Boosting Descriptors Condensed from Video Sequences for Place Recognition

Tat-Jun Chin, Hanlin Goh and Joo-Hwee Lim
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613.
{tjchin, hlgoh, joohwee}@i2r.a-star.edu.sg

Abstract

We investigate the task of efficiently training classifiers to build a robust place recognition system. We advocate an approach which involves densely capturing the facades of buildings and landmarks with video recordings to greedily accumulate as much visual information as possible. Our contributions include (1) a preprocessing step to effectively exploit the temporal continuity intrinsic in the video sequences to dramatically increase training efficiency, (2) training sparse classifiers discriminatively with the resulting data using the AdaBoost principle for place recognition, and (3) methods to speed up recognition using scaled kd-trees and to perform geometric validation on the results.

Compared to straightforwardly applying scene recognition methods, our method not only allows a much faster training phase, the resulting classifiers are also more accurate. The sparsity of the classifiers also ensures good potential for recognition at high frame rates. We show extensive experimental results to validate our claims.

1. Introduction

Our objective is *place recognition* i.e. given an input image we wish to determine the identity of places by *recognizing their facades*. By "places" we mean landmarks or buildings with *prominent facades*. Place recognition is instrumental in Mobile Augmented Reality (MAR) systems [5], where proponents envision an application that allows users to point their camera phone (or other mobile devices with a camera) at a place to access more information about it.

An image-based system can recognize a place in an image only if the given facade was observed previously under roughly the same conditions (e.g. viewpoint, lighting). Consequently data collection becomes complicated, especially for large buildings or landmarks: Capturing from afar to fit a building in an image causes important visual features to disappear, so naturally we prefer close range images. However, close distances allow a large number of distinct viewing positions and viewpoints, and hence, distinct (sub)facades of the building[1]. The data collector is presented with the dilemma of choosing among them, especially if each facade has the equal probability of being presented as a query image.

To alleviate this problem we propose to capture facades with video recordings. Instead of snapping just a few images, the collector pans smoothly to capture a place in video e.g. in a pure planar motion. For a given viewing position, a set of video recordings can acquire much more information about a place than a set of images. Fig. 1 illustrates the idea.

(a) Coarsely sampled viewpoints of a viewing position.

(b) Video frames captured *from the same position* include all viewpoints.

Figure 1. Collecting images of facades of a large building.

From a cursory glance distinguishing between still images and video seems vacuous— after all one can always break up a video sequence into still images for individual treatment. However we maintain such a distinction for two reasons: (1) Panning in video dramatically simplifies data collection, since given a viewing position the data collector

[1] In this work, "viewing position" differs from "viewpoint. The former implies an x-y coordinate on the surface of the earth, while the latter refers to the direction towards which to observe a place given a viewing position.

978-1-4244-2340-8/08/$25.00 ©2008 IEEE

does not need to choose viewpoints[2]. One should also be able to appreciate not needing to manually snap images continuously during collection. (2) More crucially, the deluge of visual information from video sequences presents the significant challenge of effectively processing them. Straightforwardly breaking up the video sequences into separate frames and applying scene or object category recognition techniques [2, 7, 9, 10], which traditionally dealt with individual still images, will not be efficient. However, since temporal continuity (which cannot be assumed to exist in a raw collection of images) is intrinsically preserved in video sequences, it can be exploited for efficient processing.

1.1. Contributions

We propose a solution, described by the following steps, to process a video sequence database captured in the manner of Fig. 1(b) to construct a robust place recognition system:

1. A method to *filter* and *condense* the visual information in the video sequences into a more *compact* form (see §2) by exploiting temporal continuity.

2. Train sparse and discriminative classifiers *efficiently* from the results of Step 1 using the AdaBoost algorithm (see §3). Using the results of Step 1 instead of treating video frames individually allows us to avoid being overwhelmed with the vast amount of raw visual information from the input video sequences (see §3.3).

3. Speeding up recognition with fast nearest neighbour searches using a modified kd-tree structure (see §4.1), and performing geometric validation on the recognition results to increase recognition accuracy (see §4.2).

Note that in our system, during the testing phase queries are accepted and processed in the form of still images.

1.2. Related work

Significant progress has been made in the area of *scene* or *object category* recognition [2, 13, 14, 7, 9, 10]. Though we emphasize that *place recognition* is slightly different since we aim to recognize *specific* places rather than scene *categories*, ideas from [2, 13, 14, 7, 9, 10] can certainly be applied. Generally, most of the methods involve building visual vocabularies from keypoint descriptors and learning generative models from quantized descriptors. However, for MAR on mobile devices we cannot afford to load large data structures like visual vocabularies, quantize descriptors and then perform recognition. Hence we favour an approach without visual vocabularies and quantization, and which is also discriminative such as [10] which promises faster recognition than generative methods.

[2]One can also regard this as a view sampling issue, whereby our video capture methodology advocates a very dense sampling.

A work similar in objective is [3], where the "Informative Features Approach" is used to select informative SIFT features [8], via local entropy estimations, that describes a set of places. The selected features are used for recognition in a MAP decision making process. Our work differs by using a very different idea for feature selection, i.e. the AdaBoost algorithm which directly selects features based on their classification potential. We believe our method is simpler and results in sparser and faster classifiers. Moreover, we also tackle the issue of building a practical large scale MAR system (see §5), whereas [3] tested on small datasets only (e.g. the ZuBuD [11] with only 5 images per building).

Our video processing framework is similar in spirit with [13, 14], where the basic idea is to summarize the local features which exist throughout a video sequence. However the final objectives are starkly different: In [13, 14], the goal is to build an index for the visual features in a movie to facilitate the searching of objects of interest i.e. the query is a subimage of a movie frame containing an object. Our aim is to train classifiers from the condensed visual features, and the query is an unseen before image. Moreover the different nature of the training video sequences (feature-length films in [13, 14] versus video recordings of places) warrants different video processing methods (more details in §2).

A previous research towards densely capturing facades for place recognition is [6], where "route panoramas" are obtained by line scanning a scene with a camera mounted on a vehicle as it traverses a street in a city. Given a query image, the camera position is recovered (hence, the place is recognized) by finding it's epipole in the route panorama. This can be costly since a RANSAC-like procedure is required for each query [6]. In contrast our method only requires matching a small set of local features (see §3 and §4).

In [15] video sequences are used as inputs for *querying* in place recognition, where video motion coders from mobile phones are exploited to aid in tracking local features across the video frames. The aim is to quickly identify seen-before and newly emerged keypoints so as to speed-up feature extraction in the *querying* phase. Our method also involves tracking keypoints, but our focus is on the *training* phase, i.e. to train a place recognition system using video sequences as samples, and hence is complementary to [15].

2. Processing Videos of Places

A video database where each video is labeled according to place identity is first collected. The videos are captured in the manner of Fig. 1(b), and a place can have several videos depending on its physical size. Given a video sequence, based on the SIFT [8] framework we detect scale invariant keypoints in *every* frame and assign descriptors to them.

Even in one sequence, in total a massive number of keypoints are obtained, and we aim to reduce the number of keypoints to consider for classifier training. Since our

videos are recorded in a smooth panning motion, many of the keypoints in a frame will be re-occurances from the previous frames (but in slightly differing views). We can track keypoints across the sequence to identify the overlaps.

2.1. Finding keypoint overlaps

Let $\{(\mathbf{x}_i, \mathbf{p}_i)\}$ and $\{(\mathbf{y}_j, \mathbf{q}_j)\}$, $1 \leq i \leq m$ and $1 \leq j \leq n$, be the sets of keypoints detected in two successive frames, with $\mathbf{x}_i$ and $\mathbf{y}_j$ denoting the keypoint positions and $\mathbf{p}_i$ and $\mathbf{q}_j$ their descriptors. Since the images represent two views of the same scene, a homography $\mathbf{H}$ exists between corresponding points:

$$\mathbf{H}\tilde{\mathbf{y}} \times \tilde{\mathbf{x}} = 0 \ ,$$

where $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{x}}$ indicate the homogenous coordinates of $\mathbf{y}$ and $\mathbf{x}$. Our aim is to find the best homography $\mathbf{H}^*$.

To achieve this, we first compute a pairwise similarity matrix using the Euclidean distance between $\mathbf{p}_i$ and $\mathbf{q}_j$. All possible corresponding keypoints between the two frames are identified by considering that a pair of keypoints are matching if the distance of their SIFT descriptors are below a pre-defined threshold. $\mathbf{H}^*$ is determined as the $\mathbf{H}$ that allows the most number of corresponding keypoints to overlap (i.e. the distance between $\tilde{\mathbf{x}}$ and $\mathbf{H}\tilde{\mathbf{y}}$ is below a certain threshold). We perform a RANSAC procedure to estimate $\mathbf{H}^*$. For more details, refer to [4].

The process is repeated successively on each frame pair, and overlapping keypoints are accumulated into the same track (keypoints without matches are simply discarded). Fig. 2 illustrates the idea, and Fig. 3 shows an example result. As an indication of effectiveness, a typical 25-frame video sequence in our database contains a total of about 30,000 SIFT keypoints. Among these only about 4,000 are determined as unique by the method. In fact this can also be considered a *filtering* process, where only keypoints consistently detectable in multiple views are kept.

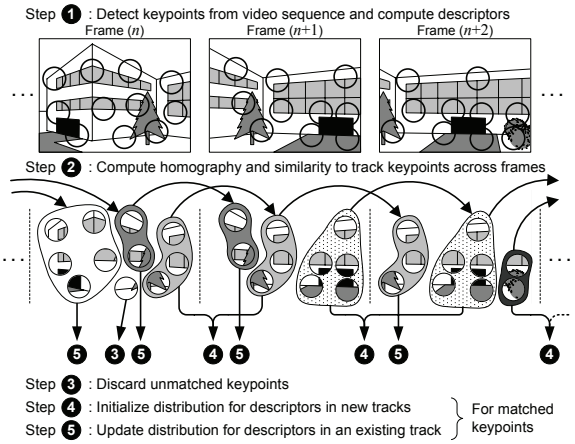


Figure 2. Finding keypoint overlaps using temporal continuity.

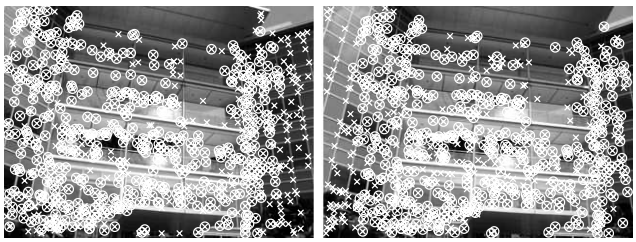


Figure 3. Overlapping keypoints are re-occurances of the same local feature. In this pair, crosses are detected keypoints, and those with bounding circles indicate that an overlap is found.

Our aim is similar to [13], where each keypoint is tracked individually since there could be multiple moving objects of interest in their video. Our idea is more suited here, since we wish to separate the static background (the facade of interest) from dynamic occlusions (the keypoints of which are discarded for not obeying the global homography).

2.2. Estimating descriptor distributions

We derive a parsimonious representation for the descriptors in a particular track by representing them with a Gaussian distribution. A distribution is more expressive than a simple average, as was done in [13]. We use a diagonal instead of a full covariance since a compact representation is desired. The distributions are updated incrementally so we do not have to maintain a large number of descriptors. For a particular track, at time t let $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ be the mean and covariance of its distribution of m descriptors. At time $t+1$, if a new descriptor $\mathbf{p}_{t+1}$ is added, $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ are updated as

$$\begin{aligned} \boldsymbol{\mu}_{t+1} &= \tfrac{m}{m+1}\boldsymbol{\mu}_t + \tfrac{1}{m+1}\mathbf{p}_{t+1}, \\ \boldsymbol{\Sigma}_{t+1} &= \tfrac{m}{m+1}\boldsymbol{\Sigma}_t + \tfrac{m}{(m+1)^2}(\boldsymbol{\mu}_t - \mathbf{p}_{t+1})(\boldsymbol{\mu}_t - \mathbf{p}_{t+1})^T. \end{aligned}$$

The off-diagonal elements of $\boldsymbol{\Sigma}_{t+1}$ are then zeroed. The first two descriptors of the track are used to initialize the mean and covariance matrix. For d-dimensional descriptors, at any point in time only $2d$ unique values, corresponding to the nonzero elements of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, are kept for each track (as opposed to $d(d+3)/2$ for the full covariance case).

2.3. Preserving geometric relations

The visual features on a facade do not exist independently of each other, and must appear on a fixed configuration. In this work, we make the assumption that the accumulated keypoints from a video sequence lie on a common 2D surface (the facade). Given two frames with a set of overlapping keypoints, the translation between the two frames is approximated by computing the average difference between the keypoint coordinates. By keeping track of the successive translations in a sequence, we can maintain a rough configuration of detected keypoints relative to the first frame. This information can be used later to perform geometric validation on recognition results (see §4.2).

3. Boosting Descriptor Distributions

After applying the steps outlined in §2, each video sequence in the database is reduced to a set of descriptor distributions. Each set also inherits the class (place) label of its video sequence. We train discriminative classifiers for place recognition by *boosting the descriptor distributions*. Via the AdaBoost algorithm, we select a small subset of descriptor distributions which are unique and informative. The resulting classifiers are sparse and can be evaluated quickly.

3.1. Boosting local features for place recognition

The AdaBoost algorithm for generic object recognition was introduced in [10]. Deriving from [10] for place recognition, AdaBoost aims to train classifiers of the form

$$H^c(\mathbf{I}) = \textstyle\sum_{t=1}^{T} \alpha_t^c \, h_t^c(\mathbf{I}) \ , \qquad (1)$$

where $H^c(\mathbf{I})$ gives the confidence of input image $\mathbf{I}$ containing the c-th class. $H^c(\mathbf{I})$ is obtained by boosting a series of weak classifiers $h_t^c(\mathbf{I})$, $t = 1, \ldots, T$, each with weight α_t^c, to become a strong classifier. A weak classifier is defined as

$$h_t^c(\mathbf{I}) = \begin{cases} 1 \text{ if } \min d(\mathbf{e}_t^c, \mathbf{v}_j) \leq \theta_t^c, \, \forall \, j = 1, \ldots, J \\ 0 \text{ otherwise} \end{cases} \qquad (2)$$

where $\mathbf{e}_t^c$ is the *defining local feature* of h_t^c, and $\mathbf{v}_j$ is one of the J local features of $\mathbf{I}$. Function $d(\cdot, \cdot)$ is a *dissimilarity measure* used to compare $\mathbf{e}_t^c$ and $\mathbf{v}_j$ with threshold θ_t^c, and the exact form of $d(\cdot, \cdot)$ is dependent on the forms of $\mathbf{e}_t^c$ and $\mathbf{v}_j$. More intuitively, $h_t^c(\mathbf{I})$ checks whether a feature sufficiently close to $\mathbf{e}_t^c$ exists in $\mathbf{I}$. Given a pre-determined T by the user, the AdaBoost algorithm finds the optimal $\mathbf{e}_t^c$, θ_t^c and α_t^c successively for $t = 1, \ldots, T$.

3.2. AdaBoost with descriptor distributions

Before invoking AdaBoost to obtain the h_t^c's, a *minimum dissimilarity matrix* $\mathbf{K}$ must be computed for the c-th class. In the current context, where our local features are the descriptor distributions, each entry $\mathbf{K}_{mn}$ contains the value

$$\mathbf{K}_{mn} = D_{KL}(\mathbf{f}_n, \mathbf{e}_m) \ . \qquad (3)$$

D_{KL} signifies the Kullback-Leibler Divergence (KLD), while $\mathbf{e}_m$ is the m-th descriptor distribution from the c-th place. Put more crudely, we lump all descriptor distributions from the c-th class together, and $\mathbf{e}_m$ is one of them.

The term $\mathbf{f}_n$ indicates the nearest neighbour of $\mathbf{e}_m$ in the n-th video sequence (both c and $\bar{c}$) in the following sense:

$$\mathbf{f}_n = \text{argmin}_{\mathbf{f}_i} D_{KL}(\mathbf{f}_i, \mathbf{e}_m) \ , \ \mathbf{f}_i \in \mathcal{F}^n \ . \qquad (4)$$

$\mathcal{F}^n$ is the set of descriptor distributions from the n-th video sequence. For Gaussians, the KLD has the closed form

$$\begin{aligned} D_{KL}(\mathbf{f}_i, \mathbf{e}_m) = 0.5[\, \log |\boldsymbol{\Sigma}_m| |\boldsymbol{\Sigma}_i|^{-1} + \text{tr}(\boldsymbol{\Sigma}_m^{-1} \boldsymbol{\Sigma}_i) \\ + (\boldsymbol{\mu}_m - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_m^{-1} (\boldsymbol{\mu}_m - \boldsymbol{\mu}_i) - d \,], \end{aligned} \qquad (5)$$

where $\{\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}$ and $\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ respectively characterize the distribution of $\mathbf{e}_m$ and $\mathbf{f}_i$. Given $\mathbf{K}$, we can apply the AdaBoost algorithm [10] to choose among the $\mathbf{e}_m$'s to form the set of T discriminative descriptor distributions $\mathbf{e}_t^c$. The details of the algorithm is beyond the scope of this paper, and the interested reader is referred to [10].

In our application, place recognition is eventually performed on still images. Consequently we use the Mahalanobis distance for $d(\cdot, \cdot)$ in Eq. (2):

$$d(\mathbf{e}_t^c, \mathbf{v}_j) = (\mathbf{v}_j - \boldsymbol{\mu}_t^c)^T (\boldsymbol{\Sigma}_t^c)^{-1} (\mathbf{v}_j - \boldsymbol{\mu}_t^c) \ , \qquad (6)$$

where $\{\boldsymbol{\mu}_t^c, \boldsymbol{\Sigma}_t^c\}$ define the distribution of $\mathbf{e}_t^c$, and $\mathbf{v}_j$ is one of the J keypoint descriptors (vectors) in the query image. The "Weak Hypothesis Finder" routine [10] is straightforwardly modified to retain only the third term of Eq. (5) to compute the threshold θ_t^c.

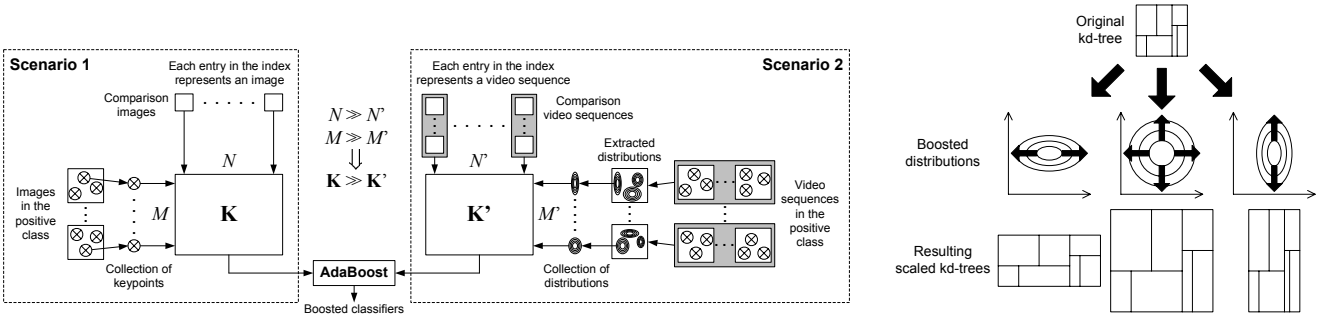
How we compute $\mathbf{K}$ and define the weak classifiers constitute the major differences between our work and [10], where $\mathbf{e}_m$ and $\mathbf{f}_n$ are simply descriptors (vectors), and the Euclidean distance is used as a dissimilarity measure, with $\mathbf{K}_{mn} = \|\mathbf{e}_m - \mathbf{f}_n\|$, and $d(\mathbf{e}_t^c, \mathbf{v}_j) = \|\mathbf{e}_t^c - \mathbf{v}_j\|$.

3.3. Why exploiting temporal continuity is useful

Given a completed $\mathbf{K} \in \mathbb{R}^{M \times N}$, finding a single $\mathbf{v}_t^c$ with the Weak Hypothesis Finder routine involves a complexity of $\mathcal{O}(N.F)$, where F is the average number descriptor distributions in a video sequence. The main computational burden is in fact the computation of $\mathbf{K}$ [10], where each entry requires a nearest neighbour search.

The benefits of processing videos, as opposed to individual treatment of each frames, is obvious by observing the size of $\mathbf{K}$. M is the total number of features from the video sequences of the positive class in consideration (i.e. the c-th class). By straightforwardly applying [10] on the video frames, since each detected keypoint (in fact, descriptor vector) counts as a feature, M can be a massive number: For our database (see §5), M can reach 100,000! In constrast, by identifying keypoint overlaps (as in §2), M can be reduced to a much more manageable value of about 12,000.

Of equal importance is the value of N, which is the total number of "samples" of places in the database. Using our framework, each video sequence is a sample (hence $N < 200$ for our database), whereas by directly applying [10], every frame in the video sequences is a sample, and N can reach up to 5000! Hence, by drastically lowering M and N we can reduce the effort required to construct $\mathbf{K}$. Since it is also more feasible to wholly load a small $\mathbf{K}$ into main memory, I/O overhead arising from fetching elements of $\mathbf{K}$ is non-existent. These factors contribute towards a much faster training phase of the AdaBoost algorithm (more details in §5). In fact, without applying our methods in §2 and §3, training AdaBoost classifiers on a video database of



(a) (b)

Figure 4. (a) Scenario 1 depicts the size of matrix $\mathbf{K}$ when applying the AdaBoost algorithm [10] by treating frames extracted from video sequences as individual images. Scenario 2 illustrates what happens to matrix $\mathbf{K}$ when we apply the proposed video processing method. (b) How kd-trees are scaled with Gaussian distributions of diagonal covariances.

modest size (≈10 video sequences) is barely feasible. Refer to Fig. 4(a) for an illustration of the idea.

3.4. Fast nearest distribution search with PDS

Fast nearest neighbour searches can further speed up the computation of $\mathbf{K}$. Unfortunately since the KLD is not a proper metric, common methods like kd-tree structures cannot be applied. We propose a novel fast nearest neighbour method for diagonal Gaussian distributions with the KLD.

Trivially, since 0.5 and d in Eq. (5) are common terms, evaluating Eq. (4) is equivalent to evaluating

$$\mathbf{f}_n = \text{argmin}_{\mathbf{f}_i} \tilde{D}(\mathbf{f}_i, \mathbf{e}_m) \ , \ \mathbf{f}_i \in \mathcal{F}^n \ , \tag{7}$$

where $\tilde{D}(\mathbf{f}_i, \mathbf{e}_m) = D_1(\mathbf{f}_i, \mathbf{e}_m) + D_2(\mathbf{f}_i, \mathbf{e}_m)$, and

$$D_1(\mathbf{f}_i, \mathbf{e}_m) := \log|\mathbf{\Sigma}_m||\mathbf{\Sigma}_i|^{-1},$$

$$D_2(\mathbf{f}_i, \mathbf{e}_m) := \text{tr}(\mathbf{\Sigma}_m^{-1}\mathbf{\Sigma}_i) + (\boldsymbol{\mu}_m - \boldsymbol{\mu}_i)^T\mathbf{\Sigma}_m^{-1}(\boldsymbol{\mu}_m - \boldsymbol{\mu}_i).$$

Let the covariance matrices be decomposed as such:

$$\mathbf{\Sigma}_i = \mathbf{L}_i\mathbf{L}_i^T \ \text{ and } \ \mathbf{\Sigma}_m^{-1} = \mathbf{L}_m\mathbf{L}_m^T \ . \tag{8}$$

Since $\mathbf{\Sigma}_i$ and $\mathbf{\Sigma}_m$ are diagonal, $\mathbf{L}_i$ and $\mathbf{L}_m$ are diagonal as well. We can compute D_1 and D_2 as the following:

$$D_1(\mathbf{f}_i, \mathbf{e}_m) \ := \ \log|\mathbf{\Sigma}_m| - \log|\mathbf{\Sigma}_i| \ , \tag{9}$$

$$D_2(\mathbf{f}_i, \mathbf{e}_m) \ := \ \|\mathbf{M}(\mathbf{p} - \mathbf{q})\|^2 \ , \tag{10}$$

where $\mathbf{p} := [\ \boldsymbol{\mu}_m\ \ \mathbf{0}\ ]^T$, $\mathbf{q} := [\ \boldsymbol{\mu}_i\ \ \text{diag}(\mathbf{L}_i)\ ]^T$ and

$$\mathbf{M} := \begin{bmatrix} \mathbf{L}_m^T & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_m^T \end{bmatrix} \ . \tag{11}$$

Finally, by observing that

$$D_2(\mathbf{f}_i, \mathbf{e}_m) = \sum_{j=1}^{2d} \mathbf{M}_{jj}^2(\mathbf{p}_j - \mathbf{q}_j)^2 \ , \tag{12}$$

we can implement a Partial Distance Search (PDS) [1] to find the solution of Eq. (4) in a shorter duration than an exhaustive nearest neighbour search. Note that since $\mathbf{\Sigma}_m$ is positive definite, $\mathbf{M}_{jj} > 0$ for all j.

4. Performing Place Recognition

Given a query image $\mathbf{I}$, a set of classifiers $H^c(\mathbf{I})$ with $c = 1, \cdots, C$ are evaluated, where C is the total number of distinct classes. The image $\mathbf{I}$ is assigned the class label of which the $H^c(\mathbf{I})$ response is highest, subject to it surmounting a pre-defined threshold. This section explores how to speed up recognition (§4.1), and how to perform geometric validation on the recognition results (§4.2).

4.1. Faster recognition with scaled kd-trees

Evaluating $H^c(\mathbf{I})$ involves performing T nearest neighbour operations, one for each weak classifier $h_t^c(\mathbf{I})$, as defined in Eq. (2). Effort towards increasing recognition speed should begin with speeding up nearest neighbour searches. We propose a method adapted from [12]. Basically for each H^c we have two sets of features, descriptor distributions $\mathbf{e}_t^c = \{\boldsymbol{\mu}_t^c, \mathbf{\Sigma}_t^c\}$ from H^c with $t = 1, \ldots, T$, and descriptor vectors $\mathbf{v}_j$ from the query image with $j = 1, \ldots, J$. Solving Eq. (2) is based on the Mahalanobis distance defined in Eq. (6). To this end we build a kd-tree structure on the *query image* descriptors $\mathbf{v}_j$. Each $\mathbf{e}_t^c$ is then used to "query" the structure to find $\min d(\mathbf{e}_t^c, \mathbf{v}_j)$. Since $\mathbf{e}_t^c$ is in fact a Gaussian distribution, we can re-scale the original Euclidean space in which the kd-tree resides according to $\mathbf{\Sigma}_t^c$ followed by querying the tree with $\boldsymbol{\mu}_t^c$. This almost always results in a faster operation than an exhaustive search. Refer to Fig. 4(b) for an illustration of the idea.

More formally, since we can decompose $(\mathbf{\Sigma}_t^c)^{-1} = \mathbf{L}_t^c(\mathbf{L}_t^c)^T$, we can compute Eq. (6) as

$$d(\mathbf{e}_t^c, \mathbf{v}_j) = \|(\mathbf{L}_t^c)^T(\mathbf{v}_j - \boldsymbol{\mu}_t^c)\|^2 \ . \tag{13}$$

$\mathbf{L}_t^c$ is diagonal as a result of $\mathbf{\Sigma}_t^c$ being diagonal, hence it is clear that the difference $(\mathbf{v}_j - \boldsymbol{\mu}_t^c)$ is scaled at every dimension by a value at the diagonal of $\mathbf{L}_t^c$ at the corresponding dimension. Since kd-trees partition along the standard basis of the Euclidean space, and each node contains the pivot

value for partitioning a dimension, we can re-scale the pivot value of each node with the corresponding scale factor from the diagonal of $\mathbf{L}_t^c$ while descending a tree with $\boldsymbol{\mu}_t^c$. Note that each query image requires building a kd-tree, but the computational cost required is practically negligeable.

Secondly, our goal is not exactly in finding the nearest neighbour of $\mathbf{e}_t^c$ among the $\mathbf{v}_j$'s. We are only interested about if at least one near enough neighbour (i.e. $d(\mathbf{e}_t^c, \mathbf{v}_j)$ is below θ_t^c) exist to decide the output of $h_t^c(\mathbf{I})$. Hence the scaled kd-tree search can be terminated prematurely if the distance of current nearest neighbour with $\mathbf{e}_t^c$ is below θ_t^c.

4.2. Geometric validation

For descriptor distributions $\mathbf{e}_t^c$ selected for a classifier H^c in the boosting process, based on their relative positions computed during the video processing step in §2.3, a nearest neighbour index $\mathbf{W}^c \in \mathbb{R}^{T\times T}$ is built to represent their geometric configuration. The i-th row of the index contains

$$\mathbf{W}^c(i) = [\ W_{i,1}^c\ ,\ W_{i,2}^c\ ,\ \ldots\ ,\ W_{i,T}^c\ ]\ , \qquad (14)$$

where each $W_{i,t}^c \in [\ 1\ T\ ]$ indicates the index of the neighbours of the i-th descriptor distribution arranged in decreasing proximity, i.e. $W_{i,1}^c = i$. Also if $b > a$, the $W_{i,a}^c$-th descriptor distribution is closer in terms of spatial position to $\mathbf{e}_i^c$ than the $W_{i,b}^c$-th descriptor distribution.

When tested against the c-th class, the local features $\mathbf{v}_j$ of a query image will activate some of the descriptor distributions (i.e. $h_t^c(\mathbf{I}) = 1$). The indices of the activated h_t^c's, which range from 1 to T, are retained and a nearest neighbour index $\tilde{\mathbf{W}}^c \in \mathbb{R}^{\tilde{T}\times\tilde{T}}$ is built based on the spatial positions the $\mathbf{v}_j$'s which caused the activations. The value of $\tilde{T}$ depends on how many h_t^c's are activated. We then compute the number of intersections between the corresponding rows of $\mathbf{W}^c$ and $\tilde{\mathbf{W}}^c$ as a measure of geometric consistency.

5. Experimental Results

The dataset. First, we describe the collection process of our place video database. Places of interest (mainly buildings with noticeable facades) in our campus were captured in video in the manner described in §1 and Fig. 1(b). The device used is an off-the-shelf consumer digital camera. In our database, the length of the videos range from 1s to 10s depending on the size of the place, while the framerate is kept at 25 fps. Also depending on the physical size, 3 to 6 video sequences were recorded for each place. In general larger buildings require not only more videos, the sequences are also lengthier. Fig. 5 illustrates the types of places we have collected. We recorded 40 different places which amount to about 21,000 image frames or 1.5GB of data. Videos of the same place were assigned the same class label.

At each place, a separate testing set of *still images* (collectively 1349 images) were also captured in an *unconstrained* manner on different days, from different viewing positions and viewpoints. They are labeled accordingly to the place identity. Note that our digital camera, like most consumer models, captures video in a much lower resolution (480×640) than still images. Artifacts from video coding are also introduced in the video frames. Hence the training and testing set differ in image quality. These factors result in a much harder dataset than the ZuBuD [11]. The pre-processing steps we applied include resizing the images to 240×320 pixels and a colour to greyscale conversion.

Figure 5. The types of places in our video database. These images were extracted from their respective video sequence.

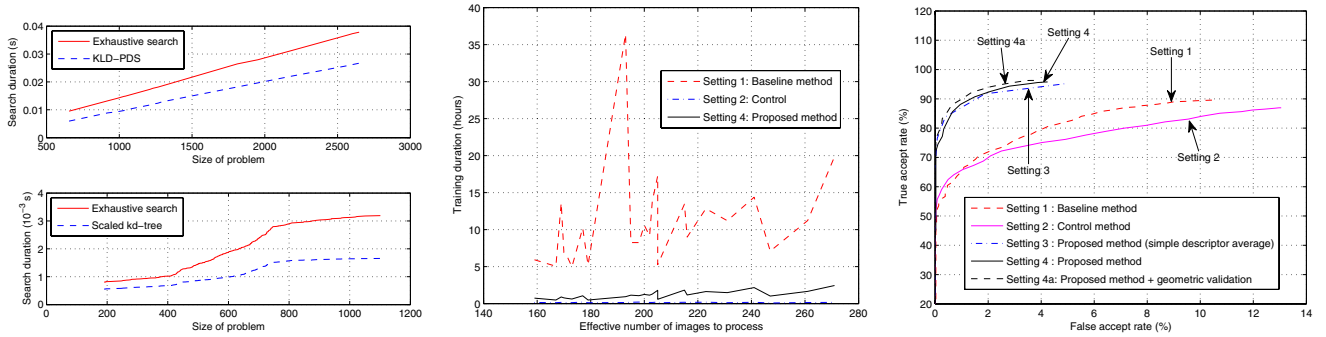
Fast nearest neighbour searches. We first investigate the performance of the fast nearest neighbour routines of KLD-PDS (§3.4) and Scaled kd-trees (§4.1). Since they are conceived primarily for place recognition (KLD-PDS for training, Scaled kd-trees for testing), we analyze them empirically only. More specifically, we directly examine their performance during training and testing of a database of video sequences for place recognition (see Setting 4 of the next subsection). The time required by an exhaustive nearest neighbour search is recorded for comparisons. The resulting times are then plotted against the problem size i.e. the number of distributions/vectors among which to search for the nearest neighbours. Fig. 6(a) illustrates the results.

The divergent curves in Fig. 6(a) indicate that both KLD-PDS and Scaled kd-trees have better scaling characteristics than exhaustive nearest neighbour searches. In terms of search duration, KLD-PDS provided consistent time savings of about 30%, while Scaled kd-trees performed even better with times savings in the range of 20% (problem size $\approx$ 200) to 50% (problem size $\approx$ 1000). Note that problem sizes for the Scaled kd-tree experiments directly correspond to the number of keypoints extracted in the testing images.

Place recognition experiments. Four experimental settings were investigated. They are:

1. Directly apply [10] on the video sequences by treating each video frame individually as single images.

2. Randomly sample a few frames from each video and directly apply [10]. This is to simulate the recording of a place from coarsely sampled viewpoints like in



(a) A comparison of nearest neighbour search times. Top: KLD-PDS, bottom: Scaled kd-trees.

(b) Place recognition experiments: Comparison of training times among 3 settings.

(c) Place recognition experiments: Averaged classification accuracy of 4 settings from 24 subsets.

Figure 6. Experimental results.

Fig. 1(a). For fair comparisons, the number of frames selected per class is the same as the number of video sequences available per class. This is to ensure that Settings 2, 3 and 4 have the same number of samples per class. Recall that for [10], an image is a sample, whereas for our method a video sequence represents a sample (refer to §3.3 and Fig. 4(a)).

3. Apply the procedure in §2 but represent each track with just the mean vector of its descriptors, as in [13]. Train classifiers using [10] with the descriptor mean vector.

4. Process the video sequences according to §2 and train classifiers with the method detailed in §3.

Setting 1 is the baseline against which our method, formulated in Settings 3 and 4, should be compared. In particular, we would expect Settings 3 and 4 to require much less training time than Setting 1 while achieving comparable accuracy. Finally, Setting 2 represents the control experiment.

Evaluating Setting 1 on the full dataset is not even remotely possible on standard hardware. Therefore we divide the dataset into smaller subsets and perform all four settings on each subset. Akin to cross validation, we randomly partition the 40 classes into 8 mutually disjoint subsets, each with 5 classes. The testing set is partitioned accordingly. This preprocessing is not only necessary from a feasibility point of view, but it also reflects the concept of "location priming" in MAR [5], where GPS information is used to narrow down the search space in a particular query. In fact, in a given vicinity there are usually < 10 places of interest.

Random partitioning was carried out 3 times, creating a total of 24 subsets with 5 classes each. The value of T, i.e. the number of weak classifiers per class, was maintained at 500 for all methods and subsets. For each subset, the effective number of images to process is the total number of frames from all video sequences in the subset. We plot the training durations of Settings 1, 2 and 4 against this value in Fig. 6(b) (Settings 3 and 4 have comparable training duration). Setting 2 is naturally the fastest since it merely samples a small number of images to train. Performing Setting 1 is the most time consuming, with training durations at least twice that of Setting 3, with some even surpassing a day. Setting 3 can be evaluated comfortably within 2.5 hours. This represents a dramatic improvement of training efficiency for the AdaBoost method. Note that the video processing steps of §2 require negligeable time (in the order of tens of minutes) relative to the overall training time.

We now examine the classification accuracy. The results from all subsets are averaged and their Receiver Operating Characteristic (ROC) curves are shown in Fig. 6(c). The ROCs were obtained by varying the confidence threshold on the output of Eq. (1). As expected, Setting 1 outperformed Setting 2 since the latter has only a subset of the input data. Hence we conclude that densely imaging a place is useful for training better classifiers. Next, it turns out that both Settings 3 and 4 significantly outperformed Setting 1. The video processing steps, which were originally conceived for feature reduction, are actually beneficial for classification accuracy. This is maybe because the more consistent keypoints identified are more effective for classification.

Setting	Training time	True/False accept rate
1	11.1013 hrs	89.56 / 10.44 %
2	0.1504 hrs	86.95 / 13.05 %
3	0.9608 hrs	95.13 / 4.87 %
4	1.1780 hrs	95.76 / 4.24 %
4 + Geom. val.	1.1780 hrs	96.28 / 3.72 %

Table 1. Summary of place recognition experiment results. All values are averaged from the 24 subsets. The true/false accept rates correspond to the threshold of zero, i.e. no threshold is applied.

Also, Setting 4 slightly outperformed Setting 3, indicating that it is advantageous to use descriptor distributions rather than averages. The extra memory and computational requirements during classification are only minor as described in §4.1. Using our implementation in Matlab, test-

ing an image requires on average 1.5 seconds, with the bottleneck occuring at the keypoint (SIFT) extraction process. This can be further improved by using smaller values for T (we discovered that setting $T = 100$ does not substantially reduce accuracy). Finally, we observe that the simple geometric validation method introduced in §4.2 can further improve the accuracy of place recognition, as shown by Setting 4a in Fig. 6(c). Table 1 summarizes the results.

Towards Mobile Augmented Reality (MAR). We tested the viability of our ideas for a practical MAR system. Place recognition classifiers were trained for a particular location in a city. In total six distinct places of interest were included. We then capture a query video within the vicinity of the location. Each frame (240×320) in the video is subjected *offline* to the trained classifiers ($\approx 1.5s$/image in Matlab). Images which have positive classification are tagged with the respective identity of the place recognized. The tag is attached to the image at the position which correspondes to the *median* of the x and y coordinates of the keypoints which activated the weak classifiers of the class with positive output. Fig. 7 illustrates several frames of the results. The places were robustly indentified under multiple viewpoints, and the recognition speed demonstrates a significant potential for practical application.

Figure 7. Recognition results from an MAR system.

6. Conclusions

The experimental results support our idea that having more visual information about a place contributes towards constructing more robust and accurate place recognition. Hence we advocate using dense video recordings of facades of places, instead of just several images from a few viewpoints, to train classifiers for place recognition. The results also confirm that our proposed solution for training classifiers from the video sequences is much more efficient than a straightforward application of the AdaBoost method for generic object recognition [10] on the video frames. In fact, the resulting classifiers are also more accurate.

We plan to implement the recognition algorithm in §4 on a mobile device to allow *in situ* testing to more comprehensively examine of the performance of the trained classifiers.

References

[1] D.-Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham. Fast search algorithms for vector quantization and pattern matching. In *ICASSP*, 1984.

[2] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[3] G. Fritz, C. Seifert, and L. Paletta. A mobile vision system for urban detection with informative local descriptors. In *ICVS*, 2006.

[4] R. Hartley and A. Zisserman. *Multiple view geometry in comp. vision*. Cambridge Uni. Press, 2nd edition, 2003.

[5] E. Jonietz. TR10: Augmented reality. Technical report, MIT Technology Review, 2007.

[6] S. Khan, F. Rafi, and M. Shah. Where was the picture taken: Image localization in route panoramas using epipolar geometry. In *ICME*, 2006.

[7] F.-F. Li and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

[8] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[9] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[10] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *PAMI*, 28(3):416–431, 2006.

[11] H. Shao, T. Svovoda, and L. Van Gool. ZuBuD— Zurich buildings database for image based recognition. Technical report, Computer Vision Lab, ETH Zurich, 2003.

[12] G. C. Sharp, S. W. Lee, and D. K. Wehe. Icp registration using invariant features. *PAMI*, 24(1):90–102, 2002.

[13] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *CVPR*, 2003.

[14] J. Sivic and A. Zisserman. Video data mining using configurations of viewpoint invariant regions. In *CVPR*, 2004.

[15] G. Takacs, V. Chandrasekhar, B. Girod, and R. Grzeszczuk. Feature tracking for MAR using video coder motion vectors. In *ISMAR*, 2007.