recorded inertial trajectories with encounters from audio beacons. The FollowMe system [2] collects traces of magnetic field measurements as someone walks towards a destination, e.g. from the building entrance to a particular room. Later when another user tries to navigate to the same place, her position is estimated by comparing the live magnetic signal and step information with the stored traces. However in complex environments, the discriminative power of 1D sequence matching on magnetic field magnitude is limited. On the other hand, the Travi-

Corresponding author: Bowen Du, b.du@warwick.ac.uk

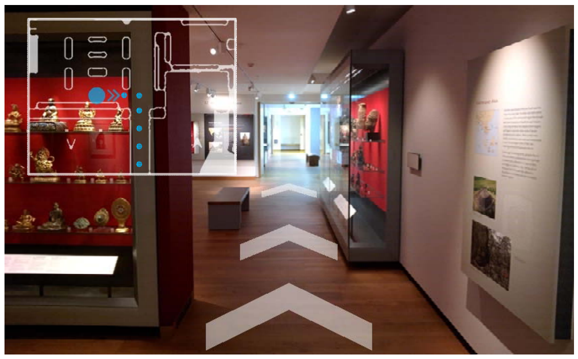


Fig. 1. The user interface of the proposed positioning system on smart glasses running in a museum environment.

Navi system [3] uses vision for teach-repeat navigation, which is promising since appearance is more informative than other modalities. In addition, with the emerging smart glass technology, vision-based solutions become more advantageous, since smart glasses are rigidly mounted on the head of the users, with cameras that are able to capture first-person point of view images (as shown in Fig.1). This is particularly useful in applications that require real-time and hands-free positioning, such as personal guidance for the visually impaired, remote assistance in industrial settings[1], and augmented reality.

However in practice, achieving real-time visual positioning on mobile and wearable devices presents a number of challenges. Firstly, processing visual data can be prohibitively expensive for

1. "SAP and Vuzix bring you the future of Field Service". https://www.youtube.com/watch?v=UlpGDrSmg38

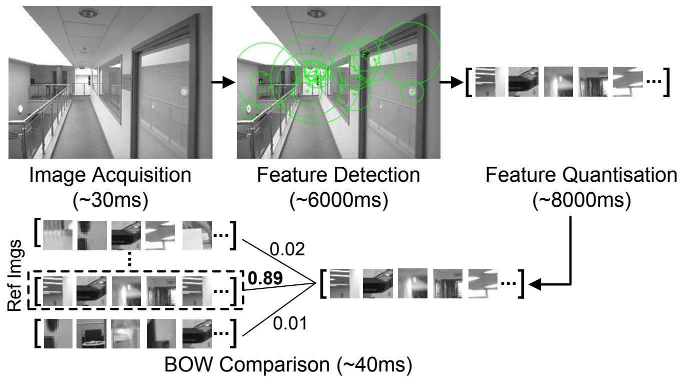


Fig. 2. Typical processing pipeline and running time (estimated on a Google Glass) of the Bag-of-Words image comparison approach [9].

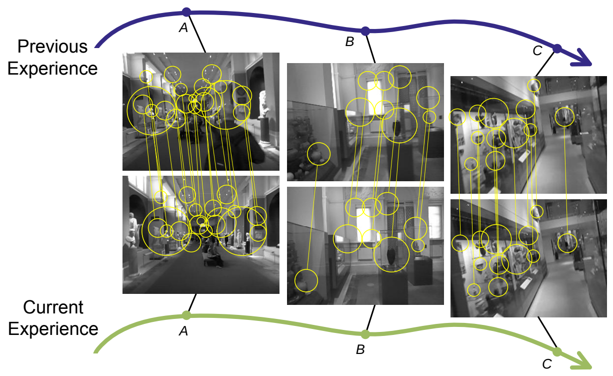


Fig. 3. Scene properties e.g. feature distribution and types of visual elements may vary significantly within a visual experience.

resource constrained platforms. For instance, Fig. 2 shows a typical pipeline of the Bag-of-Words (BOW) image processing approach used by Travi-Navi. Given an image, the detected features are quantised into a vector of visual words (i.e. the scene elements) with respect to a pre-trained vocabulary. This BOW vector is then compared against a database of reference vectors, where the likelihood that two vectors represent the same scene is determined by certain distance metric. In our experiments we find that on the off-the-shelf smart glasses, just the feature detection and quantisation steps can take more than 10s to complete, which is impossible for real-time visual positioning. Some of the existing work [7] considers offloading the computation to the cloud, but it may not be cost-effective because: a) communication channels such as WiFi/4G are not always available or stable in some environments, e.g. construction sites; and b) the delay during localisation can be high due to different network conditions. The Travi-Navi system [3] bypasses this by only sampling images sparsely for pathway identification (not localisation), but this does not exploit the full power of visual positioning.

Our previous work [8] reduces the image processing time by pruning the visual vocabulary based on mutual information between words. However, such a *global* optimisation approach treats the entire environment equally, and doesn't consider the appearance variations across different locations. For instance Fig. 3 shows an example of images when following a previous visual experience in a museum. We see that place A is a large hall with many different visual elements, while the scene at place B contains much fewer, but more distinctive features. This means the optimal visual vocabulary for place B may not work at place A, since it may fail to include enough words to describe the complex scene there. On the other hand, when comparing images at place B, it is not necessary to consider the complete visual vocabulary as in A, but only a subset would be sufficient. Also comparing to A, most of the features in B are close to the camera, which can still be detected at lower resolutions. Thus at place B we can safely configure the camera to sample low resolution images to save processing time, but not vice versa. In addition, at place C most features are clustered on the left, and thus we can just process those parts instead of full images, which is not possible at A or B.

In this paper, we aim to address the above challenges by moving away from the one-shot *teach-repeat* scheme, to a novel *lifelong learning* paradigm. The idea is that after following a visual experience across the space for several times, we can gradually learn visual processing parameters that are key to localisation success at different places, e.g. the minimum set of visual words, the lowest possible image scale, and the salient image regions. The learned knowledge is then annotated to the saved experiences as metadata, and is used to dynamically adjust the localisation algorithm when experiences are followed in the future. In this way, we can massively reduce computation on visual processing, where the positioning system only needs to process the *minimum necessary* information to stay localised, and thus make real-time visual positioning possible. Concretely, the technical contributions of this paper are:

- We propose a novel lifelong learning paradigm, which infers key knowledge on how to follow previously collected visual experiences with minimum possible computation from subsequent repetitions. The learned parameters are annotated to the experiences, and are continuously improved over time.
- We design a lightweight localisation algorithm, which dynamically adjusts its visual processing pipeline according to the annotated visual experiences. This allows us to build a positioning system that is infrastructure-free, requires little set-up effort, and runs in real-time on resource constrained mobile and wearable devices.
- We implement the proposed positioning system on various mobile phones and smart glasses, and evaluate it in two different indoor settings. Experiments show that comparing to the competing approaches, our system is able to reduce the running time up to two orders of magnitude, and achieve real-time positioning with sub-metre accuracy.

The rest of the paper is organised as follows. Sec. 2 provides an overview of the proposed approach. Sec. 3 explains how to learn optimal parameters for visual processing and annotate them to the experiences, while Sec. 4 presents the real-time localisation algorithm that takes the annotated experiences into account. Sec. 5 evaluates our system, and the related work is covered in Sec. 6. Sec. 7 concludes the paper and discusses possible future work.

2 Overview

Before presenting the proposed learning approach, in this section we first discuss our key assumptions on visual experiences and the problem of real-time localisation in Sec. 2.1, and then we describe the architecture of our positioning system in Sec. 2.2.

2.1 Model and Assumptions

Visual Experiences: A visual experience E is a chain of nodes $n_1, ..., n_M$, which contain the images captured as the user moves across the indoor space [10]. A directed edge $e_{i-1,i}$ that links

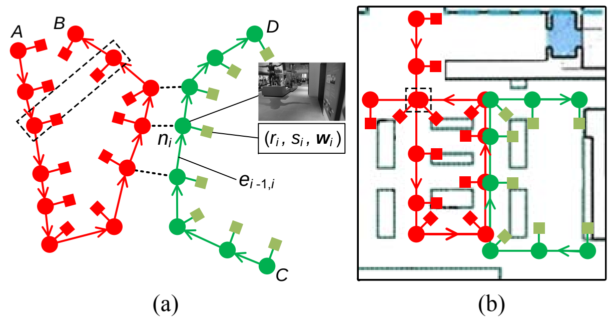


Fig. 4. (a) Two annotated experiences, where each node n_i is associated with visual processing parameters (r_i, s_i, $\boldsymbol{w}_i$). The dashed lines are co-location links, through which the user can transit from one experience to another. (b) The global embedding of the annotated experience graph over the floorplan.

two nodes n_{i-1} and n_i represents the metrical transformation between them, as shown in Fig. 4(a). In this work, $e_{i-1,i}$ is estimated using a pedestrian dead-reckoning (PDR) approach [11]. If multiple images are captured within one step, the edges between them are computed by interpolation. Conceptually, a visual experience E describes the appearance of the environment along the user trajectory towards a specific destination. Therefore when navigating to the same destination in the future, we can follow this experience E, by comparing the live images and motion measurements against those in E and work out where we are. It is also worth pointing out that an experience E doesn't have to be globally consistent, e.g. it is well known that PDR suffers from long-term drift and the generated inertial trajectories may have accumulated errors (e.g. Fig. 4(a)). However as discussed later, our system only considers *relative* localisation with respect to previous experiences, and thus as long as the user can follow those experiences locally, she can be successfully navigated to the desired destination step by step.

Visual Processing Pipeline: When following experiences, our system uses a Bag-of-Words (BOW) based [9] visual processing pipeline to process images. Without loss of generality, we use SURF [12] to extract image features, which are then quantised into vectors (i.e. bags) of visual words based on the pre-trained visual vocabulary. For instance, if the image contains a feature corresponding to a window, while the i-th word in the vocabulary represents a typical window (e.g. the average of different windows), then the i-th element of the generated BOW vector should be 1. Essentially the pipeline maps an image into a BOW vector, which describes the scene elements appear in that image, and the similarity between two images can be evaluated by the distance between their BOW vectors.

Visual Processing Parameters: At runtime, the cost of our visual processing pipeline is determined by two factors: the total volume of image pixels it has to process, and the amount of visual words to be compared with (see Fig. 2). Therefore, in this paper we consider the following parameters to configure the pipeline: a) the sampling image scale (i.e. resolution) r of the camera; b) the salient region s of the captured image given the scale r; and c) the set of key visual words $\boldsymbol{w}$ used by the pipeline to quantise image features. Intuitively, s and r together determine the cost of feature extraction step of the processing pipeline, while $\boldsymbol{w}$ governs the feature quantisation cost under the given s and r.

Annotated Visual Experience: In practice when following a previous experience, it is not necessary to use the same configuration for the visual processing pipeline throughout, since the appearance at different parts of the experience can vary significantly (as shown in Fig. 3). Therefore, we augment the visual experience E to incorporate the place dependent visual process parameters. For each node $n_i \in E$, we attach the parameters $(r_i, s_i, \boldsymbol{w}_i)$, representing the optimal configuration of visual processing pipeline when the user is at the location of n_i (as shown in Fig. 4(a)). In this way, the annotated experience E doesn't only describe the appearance of a route across the workspace, but also specifies how we should follow it in different places. The ways of creating and updating the annotated experiences will be discussed in Sec.3 in more detail.

Topometric Experience Graph: As the users continue to explore the indoor environment, our system uses a *topometric experience graph* to represent the saved experiences from different users, as shown in Fig. 4. In such a graph, each node has an Euclidean neighbourhood, but globally we assume no consistency. For example the two highlighted nodes in Fig. 4(a) are in fact at the same position (see Fig. 4(b)), but are represented differently due to the accumulated errors in inertial tracking. We also exploit the spatial overlapping between experiences by creating undirected links between nodes with similar visual appearance. Those co-location links increase the connectivity of the graph, from which one could transit between different experiences. For instance, in Fig. 4(a), to go from A to D, one could start with the experience on the left, then transit to the experience on the right via any co-location link, and follow it afterwards. Note that it is straightforward to use other sensing modalities, such as WiFi or Bluetooh beacons, to create co-location links [13], if a reliable similarity metric is provided.

Localisation with Visual Experiences: We consider *relative localisation*, where at a given time, the location of the user is specified by a pair (n_i, T). n_i is the node in the experience graph that is the closest to the current user position, and T is the user's relative displacement from n_i. Intuitively, we match the observed sensor measurements with those in the experience graph to "pin down" the user, and then use the motion data to track her accurate position with respect to matched node. Therefore in our context, localisation is not performed in a globally consistent map, but only the topometric experience graph which can be viewed as a manifold [10]. This is particularly useful in navigation scenarios, where our system can just localise the users within the experience graph and navigate them to their destinations, without the expensive process of enforcing a global Euclidean map. On the other hand, if the graph can be embedded to a consistent frame of reference, e.g. by map matching [6], localisation against the graph is equivalent to positioning within the global map (see Fig. 4(b)).

The problem tackled by this work is how to make such localisation *efficient*, and run in *real-time* on resource constrained mobile and wearable devices. To address this, we propose a positioning system that continuously learns the optimal visual processing parameters (i.e. r_i, s_i and $\boldsymbol{w}_i$) from localisation results, and annotates the learned parameters to the previous experiences. When being tasked later, our system actively tunes the visual processing pipeline according to such knowledge, to stay localised with minimum possible computation. Now we are in a position to explain the architecture of the proposed system.

2.2 System Architecture

Fig. 5 shows the architecture of the proposed positioning system, which consists of a front-end that runs on the mobile devices, and a back-end which resides on the cloud.

Front-end: The front-end localises the users with respect to the previous experiences (i.e. the experience graph) based on live

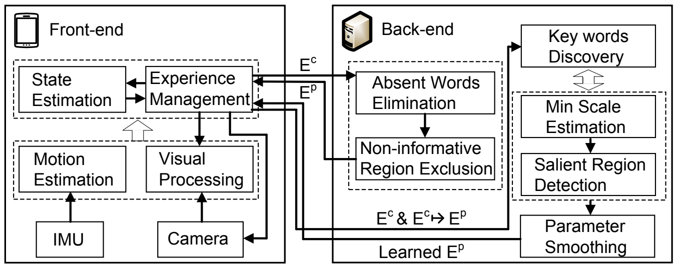


Fig. 5. The architecture of the proposed system, where the front-end runs on the user carried devices, and the back-end resides on the cloud.

sensor observations. In practice, it is possible to localise with respect to more than one experiences, i.e. the sensor observations can be matched to co-located nodes from different experiences, but for simplicity here we only localise using the best matched experience in the graph. Let E^p be the experience, and $n_i \in E^p$ be the node that the user is currently localised to. Then the live frame is passed through the visual processing pipeline, which is adjusted according to the parameters encoded in n_i. The matching results is then fused with measurements from IMUs, and the user position is determined by a state estimation algorithm. At the meantime, the front-end saves the current experience E^c by logging the live images and motion data, which will be used by the back-end later. We consider a motion-guided image sampling strategy as in [8], while the sampling rate depends on the accuracy requirement and energy budget set by the users (typically $<$1Hz). When localisation fails, i.e. the user can no longer be localised within the current experiences, the front-end pauses the state estimation process and only saves the observed sensor streams as a new experience E^n, until localisation can be reinstated. In practice, such localisation failure would occur if the user starts to explore a new route that hasn't been covered by exiting experiences, or when the appearance of a previously traversed route has changed dramatically, e.g. due to variations in lighting. Note that in our system, we tend to record dense E^n by sampling images at a much higher rate, to build an initial survey of the new environment/appearance. In typical indoor environments, this process won't happen frequently, and the experience graph tends to converge as more experiences are accumulated.

Back-end: Once the user finishes following a previous experience E^p (assuming it has been annotated with visual processing parameters), the current experience E^c and the localisation results, i.e. the mapping between nodes in E^c and E^p will be uploaded to the back-end for learning when appropriate, e.g. the device is charged and/or connected to WiFi. If a new experience E^n has been created, e.g. the user has just explored a new trajectory, the saved E^n will also be uploaded. In the former case, the back-end iteratively computes the minimum key word set $\boldsymbol{w}_i$, the optimal image salient region s_i and scale r_i, with which the correspondence between E^c and E^p can still be maintained. The learned parameters are used to update those in E^p, and are referred to by the front-end when the user is localised against E^p in the future. On the other hand, given the new experience E^n, for each node $n_i \in E^n$, the back-end computes the initial estimates of the visual processing parameters by pruning the redundant visual words and non-informative image regions (details will be discussed in Sec. 3.1). Then it assembles the annotated experience to the experience graph by exploiting co-location links (e.g. as in our previous work [13]), where the updated graph will be downloaded and used by the front-end in next localisation. In practice, the above experience annotation process runs on the cloud infrastructure or local cloudlet [7], which typically have sufficient computational power to handle the overhead. In addition, our system doesn't require real-time experience annotation or constant communication between the front-end and back-end. When the annotated experiences are ready and downloaded to the front-end, it can operate without the cloud. Therefore, in our system localisation performance won't be affected by network latency, which is very desirable in practice.

In this way the proposed system forms a feedback loop, which doesn't just *learn about* the indoor environment for once and then localise the users with this one-shot learned experiences, but also continuously *learns from* the subsequent traversals to improve itself and work smarter over time.

3 Experience Annotation

This section discusses the proposed approach of experience annotation, which continuously learns how to configure the visual processing pipeline to achieve more efficient localisation in the future. As discussed in Sec. 2.1, the computational bottleneck of visual processing is the feature extraction and quantisation steps (see Fig. 2), whose cost is determined by: a) the total volume of pixels one has to process; and b) the amount of visual words to be compared against. For a given node n_i in an experience, the former is actually the size of the salient region s_i under the image scale r_i (denoted as $|s_i|_{r_i}$), while the latter is the cardinality $|\boldsymbol{w}_i|$ of the key word set. Therefore our goal is to find the set of parameters r_i, s_i and $\boldsymbol{w}_i$, which yield the minimum possible $|s_i|_{r_i}$ and $|\boldsymbol{w}_i|$. In Sec. 3.1, we first discuss how to compute the initial estimates of the parameters given a newly created experience. Then in Sec. 3.2 we show how the parameters can be continousely optimised by learning from subsequent repetition of the experiences.

3.1 Parameter Initialisation

As discussed in Sec. 2.2, when the user explores a trajectory for the first time, or the appearance of a previously traversed route has changed significantly, the front-end creates a new visual experience E^n and upload it to the cloud when communication is available. At this stage, our system tries to compute good initial estimates of the visual processing parameters by exploiting the scene properties at different parts of E^n (as shown in Fig. 6).

3.1.1 Eliminate Absent Visual Words

In the standard visual processing pipeline, each image has to be quantised with respect to the complete visual vocabulary, which could be substantial. For instance, FAB-MAP [14] uses a vocabulary of roughly 10k words for outdoor place recognition, and our experiments considers around 4k words trained from various indoor environments. In this case, comparing against each word in the vocabulary is prohibitively expensive for resource constrained devices: as shown in Fig. 2, this needs $\sim$8s to complete on Google Glasses. However in practice, we find that different parts of the experience tend to contain very different sets of visual words. For example, in a corridor with many doors alongside, we would expect to see lots of door handles or frames; while in an atrium with stairs, the scene could be occupied by elements such as railings (as shown in the first and third image in Fig. 6). Therefore, we don't need to compare against all the words in the vocabulary, but just have to consider the words that appear within a local area.

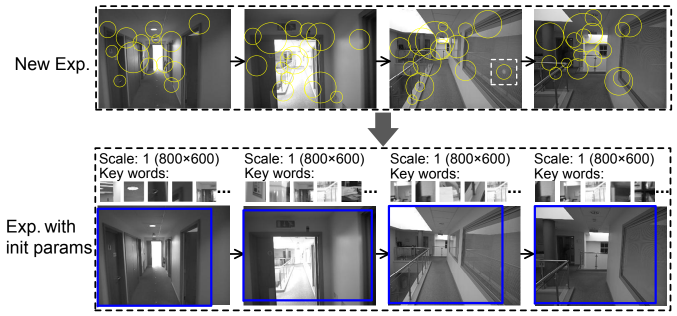


Fig. 6. For a newly created experience, the proposed system estimates the initial parameters by pruning unnecessary information.

Based on this intuition, for each node n_i in the newly created visual experience E^n, we initialise the key word set $\boldsymbol{w}_i$ as follows. We consider a sliding window of $2k$ nodes $[n_{i-k+1}, ..., n_{i+k}]$ centred at n_i. The images within the window are fed to the visual processing pipeline, where features are extracted and quantised into Bag-of-Words vectors with the original visual vocabulary. Assuming the vocabulary contains N visual words $w_1, ..., w_N$. Then the window of images can be represented as a $2k \times N$ matrix F. Each row $F(l,:)$ represents a particular image, and the n-th element $F(l,n)$ is the frequency that word w_n appears in that image. Finally, whether the word w_n should be included in the key word set $\boldsymbol{w}_i$ is given by the indicator function:

$$\mathbb{1}(w_n) = \begin{cases} 1, & \sum_{l=i-k+1}^{i+k} F(l,n) > 0 \\ 0, & \sum_{l=i-k+1}^{i+k} F(l,n) = 0 \end{cases} \tag{1}$$

This effectively rules out the visual words that never present within the neighbourhood of an experience node n_i, and selects a much smaller set of key words that have to be compared against, as shown in Fig. 8(a) and (b).

3.1.2 Prune Non-informative Image Regions

In addition to removing the unseen visual words, at this stage our system also tries to find a smaller salient region s_i for the image stored in node n_i. Note that here we keep the image scale r_i unchanged, because for now we are unable to determine the minimum possible r_i for successful localisation with respect to the experience E^n (we will show how to learn the optimal scale r_i with more experiences in the next section). Concretely, for each node n_i our system tries to locate the image patches containing no features (e.g. the slice of white wall in the first image of Fig. 6), or only the *non-informative features* (as discussed below), and eliminate those patches from the salient region s_i.

Let $\mathfrak{f}_k$ be an extracted feature of the image in node n_i. During the image quantisation step, for each word w_n in the vocabulary, we compute the distance between feature $\mathfrak{f}_k$ and word w_n. Then $\mathfrak{f}_k$ is quantised to the word with the smallest distance, indicating that $\mathfrak{f}_k$ belongs to the same type of visual element represented by that word. Let $d(\mathfrak{f}_k)$ be the smallest distance when quantising feature $\mathfrak{f}_k$. $d(\mathfrak{f}_k)$ indicates how well the feature $\mathfrak{f}_k$ can be described with the current vocabulary. In practice, large $d(\mathfrak{f}_k)$ means that the vocabulary doesn't contain visual elements similar to the feature $\mathfrak{f}_k$, i.e. we are not sure what $\mathfrak{f}_k$ represents. For instance, the highlighted feature in the third image of Fig. 6 is the reflection of a light on the window, which can't be well represented by the current visual vocabulary. As a result, such a feature won't contribute to

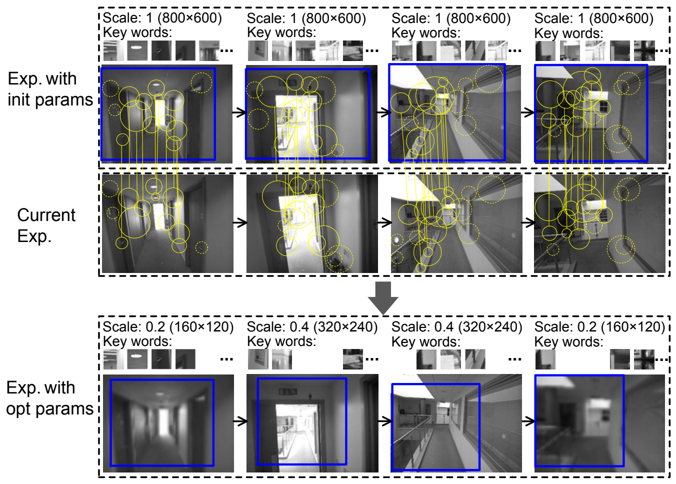


Fig. 7. Given the localisation results, our system updates the experience annotations by learning the optimal parameters for visual processing.

the BOW matching process but could introduce noises. Therefore, our system prunes those features and set the initial salient region s_i according to the bounding box of the rest visual features (as shown in Fig. 6). In practice, we typically set the initial s_i to be slightly larger than the bounding box, to account for potential view point changes when following the experience E^n. In addition, if the computed bounding box is too small comparing to the image dimension (in our experiments we consider <50%), e.g. when images contain very few informative features due to blurriness, we set the initial s_i as the original image size for now and leave it to the later learning stage.

After the above initialisation process, the newly created experience E^n has been annotated with the initial estimates of visual processing parameters. As discussed in Sec. 2.2, this annotated experience will be assembled to the experience graph through co-location links, and downloaded to the user devices when it is needed for future localisation.

3.2 Lifelong Parameter Learning

With the initial estimates of the visual processing parameters, our system is able to exclude some unnecessary information during image processing, e.g. the redundant visual words or non-informative images regions. This can already reduce the runtime cost when following the experiences. However in many cases, we could further improve performance by learning from the subsequent repetitions of the previous experiences, just like what humans would do. For instance, when we first follow someone along a trajectory, we tend to stay alert throughout and watch out for as many visual clues possible. However after a few more traversals, we become more familiar with the route, and will discover place-dependent information that is vital for localisation/navigation success, e.g. in some places we may only need to pay attention to a few landmarks to keep on the right track. Follow this intuition, our system employs a *lifelong learning* paradigm, which keeps calibrating the optimal visual processing parameters through continued use.

Let us assume the user has followed a previously annotated experience E^p, and the front-end of our system has saved the live sensor observations during this localisation as the current experience E^c. In this context, the localisation results can be viewed as the mapping between the nodes of current and previous

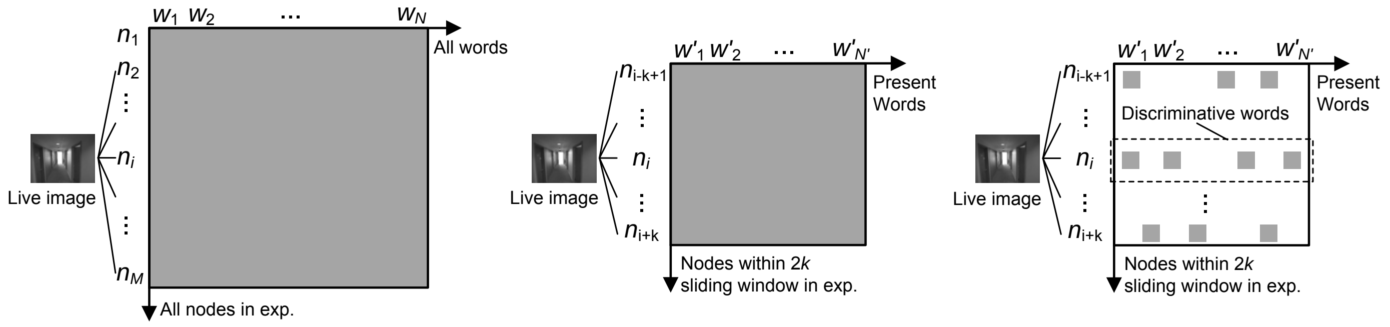


Fig. 8. During localisation, a live image can be compared with (a) all nodes using the complete vosual vocabulary (stanrdard approach); (b) a sliding window of nodes using only the present words (after parameter initialisation); and (c) the most discriminative words (after parameter learning).

experiences $E^c \mapsto E^p$. Note that here the visual processing parameters encoded in E^p can be either computed by the initialisation step as above, or from the previous learning iteration.

In our case, the goal of the learning process is to compute the optimal visual parameters $(r_i, s_i, \boldsymbol{w}_i)$ given the known correspondence between experiences $E^c \mapsto E^p$, which are the solution of the following constrained optimisation problem:

$$\begin{aligned} \underset{r_i, s_i, \boldsymbol{w}_i}{\text{minimize}} \quad & |s_i|_{r_i}, |\boldsymbol{w}_i| \\ \text{subject to} \quad & p(h_j \mapsto n_i | E^p, r_i, s_i, \boldsymbol{w}_i) \geq \epsilon, \\ & h_j \in E^c, n_i \in E^p \end{aligned}$$

$p(h_j \mapsto n_i | E^p, r_i, s_i, \boldsymbol{w}_i)$ is the likelihood that the image in node h_j matches that of n_i given the current parameters, and is evaluated with the FAB-MAP [14] approach. The constraint requires the matching likelihood of h_j to n_i exceed a threshold ϵ. In out implementation we typically require $\epsilon > 0.5$, so that in the majority cases the node h_j should be correctly matched to node n_i. Intuitively, $|s_i|_{r_i}$ and $|\boldsymbol{w}_i|$ in the objective function are correlated. Images at lower scale or with smaller salient region (i.e. smaller $|s_i|_{r_i}$) tend to contain fewer visual features, and thus could require a sparser key word set to quantise. On the other hand, if just a few words are essential for correct matching, we can work at lower image scales, and/or only on image patches corresponding to those key words. Therefore, the proposed system optimises the two parts of the objective function iteratively. In each iteration, we first find the set of key visual words $\boldsymbol{w}_i$ that are vital for successful matching (Sec. 3.2.1). Then given the computed $\boldsymbol{w}_i$, we estimate the salient region s_i together with the suitable scale r_i (Sec. 3.2.2). In the next iteration, the estimated r_i and s_i are used to evaluate a new key word set $\boldsymbol{w}_i$ accordingly. This process terminates when the parameters $(r_i, s_i, \boldsymbol{w}_i)$ converge, or a certain number of iterations has been reached. In practice, it is possible that before the learning process the matching likelihood $p(h_j \mapsto n_i | E^p, r_i, s_i, \boldsymbol{w}_i)$ is already below the threshold ϵ. In those cases, our system resets the parameters to their initial values and starts learning from there. Finally, the learned parameters are smoothed within a local neighbourhood to improve robustness and account for spatial correlations (Sec. 3.2.3). Now we are in a position to explain the optimisation steps in more detail.

3.2.1 *Discover the Most Discriminative Visual Words*

For each node n_i in the previous experience E^p, the key word set $\boldsymbol{w}_i$ has been initialised as the words that appear within its neighbourhood of $2k$ nodes, as discussed in Sec. 3.1.1 (see Fig. 8(b)). Given the known mapping $h_j \mapsto n_i$ (from the localisation results), our system further reduces $\boldsymbol{w}_i$, to only include the most discriminative words, with which the images in nodes h_j and n_i can be matched. For instance, in our experiments we found visual elements representing the carpet tiles are common in most of places, which contribute very limited discriminative power when matching images, and thus should be excluded from the key word set. Therefore, we wish to find a *minimum* subset of the current key words $\boldsymbol{w}_i$ so that the mapping $h_j \mapsto n_i$ holds. In this process, our system also considers a sliding window of $2k$ nodes, and works as follows. Firstly, with the current parameters $(r_i, s_i, \boldsymbol{w}_i)$, the images within the window are processed into Bag-of-Words (BOW) vectors. Assuming the current key word set contains N' visual words $w'_1, ..., w'_{N'}$. Like in the previous Sec. 3.1.1, here we also consider a $2k \times N'$ matrix F' to represent the images in BOW format, whose elements are the frequency of words. For a given word w'_n, we define its discriminative power within the $2k$ window as:

$$H(w'_n) = -\sum_{l=i-k+1}^{i+k} F'(l,n) \ln F'(l,n) \tag{2}$$

In fact, if we normalise the n-th column $F'(:,n)$ into a distribution, the above $H(w'_n)$ is essentially its *information entropy*. Intuitively, the word w'_n that only appears in a few images is more promising to distinguish them from the others. In this way, by ranking the entropy $H(w'_n)$ (i.e. discriminative power of words), we obtain a ranked word set $\vec{\boldsymbol{w}}_i$.

Finally, our system evaluates a new key word set $\boldsymbol{w}'_i$ based on the computed $\vec{\boldsymbol{w}}_i$. Conceptually, this can be done by iteratively adding words to $\boldsymbol{w}'_i$, until the node h_j can be reliably matched to n_i. To speed up this process, we initialise $\boldsymbol{w}'_i$ as the first half of ranked set $\vec{\boldsymbol{w}}_i$ (those are more informative), while the rest is considered as a candidate set. Then in each iteration, we use the current $\boldsymbol{w}'_i$ to compare the image of h_j against those within the sliding window, and evaluate the matching likelihood $p(h_j \mapsto n_i)$. If $p(h_j \mapsto n_i)$ exceeds ϵ, we reduce $\boldsymbol{w}'_i$ by half in the next iteration; otherwise we move the first half of the words in the candidate set to $\boldsymbol{w}'_i$. After at most $\log_2 |\vec{\boldsymbol{w}}_i|$ iterations, the desired $\boldsymbol{w}'_i$ can be computed, which contains the minimum amount of words to support the known mapping $h_j \mapsto n_i$ within the neighbourhood of $2k$ nodes. By applying the above process to each $n_i \in E^p$, our system learns the most informative visual elements across different segments of the previous experience E^p. Therefore in future localisation, it only needs to query a very sparse key word set when processing the live image frames (as

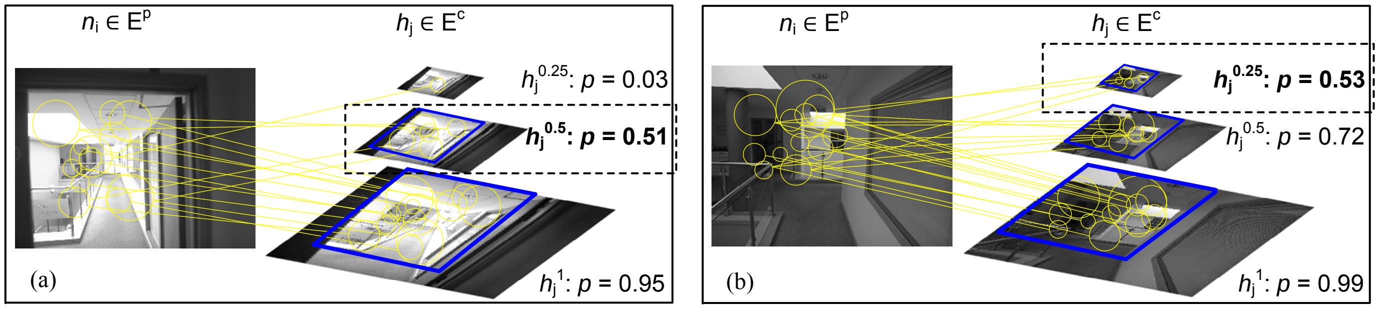


Fig. 9. Matching results of image pyramids for cases where dominating features are (a) far away from; and (b) close to the camera. Blue bounding boxes illustrate the estimated salient regions at different layers.

shown in Fig. 8(c)). In Sec. 5, we will show that comparing to the standard approach, using the minimum key words sets in localisation could reduce up to 80% of feature quantisation cost.

3.2.2 Detect Salient Regions at Multiple Scales

Now we show how to further minimise the total amount of pixels $|s_i|_{r_i}$ to be processed given the current key word set $\boldsymbol{w}_i$. $|s_i|_{r_i}$ is a function of the sampling image scale r_i and the salient region s_i, and has direct impact on the cost of feature extraction and quantisation. Intuitively, r_i indicates the level of detail one should consider, e.g. if most of the visual features are close to the camera (see Fig. 9(b)), it would be sufficient to sample images at lower scales to maintain the correct mapping. On the other hand, under a fixed scale r_i, the dominating visual elements may be well clustered within certain salient region s_i, e.g. as shown in Fig. 9(b), most of the informative features are within the top left part of the image. Therefore, if we assume the device's point of view remains relatively stable, when localising against previous experiences, it is sufficient to sample live images at the lowest possible scales and only process the smallest salient regions.

Our system considers a progressive approach to evaluate the optimal r_i and s_i for each $n_i \in E^p$. Let $h_j \in E^c$ be the node matched to n_i. We first create an image pyramid for h_j by down-sampling at different scales. Fig. 9 shows an example of image pyramids with three layers, where the lowest layer h_j^1 contains the original image (scale 1), and top two layers contain images at scale 0.5 and 0.25 respectively (i.e. 1/4 and 1/16 in size of the original image). In practice, the scales of the pyramid are determined by the camera hardware (e.g. limited by the supported sampling resolutions), and the number of layers can be tuned for different environments. Then from h_j^1 upwards, images at different scales are passed through the visual processing pipeline, and compared with images in the previous experience E^p. To capture the scene variations in different parts of E^p, we also consider a sliding window of $2k$ nodes centred at n_i. In addition, our system only uses the learned key word set $\boldsymbol{w}_i$ for image quantisation, where visual features do not appear in $\boldsymbol{w}_i$ are pruned.

At the layer with scale r, if the likelihood $p(h_j^r \mapsto n_i)$ exceeds the threshold ϵ, we further try to estimate the salient image region. Concretely, our system initialises the candidate salient region s in the same way as discussed in Sec. 3.1.2, and then reduces its size by removing features in s iteratively. Let $\mathfrak{f}_1, ..., \mathfrak{f}_K$ be the set of features left in the current iteration. For simplicity, we assume a feature $\mathfrak{f}_k$ can be represented as an image patch (e.g. the circles in Fig. 9), and the current salient region s is the bounding box containing all the K features. For each feature $\mathfrak{f}_k$, we evaluate the gain and residual if it is removed from the current feature

Algorithm 1 Salient region detection

```
1:  Set salient region $s$ as the bounding box of all features
2:  while $s$ is not minimum do
3:      Set max gain $\mathcal{G}_{\max} = 0$; feature to be removed $\mathfrak{f}^* = \phi$
4:      for each feature $\mathfrak{f}_k$ do
5:          Evaluate the gain $\mathcal{G}(\mathfrak{f}_k)$ and residual $\mathcal{R}(\mathfrak{f}_k)$
6:          if $\mathcal{R}(\mathfrak{f}_k) \geq \epsilon$ and $\mathcal{G}(\mathfrak{f}_k) > \mathcal{G}_{\max}$ then
7:              Set $\mathcal{G}_{\max} = \mathcal{G}(\mathfrak{f}_k)$; $\mathfrak{f}^* = \mathfrak{f}_k$
8:          end if
9:      end for
10:     if $\mathfrak{f}^* \neq \phi$ then
11:         Remove feature $\mathfrak{f}^*$ and set the new $s = s[-\mathfrak{f}^*]$
12:     else
13:         Return the current $s$
14:     end if
15: end while
```

set. Let $s[-\mathfrak{f}_k]$ be the hypothetical bounding box if feature $\mathfrak{f}_k$ is removed. We define the gain of removing $\mathfrak{f}_k$ as the reduced amount of pixels between the hypothetical and current bounding boxes $\mathcal{G}(\mathfrak{f}_k) = |s| - |s[-\mathfrak{f}_k]|$ (Line. 5 in Algo. 1). On the other hand, the residual $\mathcal{R}(\mathfrak{f}_k)$ of excluding $\mathfrak{f}_k$ is defined as the mapping likelihood evaluated using features without $\mathfrak{f}_k$. Then we loop over all features and try to remove the one with the highest possible gain, whose residual is still beyond the threshold ϵ. If such a $\mathfrak{f}_k$ exists, the salient region is updated to $s[-\mathfrak{f}_k]$ and we proceed to the next iteration. Otherwise the current s is already minimum, and the algorithm terminates. The detailed algorithm of salient region detection is shown in Algo. 1.

In this way, our system processes each layer of the image pyramid and stops when it reaches the highest layer where the mapping likelihood exceeds ϵ. This means there is no scope to further reduce $|s_i|_{r_i}$ any more, and the learned s_i and r_i are considered to be optimal. In practice, the estimated s_i and r_i could vary across different parts of the experience. For instance, in Fig. 9(a) the estimated salient region is at scale 0.5, while that in Fig. 9(b) is at scale 0.25 ($\sim$4 times smaller). This is because in the scene of Fig. 9(a), most of the features are quite far away from the camera, and would disappear when considering lower scales. On the other hand, in Fig. 9(b) most of the features are relatively close, and thus images at lower scales can still be reliably matched.

3.2.3 Smooth the Learned Parameters

With the learned parameters, at node n_i in future localisation the complexity of feature extraction can be reduced by a factor of $|s_i|_{r_i} / |I|$, where $|s_i|_{r_i}$ is the size of the learned salient region s_i

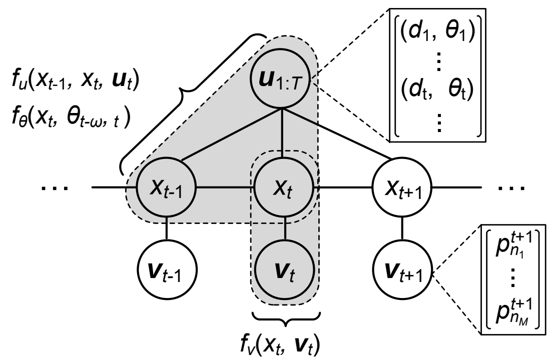

$f_u(x_{t-1}, x_t, \boldsymbol{u}_t)$
$f_\theta(x_t, \theta_{t-\omega,t})$
$\boldsymbol{u}_{1:T}$
(d_1, θ_1) ⋮ (d_t, θ_t) ⋮
x_{t-1}
x_t
x_{t+1}
$\boldsymbol{v}_{t-1}$
$\boldsymbol{v}_t$
$\boldsymbol{v}_{t+1}$
$p^{t+1}_{n_1}$ ⋮ $p^{t+1}_{n_M}$
$f_v(x_t, \boldsymbol{v}_t)$

Fig. 10. The CRF model used in the proposed system.

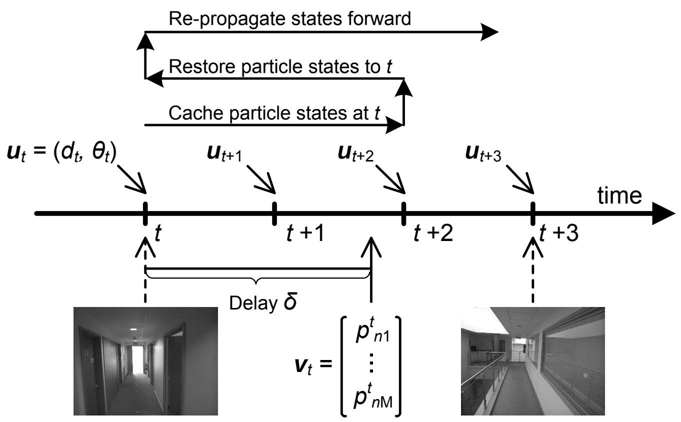

Re-propagate states forward
Restore particle states to t
Cache particle states at t
$\boldsymbol{u}_t = (d_t, \theta_t)$
$\boldsymbol{u}_{t+1}$
$\boldsymbol{u}_{t+2}$
$\boldsymbol{u}_{t+3}$
time
t
$t+1$
$t+2$
$t+3$
Delay δ
$\boldsymbol{v}_t = [p^t_{n1} \; \vdots \; p^t_{nM}]$

Fig. 11. The proposed system handles the delayed visual measurements by rolling back to particle states when the images were taken, and re-propagating the states with the user motion observed afterwards.

under optimal scale r_i, and $|I|$ is the original image size. Similarly, the complexity of feature quantisation can also be reduced at least by a factor of $|\boldsymbol{w}_i| \,/\, N$, where $|\boldsymbol{w}_i|$ is the number of words in the learned vocabulary, while N is the size of the initial vocabulary.

However, as discussed in Sec. 2.2, when following a previous experience E^p, to reduce energy consumption the current experience E^c saved by the front-end typically contains sparser nodes than E^p. This means in one learning iteration we could only update the parameters in some of the nodes in E^p. In addition, although those learned parameters are considered to be optimal for localisation, they reduce the information quite aggressively. In practice we want to increase the stability of our system, and avoid adjusting the visual processing pipeline too often. Therefore, our system also applies a smoothing process at the end of each learning iteration.

Let us consider the $2k$ nodes centred at n_i in the experience E^p. Suppose that through the learning process, we have updated the parameters in a subset of nodes N_i^{new} within the $2k$ window, while the parameters associated with the rest of the nodes N_i^{old} remain unchanged. Let W_i^{new} be the union of the key words of the newly updated nodes N_i^{new}, while W_i^{old} be the set of words that appear in the $2k$ window but not in W_i^{new}. We first let the key word set $\boldsymbol{w}_i$ of the node n_i to be W_i^{new}, and then add the top $q\%$ words in W_i^{old} based on how frequent they appear. In this way, we guarantee that the key words discovered through the learning iteration is included, while also keep some common key words appeared in the neighbourhood. For the image scale r_i and salient region s_i, we consider a weighted voting/average scheme within the $2k$ window. We typically assign more weight to the newly learned parameters, i.e. those associated with nodes in N_i^{new}, and then use a Gaussian kernel to take the spatial correlations into account. Therefore with the smoothed parameters, when localising the users with respect to the previous experiences, the proposed system is less prune to environmental dynamics e.g. view point changes caused by head movement, and can achieve better trade-off between computational efficiency and robustness.

4 Localisation with Annotated Experiences

4.1 Conditional Random Fields (CRFs)

In this section, we present the design and implementation of the proposed localisation algorithm, which is used by the front-end of our system to position the users with respect to the previously annotated experiences, as discussed in Sec. 2. Let us assume that a user is following an annotated experience E^p, which has been downloaded to her mobile device already. To position the users in real-time on resource constrained mobile and wearable devices, the localisation algorithm has to be extremely lightweight, and able to cope with the delay of visual processing (see Fig.11). To address this, our system models the position of user with respect to the previous experience E^p as the latent states, and considers a delay-tolerant sequential state estimator to fuse the inertial and visual data. In particular, we consider the undirected Conditional Random Fields (CRFs), because they are more flexible in handling correlated measurements from heterogeneous sensing modalities.

Latent States: As discussed in Sec. 2.1, the position of the user x_t can be represented as a pair (n_i, T) where n_i is the node in E^p that is the closest, and T is the relative transformation from the position of n_i to x_t. In practice, T can be estimated from the motion/odometry data, and thus localisation against the previous experience E^p can be cast into that of finding the matching nodes in E^p that can best explain the sensor measurements. Therefore, in this paper we define the state space as the set $S = \{n_1, ..., n_M, \phi\}$, where $n_1, ..., n_M$ are the nodes of the experience E^p, and ϕ is an empty node. The event $x_t = n_i$ indicates that the current user position x_t is matched to node n_i (subject to transformation T), while $x_t = \phi$ means localisation failure, i.e. the user can't be localised with respect to E^p. In practice, for big enviornments the state space S can be large, which would have a negative impact on localisation performance. However this can be mitigated by using other sensing modalities such as WiFi fingerprints to first estimate a coarse location of the user, and then positioning her within the identified subgraph.

Motion Measurements: At discrete time t, our motion engine generates $\boldsymbol{u}_t = (d_t, \theta_t)$, which is the displacement and heading change of the user since time $t-1$, as shown in Fig. 10. We consider a zero crossing detector with linear stride length model to estimate steps from the acceleration domain. For heading estimation, unlike most of the existing pedestrian dead-reckoning (PDR) solutions that require absolute heading, we only care about the *relative heading* with respect to the previous timestamp. To this end, our system uses an unscented Kalman filter to fuse the magnetic and gyroscope signals efficiently. It is well known that such a lightweight approach is not robust to abrupt device movements (e.g. rotating head when wearing smart glasses) and long term sensor drift [11]. However, our system is inherently resilient to those noises, since as shown later the estimated motion is only compared with small trajectory segments of the previous experiences, and the accumulated error is likely to be corrected by visual measurements later on.

Visual Measurements: Unlike the existing systems such as Travi-Navi, our visual processing pipeline is configured dynamically according to the current state belief and experience annotations. Let $x_t = n_i$ be the predicted state at time t. We retrieve the parameters $(r_i, s_i, \boldsymbol{w}_i)$ annotated to node n_i, and task the camera to take an image at scale r_i. Then only features appear within the salient region s_i are extracted, and quantised into a Bag-of-Words vector according to the key word set $\boldsymbol{w}_i$. Finally, the BOW vector is compared with the images of the previous experience E^p. Therefore, the visual measurement at time t allows us to derive a distribution $\boldsymbol{v}_t = [p^t_{n_1}, ..., p^t_{n_M}]$ (as shown in Fig. 10), where $p^t_{n_k}$ is the likelihood that the captured image matches n_k in E^p.

4.2 Feature Functions

In our model the conditional dependencies between states and observations can be factored as products of potentials:

$$p(x_{1:T}|\boldsymbol{u}_{1:T}, \boldsymbol{v}_{1:T}) = c^{-1} \cdot \prod_{t=2}^{T} \Psi(x_{t-1}, x_t, \boldsymbol{u}_{1:T}, \boldsymbol{v}_{1:T}) \quad (3)$$

c is a normalising constant, which integrates over all state sequences: $c = \int \prod \Psi(\cdot) dx_{1:T}$. The potentials Ψ is the log-linear combination of feature functions $\boldsymbol{f}$:

$$\Psi(x_{t-1}, x_t, \boldsymbol{u}_{1:T}, \boldsymbol{v}_{1:T}) = \exp\{\boldsymbol{w} \cdot \boldsymbol{f}(x_{t-1}, x_t, \boldsymbol{u}_{1:T}, \boldsymbol{v}_{1:T})\} \quad (4)$$

where a feature function $f \in \boldsymbol{f}$ specifies the degree to which the observed sensor data supports the belief of the consecutive states. The weights $\boldsymbol{w}$ indicate the relative importance of different features functions, and the way of learning $\boldsymbol{w}$ will be discussed later in this section. We consider the following feature functions:

Instant Motion: This feature function models how the currently observed user motion supports the transition between two consecutive states, and is defined as:

$$f_u(x_{t-1}, x_t, \boldsymbol{u}_t) = -(\boldsymbol{u}_t - \hat{\boldsymbol{u}}_{x_{t-1}:x_t})^{\mathrm{T}} \Sigma_{\boldsymbol{u}}^{-1} (\boldsymbol{u}_t - \hat{\boldsymbol{u}}_{x_{t-1}:x_t}) \quad (5)$$

where $\boldsymbol{u}_t$ is the motion measurement from $t-1$ to t, and $\hat{\boldsymbol{u}}_{x_{t-1}:x_t}$ is the noise-free motion between states x_{t-1} and x_t, which is derived directly from the previous experience E^p. $\Sigma_{\boldsymbol{u}}$ is the covariance, which captures the important correlations between user displacement and heading changes, e.g. people typically slow down when turning at corridors.

Accumulated Heading Change: This feature function checks the compatibility between state x_t and the observed heading changes over a time window $[t-\omega, t]$:

$$f_\theta(x_t, \theta_{t-\omega,t}) = \ln \frac{1}{\sigma_\theta \sqrt{2\pi}} - \frac{(\theta_{t-\omega:t} - \hat{\theta}_{\hat{x}_{t-\omega}:x_t})^2}{2\sigma_\theta^2} \quad (6)$$

where $\theta_{t-\omega:t}$ is the observed change in heading from time $t-\omega$ to t. $\hat{\theta}_{\hat{x}_{t-\omega}:x_t}$ is the heading change computed between the previously estimated state $\hat{x}_{t-\omega}$ and current state x_t, and σ_θ is the variance of heading changes from the covariance matrix $\Sigma_{\boldsymbol{u}}$ in Eqn. (5). Unlike f_u which only cares about instant user motion, here f_θ correlates the current state with a longer history of previous heading changes. Therefore, f_θ tends to reward the x_t with a neighbourhood that matches the "shape" of the observed user motion, and is especially discriminative when the user turns.

Visual Matching: The final feature function f_v describe how the observed image at time t supports the current state x_t. Recall that the visual measurement $\boldsymbol{v}_t$ is a distribution $[p^t_{n_1}, ..., p^t_{n_M}]$, where $p^t_{n_i}$ is the likelihood that the image captured at t matches the node n_i in the previous experience E^p. Then we directly define f_v as:

$$f_v(x_t, \boldsymbol{v}_t) = p^t_{x_t} \quad (7)$$

which is the likelihood of the state x_t according to the current visual matching result.

Algorithm 2 State Estimation with Delayed Measurements

```
1:  Initialisation: sample a set of particles from the initial state distribution
2:  while a new motion measurement $\boldsymbol{u}_t$ arrives do
3:      for each particle do
4:          Prediction: predict particle state by sampling from $\exp\{f_u(x_{t-1}, x_t, \boldsymbol{u}_t)\}$
5:          Weighting: update particle weights according to $\exp\{f_\theta(x_t, \boldsymbol{u}_{t-\omega:t})\}$
6:      end for
7:      Re-sample: generate new particles based on their weights
8:      if an image has been captured then
9:          Cache the current particle states
10:     end if
11:     if a visual measurement $\boldsymbol{v}_{t'}$ is available ($t' < t$) then
12:         Rollback: Restore particle states cached at time $t'$
13:         Weighting: update particle weights according to $\exp\{f_v(x'_t, \boldsymbol{v}_{t'})\}$
14:         Re-propagate: update particle states until $t$, as shown from Line 3 to Line 7
15:     end if
16: end while
```

4.3 State Estimation

Initialisation: We consider a particle filter algorithm for state estimation on the above CRF model, which can handle complex distributions, and scales well when the state space grows, e.g. as more experiences are accumulated. In practice, we bootstrap our algorithm when a sufficient number of consecutive images can be strongly matched to the previous experiences. In some cases if the experience graph has been embedded to a global map, the initial state may be determined by certain external signals or landmarks, e.g. the card swipe event at the main entrance. Our algorithm randomly draws a set of particles according to the initial state, and iteratively performs the following steps as the user moves.

Incorporating Motion Features: Firstly, given the observed user motion $\boldsymbol{u}_t$, for each particle we propagate its state by sampling from the feature function $f_u(x_{t-1}, x_t, \boldsymbol{u}_t)$, which evaluates the consistency between the observed motion $\boldsymbol{u}_t$ and the expected $\hat{\boldsymbol{u}}_{x_{t-1}:x_t}$ given the consecutive states (see Eqn. (5)). Then the particles are weighted according to $f_\theta(x_t, \theta_{t-\omega,t})$, where those agree more with the local shape of the observed user trajectory are favoured. Finally, the particles are re-sampled according to their weights.

Processing Delayed Visual Measurements: In our context a visual measurement can be delayed due to the cost of visual processing. For instance, as shown in Fig. 11, the image captured at t takes time δ to be processed, i.e. the visual measurement $\boldsymbol{v}_t$ is only seen by the state estimation algorithm at time $t+\delta$, and by then the user has moved. To cope with such delay, we cache the particle states when capturing the image at t. Once the visual measurement $\boldsymbol{v}_t$ is ready, our system waits until the next motion measurement comes (at $t+2$ in Fig. 11), and then

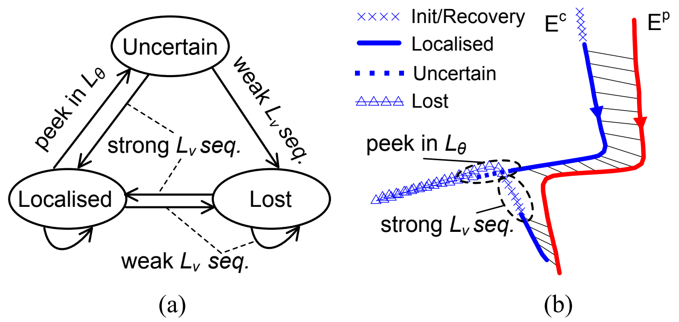


Fig. 12. (a) The decision model used by our system to handle localisation failure. (b) An example where the user tries to explore a new route.

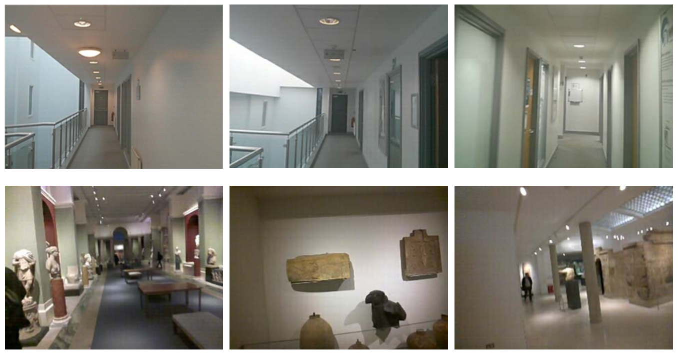


Fig. 13. Two different experiment sites. Top: the office building, where left two images are taken at two different floors. Bottom: the museum.

restores the cached particles at t. At that point, the particles are re-weighted according to the feature function f_v, and then re-propagated forward with all the motion measurements (until $t+2$ in Fig. 11) as discussed above. In this way, by periodically rolling back, our state estimation algorithm tolerates the processing delay of images, and fuses motion and visual measurements efficiently. The detailed state estimation algorithm is shown in Algo. 2.

Learning Model Parameters: In the above state estimation algoritm, the particles are weighted based on both motion and visual feature functions. In the proposed CRF model, the parameter $\boldsymbol{\omega}$ (see Eqn. 4) indicates the relative importance of different features, and is learned from the data using ground truth iteratively. Concretely, in each iteration we randomly pick a training sequence with ground truth states $\boldsymbol{x}^*$, motion measurements $\boldsymbol{u}$ and visual measurements $\boldsymbol{v}$. Then we use current parameter $\boldsymbol{\omega}$ to estimate the posterior state sequence $\hat{\boldsymbol{x}}$ as in Algo. 2, and compute the values of feature functions $\boldsymbol{f}(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{v})$. On the other hand, we also evaluate the feature values using the ground truth as $\boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{v})$. The difference $\Delta \boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{v}) - \boldsymbol{f}(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{v})$ is used to update the parameter as $\boldsymbol{\omega}' = \boldsymbol{\omega} + s\Delta \boldsymbol{f}$, where s is learning rate. Then we use the computed $\boldsymbol{\omega}'$ to re-run the state estimation process. If the localisation error exceeds certian threshold, we reduce the learning rate s by half, and estimate a new $\boldsymbol{\omega}'$ again, otherwise we terminate this iteration. We repeat this training process until the new parameter $\boldsymbol{\omega}'$ converges or certain iterations have been reached.

4.4 Handle Localisation Failure

Our system declares localisation failure when the user can no longer be localised with respect the current experience graph. In practice, this may be caused by a) the user gets lost or starts to explore a new path; or b) the current appearance of a previously traversed route has changed significantly. It detects this with a decision model (as shown in Fig. 12), by continuously monitoring the following two variables over a sliding window $[t-\omega, t]$:

$$L_\theta = 2\sigma_\theta^{-2}(\theta_{t-\omega:t} - \hat{\theta}_{\hat{x}_{t-\omega}:\hat{x}_t})^2 \tag{8a}$$
$$L_v = [\max(\boldsymbol{v}_{t-\omega}), ..., \max(\boldsymbol{v}_t)] \tag{8b}$$

L_θ describes the difference between the observed heading change $\theta_{t-\omega:t}$ and that evaluated from the estimated states $\hat{\theta}_{\hat{x}_{t-\omega}:\hat{x}_t}$ since time $t-\omega$. σ_θ is the variance as in Eqn. 6. $L_{\boldsymbol{v}}$ is the array of maximum image matching likelihood within the time window.

When L_θ rises over a certain threshold, it is likely that the user has made a turn which is not present in the previous experience E^p that she is currently following, or vice versa. In this case, our system raises an alert and watch $L_{\boldsymbol{v}}$ for further confirmation. If no consecutive strong image matchings can be found, i.e. $L_{\boldsymbol{v}}$ keeps low, localisation failure is confirmed. This means the live images are very different from those in the experience E^p, i.e. now the user is exploring a route that hasn't been traversed before. On the other hand, if we directly observe low $L_{\boldsymbol{v}}$ sequences, our system also declare localisation failure since the current appearance of the environment is significantly different from the previous experiences. In both cases, we pause state estimation, and save the current sensor observations as a new experience E^n (as discussed in Sec 2.2). When the system observes a sequence of consecutive strong image matchings, it believes that the user is back to the previous experience E^p, and reinstates the state estimation process. In this way, our system handles localisation failure gracefully, and continues to accumulate a more comprehensive representation of the workspace.

5 Evaluation

5.1 Experiment Setup

Sites and Participants: The proposed approach is evaluated in two different indoor settings: an office building and a museum. The office site is a four-storey building with similar layout and appearance at each floor (roughly $65 \times 35m^2$), as show in the top row of Fig. 13. Note that the left two images are taken at different locations across two floors. The museum site is much bigger in size ($\sim 110 \times 55m^2$), and has lots of open space and complex objects such as shelves and statues, as shown in the bottom row of Fig. 13). We recruited five participants of different genders, heights and ages, and asked them to walk normally in both sites. During the experiments, the participants wore smart glasses, and held mobile phones in their hands (cameras facing forward) while walking. In our experiments, the cameras of the glasses and mobile phones were facing towards the moving direction for most of the time. However, this is not a restriction of the system, since if the device orientation changes significantly, our system will create new experiences to capture the appearance of the environment from new angles, which can be used in subsequence localisation. The participants have repeated a set of trajectories for several times, where we randomly select a subset (across different users) to form experience graph, and use the rest for testing.

Implementation and Devices: The back-end of our system is built under Linux 3.19, and runs as a deamon process on a Ubuntu 14.04 server. The front-end is implemented under Android systems ($\geq$ 4.4), and has been tested on multiple mobile devices, including Google Glasses, Nexus 4, HTC One M8 and Nexus 6. These devices differ greatly in terms of hardware specifications and

TABLE 1
Hardware specs and computational capability of different devices.

Device	CPU	RAM	MFLOPS
Google Glass	Dual core @ 1.0GHz	1GB	53.13
Nexus 4	Quad core @ 1.5GHz	2GB	137.21
HTC One M8	Quad core @ 2.3GHz	2GB	311.95
Nexus 6	Quad core @ 2.7GHz	3GB	606.29

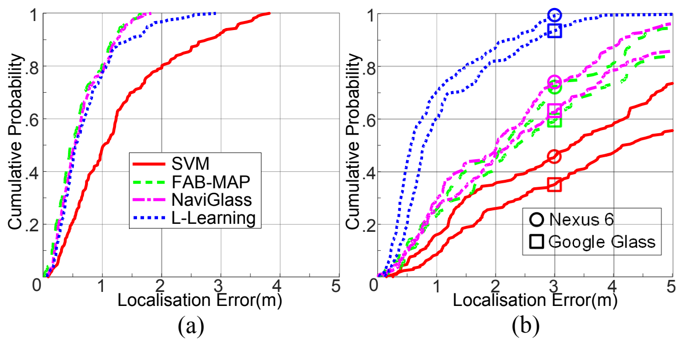


Fig. 14. (a) Error distribution of offline localisation. (b) Error distribution of online localisation on different devices.

computational power (see Table. 1), but as shown later in Sec. 5.2, the proposed approach is able to achieve significant performance gain on all of them. Our visual processing pipeline (for both front-end and back-end) is built with OpenCV 2.4.10, and uses SURF [12] to extract visual features.

Ground Truth: We use the Conditional Random Fields (CRFs) based map matching approach (in [6]) to generate ground truth. We assume the accurate metrical maps (i.e. floorplans) are available, and at certain points of the trajectories, the true positions of the users can be inferred from the captured images (e.g. at turns, or when passing by a unique landmark). Those known positions are manually labelled and fed into the CRFs model as priors, which help the map matching process converge to the correct estimates.

Competing Algorithms: We compare the proposed lifelong learning approach (referred to as **L-Learning** hereafter) with the following three competing algorithms: 1) **SVM**, which is our implementation of the existing Travi-Navi [3] system. It uses pedestrian dead-reckoning (PDR) to estimate the displacement of the user, and the Bag-of-Words (BOW) model to represent images. Given a trajectory, the images captured at nearby locations (e.g. within 3-step range) are clustered into groups to train a linear Support Vector Machine (SVM). During localisation, the observed images are matched to the saved ones based on the trained SVM. 2) **FAB-MAP**, which also uses PDR to compute the inertial trajectories, but considers the more advanced FAB-MAP model [9] for image matching. Comparing to the above SVM algorithm, it takes the important correlations between the visual words into account, and evaluates the similarity between images with a graphical model. However it does not incorporate any optimisation of the visual processing pipeline: it uses the whole vocabulary and full images at the same scale. 3) our previous work **NaviGlass** [8], which uses a similar processing pipeline as FAB-MAP, but with a globally reduced visual vocabulary. Note that comparing to the proposed L-Learning approach, it doesn't consider the optimal image scales/regions, nor the spacial variations in visual words: it just uses a smaller visual vocabulary throughout. To be fair, for all algorithms we use the same PDR implementation, SURF parameters, and state estimation algorithm as in Sec. 4.

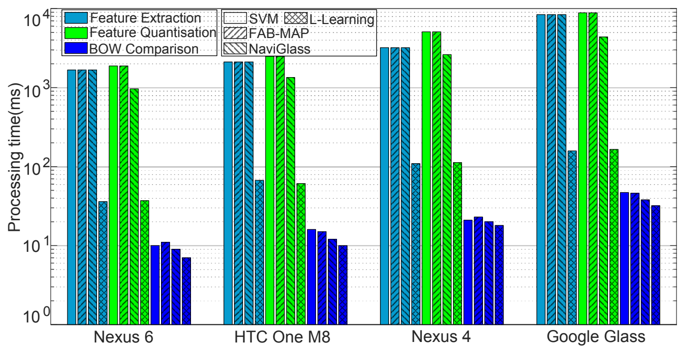


Fig. 15. The running time of feature detection and quantisation per image for different devices. The proposed approach is up to 50× faster than the competing algorithms.

5.2 Experiment Results

Localisation Accuracy: The first set of experiments evaluate the localisation accuracy of the proposed (L-Learning) and competing (SVM, FAB-MAP and NaviGlass) algorithms given their different visual processing techniques. We first consider the ideal offline scenarios, where the mobile devices are allowed to process all of the captured images beforehand, and report user positions later. Fig. 14(a) shows the distribution of localisation errors in offline. We can see that the naive SVM has much larger errors comparing to FAB-MAP, NaviGlass and the proposed L-Learning, and the gap between the latter three algorithms is very small. This means although L-Learning only processes a tiny portion of information comparing to FAB-MAP and NaviGlass, it is able to achieve nearly the same accuracy. On the other hand, in online localisation scenarios, the accuracy of SVM, FAB-MAP and NaviGlass drops significantly, as shown in Fig. 14(b). This is because the expensive visual processing pipeline severely limits the image rate, e.g. on Nexus 6 it takes about 4s to process one 800×600 image and Google glasses need almost 20s to finish (see Fig. 15). Thus those algorithms can't correct the fast growing drifts of PDR in time during online positioning. However, the proposed L-Learning algorithm does not suffer from such a problem since it is much more lightweight (<100ms on Nexus 6), and is able to localise in real-time with high accuracy (mean error 0.96m).

Visual Processing Cost: As discussed above, the cost of visual processing has enormous impact on the accuracy when localising online. The second experiment studies the visual processing time of the competing algorithms on different mobile devices. Fig. 15 (note the log scale) shows the break down of the average wall clock time of processing one image. Firstly, we see that the major computational bottleneck is feature extraction and quantisation, where the cost of BOW comparison is negligible for all algorithms. Secondly, on all devices the proposed L-Learning algorithm requires much less time in both feature extraction and quantisation than competing approaches (up to 50× faster). This is expected, since our algorithm learns to only work on a) the minimum necessary scale/region of the image, and b) the most discriminative key words in the visual vocabulary. Note that comparing to SVM and FAB-MAP, NaviGlass is able to reduce roughly half of the cost on feature quantisation since it only considers a subset of the visual words. However, comparing to the proposed L-Learning, it doesn't optimise the image scale/region, nor considers the minimum key words at different parts of the experience. Finally, the cost on different devices varies significantly, where the wearable smart glasses require much more processing time than the phones.

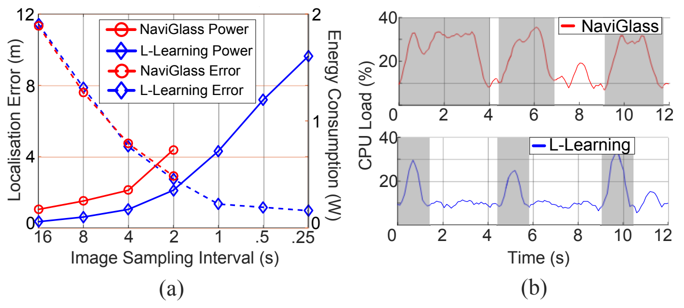


Fig. 16. (a) Localisation accuracy and energy consumption under different image sampling intervals. (b) Normalised CPU load of NaviGlass (top) and L-Learning (bottom) when sampling images every 4s.

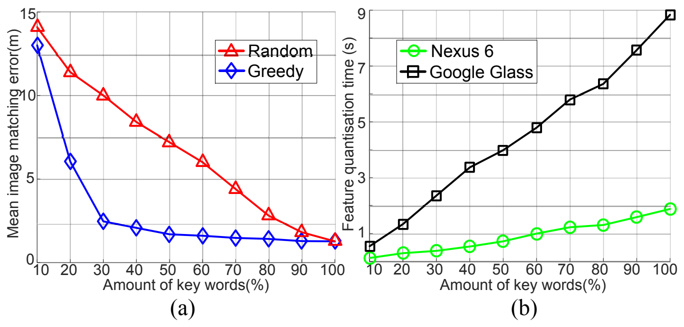


Fig. 17. (a) Mean image matching error, and (b) running time of feature quantisation per image when considering different amount of key words.

TABLE 2
Estimated battery life (hours) of running NaviGlass and L-Learning.

Sampling Interval (s)	16	8	4	2
NaviGlass	6.9	6.6	6.3	5.2
L-Learning	7.5	7.3	7.0	6.3

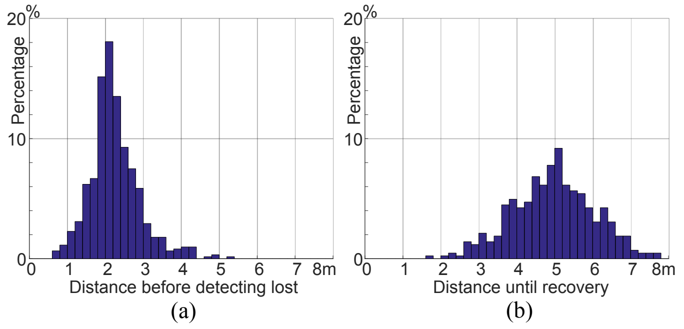


Fig. 18. Distance travelled between (a) the actual deviation point and when detecting localisation failure, and (b) the actual return point and when successful localisation is resumed.

Accuracy vs. Resource Consumption: The third set of experiments investigate the trade-off between localisation accuracy and resource consumption of the proposed system. Fig 16(a) shows the mean localisation error and the energy consumption of our system and the state-of-the-art NaviGlass when the image sampling interval varies from 16s to 0.25s. Note that here we only evaluate the systems on the Nexus 6 (with Qualcomm Trepn Profiler [15]), since on other devices NaviGlass takes too long to process images (see Fig. 15). As shown in Fig 16(a), NaviGlass is only able to process images every 2s, while the proposed L-Learning can process 4 images per second. In addition, for both approaches smaller image sampling intervals lead to lower localisation error, but also cause higher energy consumption. Table. 2 shows the estimated battery life of NaviGlass and the proposed L-Learning, which is evaluated by running the algorithms for one hour period, and then projecting the expected battery life based on the observed energy consumption. We repeat this procedure for five times and report the average. We see that for L-Learning, when capturing images at 1Hz, the positioning error has already dropped around 1m, while the gain in accuracy becomes marginal when the image sampling rate further increases. Finally, although the localisation error of NaviGlass is comparable to L-Learning, to process the same amount of images, L-Learning only consumes about half energy of NaviGlass. As a result, on Nexus 6 L-Learning can achieve up to 21% longer battery life than NaviGlass, as shown in Table. 2. This is because NaviGlass takes much longer time to process each image, where the CPU is constantly occupied, as shown in Fig. 16(b). Therefore, when energy is not an issue, only L-Learning has the option to sample denser images to improve accuracy: as in Fig. 16(a), comparing to the best performance produced by NaviGlass, L-Learning can further reduce the localisation error to about 1/3.

Impact of Key Word Discovery: This set of experiments evaluate the proposed key word discovery techniques. We keep images at the original scale without salient regions, but vary the size of the key word set, from 10% to the complete vocabulary. To exclude the impact of the inertial measurements, here we consider the image matching error, which is the mean distance between the locations of the matched images and the ground truth. Fig. 17(a) shows the image matching accuracy when using different amount of key words. We compare our greedy key word discovery algorithm with a baseline approach that randomly selects words. We can see that as more words are incorporated, the error of our approach drops much quicker than the baseline, and after 30$\sim$40% the improvement becomes marginal. This confirms that in most cases only a small amount of informative words are necessary to secure the correct matching. On the other hand, as shown in Fig. 17(b), the feature quantisation time increases linearly (note that the increasing rate of Google glasses is much steeper than that of Nexus 6). Therefore, the proposed key word discovery approach is able to reduce the cost on feature quantisation greatly without compromising accuracy, e.g. when using 30% keywords, the mean error of L-Learning is only 1.19m larger than that of using the complete vocabulary, but the running time is reduced to about 1/4.

Impact of Image Scales and Salient Regions: This set of experiments investigate the trade-off of using variable image scales, and performance gain of the proposed salient region detection approach under a fixed visual vocabulary. Firstly, using images with lower scales increases the matching error, as shown in Fig. 19(a). However, note that from the original image (800×600) at scale 1 to scale 0.3 (240×180), the average image matching error only increases by 0.67m, while from scale 0.3 down to 0.1, the error grows drastically. This indicates that we can process images at lower scales while still maintain reasonable accuracy. In addition, we see that the gap between only processing salient regions (red line with triangles) and the full images (blue line with diamonds) is tiny, i.e. the proposed salient region detection approach won't jeopardise the matching accuracy. Secondly, the relative sizes of the detected salient regions vary at different scales. As shown in Fig. 19(b), at lower scales the detected salient regions typically occupy large parts of the images (e.g. $\sim$80% of the image size at scale 0.2), while at higher scales the ratio becomes much

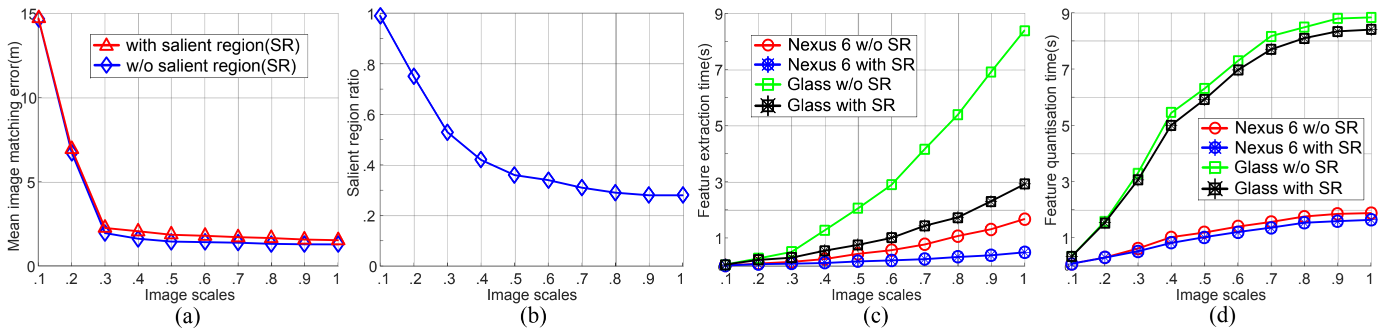


Fig. 19. (a) Mean image matching error at different image scales, with/without salient region detection. (b) Relative sizes of the detected salient regions (percentage comparing to the full image) at different scales. (c) Running time of feature detection, and (d) quantisation at different image scales.

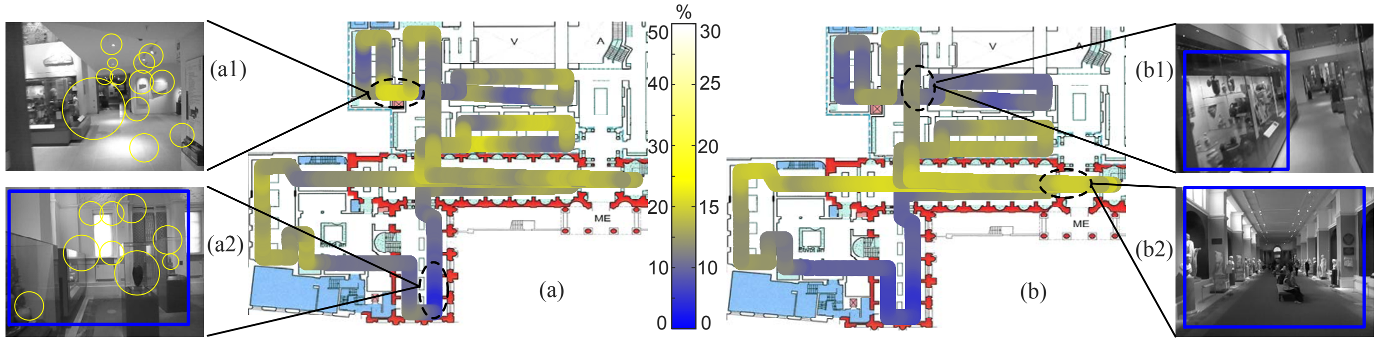


Fig. 20. (a) The optimal percentage of key words (relative to the complete vocabulary), and (b) pixels (relative to the original image size) learned by the proposed algorithm across the visual experiences at the museum site.

smaller (<30% at scale 1). This is because higher scale images typically contain more detail, and thus features extracted from smaller salient regions are sufficient to achieve correct matching. Thirdly, using variable image scales has effect on the running time of both feature extraction and quantisation. As shown in Fig. 19(c) and (d), the feature extraction time increases quadratically with respect to image scales, but the growth of quantisation time slows down at higher scales. This is also expected because under the BOW model, the quantisation cost is proportional to the number of *unique* words, where higher scale images tend have a lot of repetitions of the same visual elements. Finally, using salient regions won't be able to save much in feature quantisation (Fig. 19(d)), but could significantly reduce the feature extraction time, especially at high scales (Fig. 19(c)). This confirms that our salient region detection algorithm can effectively reduce the amount of pixels needed to be processed, while still keep most of the important visual elements appear in the images.

Spatial Variations: This experiment shows the spatial variations of the parameters learned by the proposed approach. Fig. 20(a) illustrates the sizes of optimal key word sets (percentage of the full vocabulary) across space, and Fig. 20(b) is the amount of pixels (percentage of all pixel in the original image) contained in the detected salient regions. Firstly, we see in most areas the learned parameters only contain a small portion of the original vocabulary or pixels, e.g. we only have to consider at most half of the whole vocabulary, while the average amount of pixels needed to be processed is roughly 10∼15% of the original image. However, we do observe clear spatial variations. For instance in image Fig. 20(a1), the scene is dominated by common visual elements, such as lights or door frames, and thus more words are required to distinguish it from the others within that area. On the contrary, image Fig. 20(a2) contains very unique features, e.g. statues and glass cabinets, so it is sufficient to achieve correct matching with just a few words. Similarly, Fig. 20(b1) and (b2) show two cases where different scales and salient regions are considered. As we can see, the experience segment containing image Fig. 20(b2) passes through large open space, where visual features are faraway and uniformly distributed. Therefore in that area we need higher scale images with wide salient regions to capture informative features. On the other hand, Fig. 20(b1) covers a narrow corridor where most features are clustered on the left, and thus lower scales with smaller salient regions are sufficient.

Localisation Failure Detection and Recovery: The last set of experiments illustrate our system's ability of detecting and recovering from localisation failure. In our experiments, we asked the participants to deliberately explore new routes (e.g. as shown in Fig. 12(b)), and consider those trajectories as the cases where the users get lost. In addition, we also synthesise deviations from the collected experiences. We first randomly select two nodes n_d and n_r of the current experience E^c, as the deviation and return point respectively. Then at the deviation point n_d, we create a turn with random heading change (drawn from a Gaussian distribution learned from the data), simulating the scenarios where the user deviates from following the previous experience. Then we replace the experience segment between n_d and n_r with a segment sampled from the experiences collected in another environment. In this way, we create a synthetic "new route" (starting at n_d and ending at n_r), and we use this modified E^c to evaluate how our system handles with localisation failure. Fig. 18(a) shows the distribution of the distance travelled between n_d, i.e. the real deviation point, and the point where our system reports localisation failure. As we can see that, the system is able to detect localisation failure quite quickly: in most cases it only needs 2∼3m (3∼5 steps). On the other hand, Fig. 18(b) is the distribution of gap between n_r, i.e. the actual point where the user returns, and the point where our system is able to resume localisation. We see that it generally takes longer to recover from localisation failure, since our decision model (as shonw in Fig. 12) requires a consecutive sequence of strong image matchings to reinstate successful localisation. However, we observe that in our experiments at most after 8m, the proposed system is able to re-localise the user.

6 Related Work

Teach-repeat Navigation: Teach-repeat navigation has been widely used in guiding robots in GPS-denied scenarios [16]. Recently, this teach-repeat idea has been applied to indoor navigation, and various sensing modalities have be considered such as radio, magnetic field and vision [3], [1], [2]. In this context, the teach phase is performed by a group of motivated users, such as the shop owners, who walk through the predefined routes (e.g. from the main entrance to their shops) and record the sensor measurements with their mobile phones. When a user being navigated through these routes, the live sensor observations are compared against previously saved measurements to track her current position, and navigation instructions are provided accordingly. To cope with the temporal and spatial variations of the sensor readings, e.g. the lighting changes, the experience-based navigation (EBN) [10] automatically switches to teach mode to save the new features unseen before. The proposed approach is based on EBN framework, in the sense that we also maintain variable visual experiences at different areas, but it is fundamentally different from the above teach-repeat systems. Our approach doesn't just allow the users to repeat the previously taught experiences, but also proactively learns how to follow those experiences more efficiently from the repetitions.

Vision-based Indoor Positioning: Vision-based indoor positioning techniques have attracted a lot of interests. One of the popular solutions is Structure from Motion (SfM) [17], [18], which uses images taken from different angles to reconstruct the 3D positions of feature points/lines. During localisation, the camera poses are computed by matching the live features against the constructed feature cloud with known positions. Typically SfM is expensive in computation since it has to compute the 3D model of the scene, which limits its application on resource constrained devices. The other class of approaches uses visual odometry (VO) techniques [19], which estimates the camera poses by evaluating the metrical transformation between adjacent image frames. With monocular cameras (equipped on most mobile devices), VO approaches can only estimate the user positions up to a scale since no depth information is available. The recent vision-aided inertial tracking [20] fuses inertial measurements with visual odometry, and works well on commodity mobile phones. However, VO based approach rely heavily on good initialisation, and it is very difficult to re-converge after any localisation failure, e.g. abrupt device orientation changes. Moreover, both SfM and VO based approaches require high image sampling rate (∼30Hz). On the other hand, visual matching based approaches such as Travi-Navi [3] uses sparser images, but unlike the proposed approach, the matched images are mainly used for pathway identification rather than localisation. In addition, the proposed approach aims to make real-time visual positioning practical and efficient for resource constrained devices such as wearables, which is different from most of the existing techniques.

Managing Computation on Resource Constrained Devices: There has been a large body of work on reducing computation on mobile phones and wearables devices. One common solution is to offload the heavy computation to the cloud [7], [21]. For real-time systems, the main challenge is to achieve good trade-off between communication delay and computational cost. For instance, [7] considers the cloudlet architecture, which uses local servers to achieve low-latency interactions with wearable devices. The work in [21] consider the visual place recognition problem on mobile phones, and reduces the amount of data to be transmitted by only offloading the most informative features. Unlike those systems, the proposed approach doesn't require constant offloading during operation, and thus has much less communication overhead. The other line of research tries to reduce the dimension of data to be processed. For example, [22] uses random matrix to approximate images for face recognition on mobile phones, while [23] uses compressive sensing techniques to reduce data volumes in mobile crowdsourcing. The proposed system shares similar ideas, but is also very different: rather than one time optimisation, it keeps the optimisation results (learned parameters) for future use, and continues to improve as more experiences are accumulated.

Learning Place-dependent Scene Signatures: The proposed approach is also closely related to the research on understanding unique features at different places. For instance, [24] shows that images captured at one city can be effectively distinguished from those captured at another by learning place-specific SVM classifiers on image patches. The work in [25] and [26] extends this idea to localise robots under extreme scene changes. At the training phase, for images captured from a known location, it learns the image regions that are robust to lighting and weather changes. During localisation, image patches from those regions are extracted and used as bespoke feature detectors to estimate the camera motion. Our approach also learns place-dependent scene properties, but is orthogonal to such work in that a) the learned place-dependent parameters is used to reduce the visual processing cost, but not for pose estimation; and b) our system learns not only the informative image regions, but also the optimal image scales and visual vocabulary that are essential to localisation success.

7 Conclusion and Future work

This paper proposes a novel lifelong learning approach, which makes indoor positioning with visual experiences efficient and practical. We show that by continuously learning from following the previously accumulated experiences, we are able to construct a series of visual processing parameters, which encode the optimal settings at different parts of the experiences. In future localisation, our positioning system actively tunes its visual processing pipeline according to the learned parameters, to achieve accurate and real-time localisation on resource-constrained platforms. We implement the proposed approach on an array of mobile and wearable devices, and extensive experiments have demonstrated that: a) with the learned parameters, the cost of visual processing can be reduced up to two orders of magnitude without jeopardising the localisation accuracy; b) in most cases, a small set of most distinctive key visual words are sufficient to guarantee the correct image matchings, which could save most of the feature quantisation cost; c) using images at suitable scales reduces cost on both feature extraction and quantisation significantly, while only processing the salient image regions could further cut the feature extraction cost, especially at high scales; d) the learned parameters capture the spatial variations of indoor environment, where different key word sets, image scales and salient regions are considered to seek the best trade-off between cost and accuracy. For future work, we plan to combine the proposed approach with global localisation techniques such as SLAM, incorporate more types of sensing modalities, and shift from the hand-crafted features to learned features, e.g. using deep neural networks.

REFERENCES

[1] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did you see bob?: Human localization using mobile phones," in *Proc. MobiCom*, 2010.

[2] Y. Shu, K. G. Shin, T. He, and J. Chen, "Last-mile navigation using smartphones," in *Proc. MobiCom*, 2015.

[3] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-navi: Self-deployable indoor navigation system," in *Proc. MobiCom*, 2014.

[4] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal, "Efficient, generalized indoor wifi graphslam," in *Proc. ICRA*, 2011.

[5] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," in *Proc. MobiCom*, 2012.

[6] Z. Xiao, H. Wen, A. Markham, and N. Trigoni, "Lightweight map matching for indoor localisation using conditional random fields," in *Proc. IPSN*, 2014.

[7] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. MobiSys*, 2014.

[8] Y. Zhang, W. Hu, W. Xu, H. Wen, and C. T. Chou, "Naviglass: Indoor localisation using smart glasses," in *Proc. EWSN*, 2016.

[9] 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.

[10] 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.

[11] Z. Xiao, H. Wen, A. Markham, and N. Trigoni, "Robust pedestrian dead reckoning (r-pdr) for arbitrary mobile device placement," in *Proc. IPIN*, 2014.

[12] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proc. ECCV*, 2006.

[13] H. Wen, Y. Shen, S. Papaioannou, W. Churchill, N. Trigoni, and P. Newman, "Opportunistic radio assisted navigation for autonomous ground vehicles," in *Proc. DCOSS*, 2015.

[14] 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.

[15] "Qualcomm trepn power profiler," https://developer.qualcomm.com/software/trepn-power-profiler, accessed: 2016-04-06.

[16] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[17] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof, "From structure-from-motion point clouds to fast location recognition," in *Proc. CVPR*, 2009.

[18] B. Micusik and H. Wildenauer, "Descriptor free visual indoor localization with line segments," in *Proc. CVPR*, June 2015.

[19] S. Hilsenbeck, A. Moller, R. Huitl, G. Schroth, M. Kranz, and E. Steinbach, "Scale-preserving long-term visual odometry for indoor navigation," in *Proc. IPIN*, Nov 2012.

[20] M. Li and A. I. Mourikis, "Vision-aided inertial navigation with rolling-shutter cameras," *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1490–1507, 2014.

[21] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, and E. Steinbach, "Mobile visual location recognition," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 77–89, 2011.

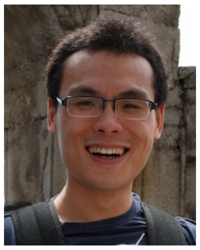
[22] Y. Shen, W. Hu, M. Yang, B. Wei, S. Lucey, and C. T. Chou, "Face recognition on smartphones via optimised sparse representation classification," in *Proc. IPSN*, 2014.

[23] L. Xu, X. Hao, N. D. Lane, X. Liu, and T. Moscibroda, "More with less: lowering user burden in mobile crowdsourcing through compressive sensing," in *Proc. UbiComp*, 2015.

[24] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros, "What makes paris look like paris?" *ACM Transactions on Graphics*, vol. 31, no. 4, 2012.

[25] C. McManus, B. Upcroft, and P. Newmann, "Scene signatures : localised and point-less features for localisation," in *Robotics: Science and Systems X*, University of California, Berkeley, CA, July 2014.

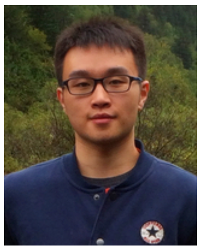
[26] C. Linegar and P. Newman, "Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera," in *Proc. ICRA*, 2016.



Dr. Hongkai Wen is an Assistant Professor in the Department of Computer Science, University of Warwick. He obtained his D.Phil at the University of Oxford, and worked as a post-doctoral researcher in Oxford Computer Science and Robotics Institute. His research interests are in mobile sensor systems, human-centric sensing, and pervasive data science.

Dr. Ronald Clark is a research fellow at the Dyson Robotics Lab, Imperial College London. He obtained his PhD from the Department of Computer Science, University of Oxford. He is interested in the general topic of visual machine perception which is needed to enable mobile devices to model, explore and understand their surroundings.

Dr. Sen Wang is an Assistant Professor in Robotics and Autonomous Systems at Heriot-Watt University and a faculty member of the Edinburgh Centre for Robotics. Previously, he was a post-doctoral researcher at the University of Oxford. His research focuses on robot perception and autonomy using probabilistic and learning approaches, especially autonomous navigation, robotic vision, SLAM and robot learning.



Xiaoxuan Lu is currently a third-year PhD student in Department of Computer Science, University of Oxford. Before that, he obtained his MEng degree at Nanyang Technology University, Singapore. His research interest lies in Cyber Physical Systems, which use networked smart devices to sense and interact with the physical world.

Bowen Du received the B.E. and M.E. degrees in Software Engineering from Tongji University, Shanghai, China in 2013 and 2016 respectively. He is currently pursuing his Ph.D. in computer science at the University of Warwick, Coventry, U.K. His research interests focus on Cyber Physical Systems, mobile computing and artificial intelligence in sensor systems.

Dr. Wen Hu is a senior lecturer at School of Computer Science and Engineering, the University of New South Wales (UNSW). Much of his research career has focused on the novel applications, low-power communications, security and compressive sensing in sensor network systems and Internet of Things (IoT). He is a senior member of the IEEE.

Prof. Niki Trigoni is a Professor at the Department of Computer Science, University of Oxford. She is currently the director of the EPSRC Centre for Doctoral Training on Autonomous Intelligent Machines and Systems, and leads the Cyber Physical Systems Group. Her research interests lie in intelligent and autonomous sensor systems with applications in positioning, healthcare, environmental monitoring and smart cities.