Visual Word based Location Recognition in 3D models using Distance Augmented Weighting

Friedrich Fraundorfer[1], Changchang Wu[2], Jan-Michael Frahm[2], Marc Pollefeys[1,2]
[1]Department of Computer Science
ETH Zürich, Switzerland
{fraundorfer, marc.pollefeys}@inf.ethz.ch

[2]Department of Computer Science
UNC Chapel Hill, USA
{ccwu,jmf}@cs.unc.edu

Abstract

For visual word based location recognition in 3D models we propose a novel distance-weighted scoring scheme. Matching visual words are not treated as perfect matches anymore but are weighted with the distance of the original SIFT feature vectors before quantization. To maintain the scalability and efficiency of vocabulary tree based approaches PCA compressed SIFT feature vectors are used instead of the original SIFT features. A different eigenspace is computed for each vocabulary tree cell to benefit from the variance reduction as result of the partitioned SIFT feature space. Experiments show a significant improvement in retrieval quality by incorporating the distance with small costs in computational time and memory.

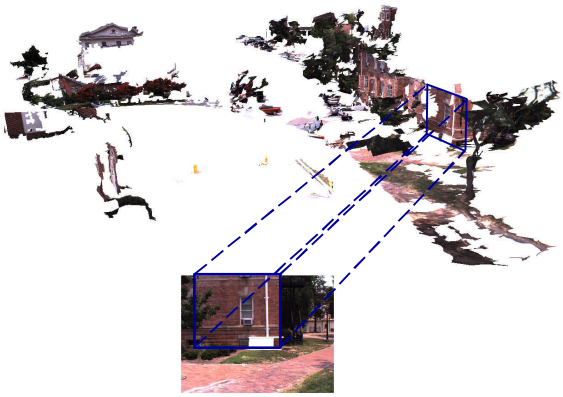


Figure 1. Given a 3D model as a map of the environment we are interested in the location from which a query image has been taken.

1 Introduction

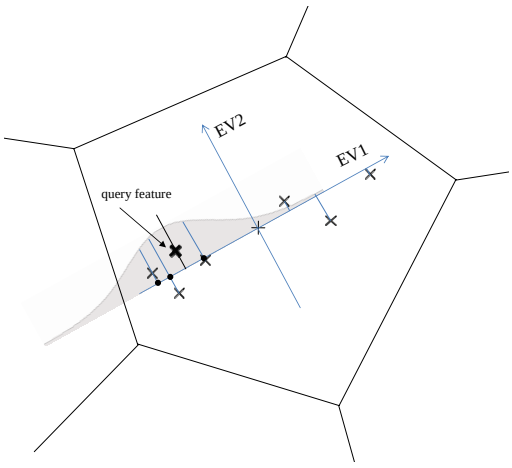
Being able to create large scale 3D reconstructions [1], e.g. from cities, opens up the possibility to do location recognition and pose estimation from a a single image taken somewhere within the mapped area. The most prominent application is certainly localization with hand-held devices, like cell phones. However, other applications include loop closing for mapping as well as stitching of multiple independent 3D reconstructions. The key idea for the current approaches to location recognition is the use of visual word based image search [2, 6, 8]. They proved to be scalable to much larger image databases than previous bag-of-features methods which mainly relied on nearest-neighbor search in feature space [3, 4, 9, 5]. In most cases a kd-tree was used to find the nearest-neighbor to the query feature. Then a vote was cast to the according document in the database. The distance between the database features and the query feature was used as a weighting factor for the vote. Other methods define a distance threshold for votes or vote for the k closest nearest-neighbors [4]. For efficient queries theses methods need the original feature vectors stored in memory. This quickly leads to scalability problems e.g. when using 128-dimensional SIFT feature vectors that require 128 bytes of memory in most implementations. For large databases the image queries turned out to take too long for many applications. The recent visual word based approaches effectively cut down the memory requirements and query times for image search applications. The main idea is to quantize a higher-dimensional feature vector into a single integer value (visual word) which allows very compact image representations and enables highly efficient retrieval methods, e.g. using an inverted file as database. Typically quantization is performed by splitting the high-dimensional feature space into multiple non-overlapping partitions by k-means clustering. Each resulting cluster cell is then mapped to a visual word. A vote is cast for all the images in which the same visual word appears in the query, which is efficiently

done through an inverted file. In contrast to the previous nearest-neighbor methods the distance of the query feature and the database feature is now unknown. It is not possible to weight the vote according to the distance or take the k closest features. In addition the visual word cells in the feature space are usually much larger than the variation of a single SIFT feature and contain large quantities of visually significantly different features. Fig. 2(b) illustrates this fact. It shows representative patches that fell into the same vocabulary tree cell. While some of them are very similar, others are significantly different.

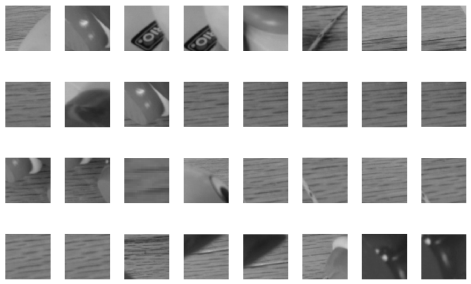
We therefore propose a weighted scoring scheme for visual word voting based on the distance between features in a vocabulary tree cell. Fig. 2(a) illustrates the method. The distance between the query feature and the features in the cell defines the strength of the individual vote. To maintain the scalability of the vocabulary tree approach, PCA compressed feature vectors are stored instead of the original SIFT feature vectors. The PCA compression is computed individually for each vocabulary tree cell to make use of the variance reduction by the feature space splitting. The distance weights are seamlessly integrated into the scoring scheme and require only a small computational overhead.

2 Related work

In recent years a lot of progress has been achieved in object recognition and image search. The object recognition system presented in [3] represents a bag-of-features approach using SIFT features. It uses a kd-tree for nearest neighbor search with a best-bin-first modification to increase the query speed. It also uses the distance between features to discriminate between reliable matches and likely mismatches. The distance ratio between the first and second nearest neighbor is computed. If the second nearest neighbor is far enough away from the first nearest neighbor the match is accepted. The necessary distance ratio is empirically set. The approach presented in [8] quantizes SIFT features into visual words. Matching is done by voting, the distance between corresponding features is not used. In [6] this approach gets extended by a hierarchical quantization with a vocabulary tree. It also introduces a hierarchical scoring. It demonstrates a very efficient and highly scalable image search. In [2] the hierarchical vocabulary tree is replaced again by a flat vocabulary tree. An approximate nearest neighbor search is used to find the cells for quantization. Schindler et al. [7] proposed the greedy N-best-path algorithm for feature quantization to deal with features close to the decision boundaries. Matching features that lie close to decision boundaries might end up with different visual words otherwise. This is especially problematic in hierarchical vocabulary trees where small shifts in higher levels might lead to completely different visual words. In addition



(a)



(b)

Figure 2. (a) We propose to weight the votes with the distance to the query feature. For efficiency the distance is computed from PCA compressed features and the weights are following a Gaussian bell curve. (b) Example image patches that fall into the same vocabulary tree cell. Some of them are significantly different.

Schindler et al. [7] proposed to build the vocabulary tree from the most informative features only to improve recognition quality. In contrast to all the previous techniques our novel approach efficiently employs the feature distances in a vocabulary tree. This joins the benefits of the two different classes of matching approaches and leads to an improved recognition performance as shown in our experiments.

3 Image search with a vocabulary tree

Our approach is along the lines of the method described in [6]. Firstly local features are extracted from images. We use DOG keypoints and compute a SIFT feature vector for

each keypoint. Each SIFT feature vector is quantized by a hierarchical vocabulary tree. All visual words from one image form a document vector which is a v-dimensional vector where v is the number of possible visual words. It is usually extremely sparse. For an image query the similarity between the query document vector to all document vectors in a database is computed. As similarity score we use the L_2 distance between document vectors. The organization of the database as an inverted file and the sparseness of the document vectors allows a very efficient scoring. For scoring, the different visual words are weighted based on the inverse document frequency (IDF) measure. We introduce an additional weight based on the distance between the individual query features and the features in the same vocabulary tree cell. For scalability we don't store the original feature vector but store PCA compressed features vectors to compute an approximate distance. The PCA compression is computed for each vocabulary tree cell differently to exploit the variance reduction by the clustering. The database images corresponding to the document vector with the lowest L_2 distance to the query vector is reported as the best match.

4 Distance augmented weighted scoring

As described in [6] the L_2 distance between the query document vector q and a database document vector d can be computed as

$$||q-d||_2 = 2 - 2\sum q_i d_i \tag{1}$$

For simple scoring the entries q_i, d_i are set to 1 if the VW i occurs in the image otherwise they are set to 0 and normalized afterwards so that the L_2-norm is 1. The document vectors are high-dimensional but very sparse and so only the non-zero products ($q_i \neq 0, d_i \neq 0$) are usually computed, which is implicitly done by inverted file scoring.

When using inverse document frequency (IDF) weighting, as we do in this work, the entries of the document vectors are normalized IDF weights. We denote the IDF weighted document vectors by $\overline{q}$, $\overline{d}$. The IDF weight $m(i)$ for each visual word i is computed by

$$m(i) = \log \frac{|D|}{|D_i|}, \tag{2}$$

where $|D|$ is the total number of documents (i.e. images) in the database and $|D_i|$ is the number of images in which the visual word i appears.

Accordingly a document vector entry $\overline{q}_i$ is given by

$$\overline{q}_i = \begin{cases} \frac{m(i)}{\sqrt{\sum_{j|q_j\neq 0} m(j)^2}} & , q_i \neq 0 \\ 0 & ,\text{otherwise} \end{cases} \tag{3}$$

and $\overline{d}_i$ by

$$\overline{d}_i = \begin{cases} \frac{m(i)}{\sqrt{\sum_{j|d_j\neq 0} m(j)^2}} & , d_i \neq 0 \\ 0 & ,\text{otherwise} \end{cases} \tag{4}$$

Each entry in $\overline{q}_i$, $\overline{d}_i$ is the IDF weight normalized by the square root of the quadratic sum of all IDF weights that occur in the image, e.g. for $\overline{q}$ all $m(i)$ where $q_i \neq 0$. The denominator is a normalization so that $||\overline{q}||_2 = 1$.

We now describe how to incorporate the proposed distance weight in the scoring process. We assume that the distance between two instances of the same SIFT feature is a normal distribution with $N(\mu, \sigma)$ and $\mu = 0$. Then the match probability p is computed as

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{5}$$

where x is the L_2 distance between two SIFT feature vectors, $x = ||s_i - s_j||$. We want the actual weight w to be in the interval $[0...1]$, so we do not apply the normalization by $\sigma\sqrt{2\pi}$ in Equ. (5). Accordingly the weight w is computed as

$$w(x) = e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{6}$$

The proposed scoring process now computes the weighted distance between document vectors $\overline{q}$ and $\overline{d}$ using the weight defined in Equ. (6). This writes as

$$||\overline{q} - \overline{d}||_2 = 2 - 2\sum \overline{q}_i \overline{d}_i w(\text{dist(i)}), \tag{7}$$

where dist(i) is a function that computes the SIFT feature distance for the visual word matches of $\overline{q}_i$ and $\overline{d}_i$.

The effect of the weight w is that an increased distance between features leads to a decreased similarity between document vectors. Adding unweighted $\overline{q}_i\overline{d}_i$ terms for all the features in a cell wrongly increases the similarity of non-matching document vectors. These erroneous votes get corrected by the distance weight w from Equ. (6). A feature that is far away from the query feature is down-weighted with a w of almost zero. This means that the feature match is practically not considered in the decision process during the query, as if the feature would have a different visual word. Thus the similarity between the corresponding document vectors does not increase wrongly. For features very close to the query feature the weight w is nearly one and results in a full vote.

5 PCA compression of feature vectors

In the previous section we introduced our distance augmented weighting in the vocabulary tree. The weight hereby was computed from the distance between feature

vectors. It would require to keep those vectors in memory which is a significant drawback. We analyzed the distribution of the feature descriptors in the cells of the vocabulary tree and found that they typically can be represented by a lower dimensional space. Accordingly the original SIFT feature vectors are PCA compressed first before storing them in the database resulting again in a memory efficient approach. A PCA analysis is performed for each vocabulary tree cell thus exploiting the variance reduction as result of the partitioned SIFT feature space. Let S_i be the set of all SIFT feature vectors for the vocabulary tree cell i. S_i is a $128 \times n_i$ matrix where n_i is the number of SIFT features in the cell. The covariance matrix C_i for cell i is a 128×128 matrix. PCA analysis gives the eigenvectors V_i (128×128) and the eigenvalues of C_i. The SIFT feature vectors can be projected into PCA space by multiplication with the eigenvectors V_i. The eigenvectors with the largest eigenvalues define a subspace that contains the principal variations of the SIFT features in the cell. By projecting each SIFT feature into a subspace with a smaller dimension than 128 we obtain a compressed SIFT feature vector. Let V_i' be the $k \times 128$ matrix that contains the k eigenvectors that belong to the k largest eigenvalues (with typically $k << 128$). A compressed SIFT feature vector is computed as

$$s_j' = V_i'(s_j - u_i), \tag{8}$$

where u_i is the mean vector of the cell features used to compute C_i. The distance $d_a(j,l)$ between two compressed feature vectors is an approximation of the original distance of two feature vectors s_j and s_l which are located in the same cell.

$$d_a(j,l) = ||V_i'(s_j - u_i) - V_i'(s_l - u_i)||_2. \tag{9}$$

If most of the variation is contained in the first k eigenvectors then d_a is a good approximation of the exact distance. Fig. 3 shows an analysis of the total cell variation for vocabulary trees with different number of cells. The analysis shows that in a 10^5 vocabulary tree more than 90% of the total variation is described by already 40 eigenvectors. The analysis also shows that the clustering of the feature space reduces the variation within the cells.

Important for the search efficiency is that the eigenvectors for each cell can be computed during vocabulary tree clustering which is an off-line training.

6 Implementation details

In the most basic vocabulary tree recognition scheme only one integer value (4 bytes) is stored for each image feature. In most cases however additional data is stored as well, e.g. x, y coordinates for geometric verification etc. The proposed use of distances in scoring needs additional data, too.

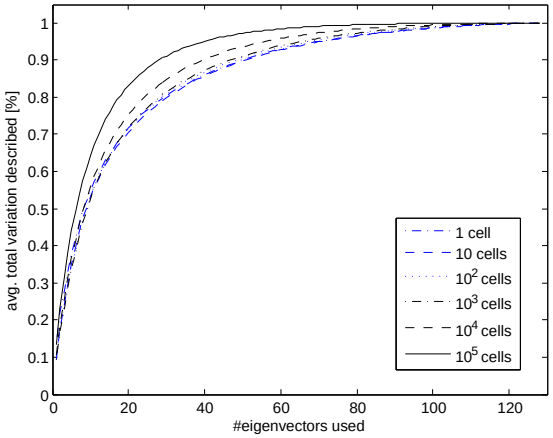


Figure 3. Total variation of vocabulary tree cells (avg. over all cells at the same level). A smaller number of eigenvectors is necessary to describe the cell variation in vocabulary trees with higher number of cells.

Firstly, for each visual word (VW) in the database the corresponding compressed feature vector needs to be stored. The original SIFT feature vector consist of 128 entries each with a range from $0..255$ thus we would need 128 bytes. The compressed feature vector has a much lower dimension, e.g. 10 dimensions as used in our experiments. The compressed feature vector will be stored as signed bytes, cutting off all values higher and lower than the maximum range. Thus we need 1 byte for each dimension, i.e. 10 bytes for the 10 dimensional case. It is important to keep the amount of additional data low as it grows with the number of features in the database. Secondly, we need to store the eigenvectors to compute the compressed SIFT vectors. For a k dimensional compressed SIFT feature we need to store a $k \times 128$ eigenvector matrix, consisting of $128 \times k$ float values, resulting in $512 \times k$ bytes for each cell. We store a different set of eigenvectors for each possible VW. However, the number of VW's is determined by the vocabulary tree and remains fixed. It is therefore a constant cost. In addition it is most likely that not the whole range of VW's is actually used. In an application where the database remains fixed the VW's used can be pre-computed and eigenvectors need to be loaded only for this set. This holds even when new VW's are used in the query. As they have no matching VW's in the database they do not contribute to the score. It is therefore not necessary to compress them.

Additional computations are necessary for scoring and when online adding new images to the database. During the add process the SIFT feature vectors have to be compressed. This needs a $k \times 128$ matrix multiplication for each SIFT feature detected in the new image. For an image query the

same computation has to be done. For each SIFT feature in the query image the multiplication with the eigenvectors has to be carried out. In the scoring itself the distances between the compressed query vector and the database vectors have to be computed. We use the L_2 norm. For each VW in the query image we compute distances to each entry in the corresponding inverted file bin. However, the distance computation with the compressed feature vectors needs much less calculations compared to distance computation using the original SIFT features.

6.1 Setting of the σ parameter

The best way to determine the σ parameter for the weighting function Equ. (6) is to conduct a limited number of recognition experiments with varying σ in a small range. To find a useful range for the σ value we used a histogram of the distances between the features in the used vocabulary tree. Only distances between features in the same cell are computed. Fig. 4 shows the histogram. Most of the distances occur in the interval [20..200], so this is the range where a meaningful value for σ can be determined. Table 1 lists the values we found to be useful for our experiments. The values vary with the number of eigenvectors used for compression.

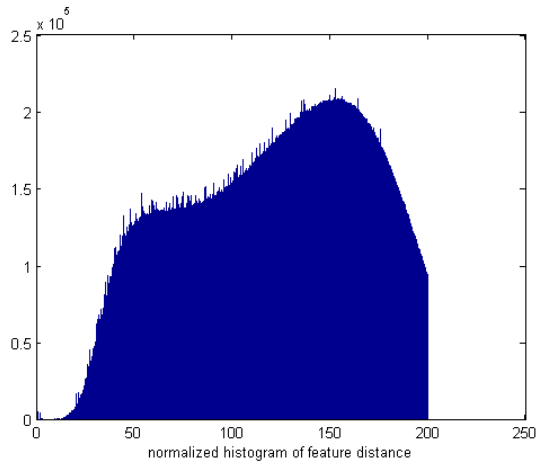


Figure 4. Histogram of feature distances for all the cells of the 10^5 vocabulary tree. The range of occurring distances allows to determine a range of values for the σ parameter of the weighting function.

7 Efficient 2-pass scoring

A major speedup in querying can be gained by a 2-pass scoring scheme that combines the standard scoring with our weighted scoring. For a query, firstly standard scoring is used to generate a similarity ranked document list. Then weighted scoring is applied in a second pass but considering only the n-top ranked documents. The major computational overhead for our method comes from the distance computation between the feature vectors. In the 2-pass method the distances are only computed for features that occur in the n-top ranked documents. This leads to a significantly reduced computational overhead. Our experimental results (see section 8.1 show no loss in retrieval quality compared to the significantly slower 1-pass scoring scheme.

method	sigma
compressed (10 eigen.)	40
compressed (20 eigen.)	55
compressed (40 eigen.)	65
distance scoring (exact)	110

Table 1. σ parameter settings used in our experiments.

8 Experiments

8.1 UKY dataset

We tested our approach on the well known UKY benchmark dataset[1]. The dataset contains 4 images each of 2550 objects from different viewpoints and with changing scale resulting in 10200 images. Our initial experiment uses a database of 2550 images out of the 10200 images representing one view of each object. The query dataset of 2550 images uses a different view of each object. For SIFT feature detection we used the publicly available SiftGPU software[2] which resulted in up to 3000 features per image.

For the retrieval quality we count the number of correct best matches (top-1 rank). Table 2 shows the results and timings for the standard scoring (IDF weighted non-hierarchical scoring [6]), our new method (with different compression settings) and an exact distance scoring. The results show that even with the highest compression of using only 10 dimensions (out of 128) a significant improvement with respect to the standard scoring scheme can be achieved. In addition it is shown that the quality achieved with 40 dimension is very close to the quality of the exact distance scoring where all the 128 dimensions were used. The computational overhead of our method is visible in the timings. The listed execution times are the average for a single image query and include the time for feature quantization. Feature quantization accounts for a substantial part

[1] Available at http://vis.uky.edu/~stewe/ukbench/
[2] SiftGPU available at http://cs.unc.edu/~ccwu/siftgpu/

method	retrieval quality	query time [ms]
standard scoring	0.645	21.7
compressed (10 eigen.)	0.71	40.3
2-pass (10 eigen.)	0.71	28.6
compressed (20 eigen.)	0.726	49.2
2-pass (20 eigen.)	0.726	31.8
compressed (40 eigen.)	0.737	66.4
2-pass (40 eigen.)	0.732	39.1
distance scoring (exact)	0.739	56.9
feature quantization	-	19.6

Table 2. Retrieval quality on database with 2550 images (subset of UKY dataset) compared to the standard approach. The results show that the quality achieved by the compression with 40 eigenvectors is very close to the exact method. A vocabulary tree (10^5 leaves, 5 levels, 10 branches) trained on a different image set was used. The execution times in the table are for a single image query and include the time needed for feature quantization.

of the query time for a database of this size. From Table 2 it can be seen that the 2-pass method provides a significant performance gain without loss of retrieval quality. The execution times were measured on a 2.4GHz Intel Quadcore with 4GB of memory. Even if we look at the top-10 or top-100 best ranked images the proposed approach has an advantage, although it is less significant. However, we find it important that correct matches are ranked as top-1 if possible. To identify a correct match in the top-n ranks, additional post-processing like geometric verification is necessary which might be fast and reliable for certain applications (e.g. CD demo in [6]) but might be difficulty and slow for other scenarios, e.g. non-static scenes, multiple moving objects, etc. For visual word quantization we use a vocabulary tree that was trained from 10 million features extracted from independent images from the web . The trained vocabulary tree has 5 levels and a branching factor of 10 which results in 10^5 leaf nodes. The number of leaf nodes has been chosen based on the analysis in [6]. It was shown that good retrieval performance can be reached with 10^5 leaf nodes and that there is not a significant increase when using more nodes.

The second experiment was conducted on a smaller database, consisting of the first 125 objects from the UK dataset. The results are listed in Table 3. Although the retrieval quality is already very high for the standard approach, our method still improves the results significantly.

method	retrieval quality	query time [ms]
standard scoring	0.814	17.2
compressed (10 eigen.)	0.864	25.0
compressed (20 eigen.)	0.888	31.1
compressed (40 eigen.)	0.888	39.4
distance scoring (exact)	0.896	25.7
feature quantization	-	16.0

Table 3. Retrieval quality on database with 125 images (subset of UKY dataset) compared to the standard approach. Although the result with the standard approach is already very good it can still be improved with the distance weighting. A vocabulary tree (10^5 leaves, 5 levels, 10 branches) trained on a different image set was used. The execution times in the table are for a single image query and include the time needed for feature quantization.

8.2 Location recognition in urban environments

One of our main motivations for this work was to improve the performance for location recognition in urban environments. The image sets for this experiments were acquired with car mounted cameras. Two cameras were mounted on the roof of a car, one was pointing straight side wards the other one was pointing forward in a 45° angle. The fields of view of both cameras do not overlap but as the system is moving over time the captured scene parts will overlap. To retrieve ground truth data for the camera motion the image acquisition was synchronized with a highly accurate GPS-INS system. Accordingly we know the location of the camera for each image. In this experiment we now match the images from the side camera with the images from the forward camera. Usually the area that is viewed in the forward camera can be seen in the side camera after about 30 frames of straight movement, but of course with a quite large viewpoint change of 45°. The images have a resolution of 1024×768. For feature extraction we used DOG keypoints and SIFT descriptors as in the previous experiments which resulted in up to 5000 features per image. And we reused the vocabulary tree trained for the previous experiment in Section 8.1. We created a database of 2644 images from the side camera and queried them with the 2644 images from the forward camera using the standard scoring and our new method with 10 eigenvectors. The location recognition results are visualized by plotting lines between the matching camera positions (see Fig. 5). The

identical camera paths of the forward and side camera are shifted by a small amount in x and y direction to make the matching links visible. We only draw matches below a distance threshold of 20m. Matches with a distance lower than 20m are counted as correct and matches with higher distance are counted as mismatches. While we can't guarantee that all matches below 20m are actually correct we can be almost certain that matches with a distance higher than 20m are wrong. We therefore evaluate the number of mismatches for the standard method and the proposed method. The standard approach counts 54% mismatches while our proposed method reduces the mismatches to 40%, which is a 26% reduction of mismatches. Fig. 5(a) shows the location recognition results for the standard approach. Fig. 5(b) shows the results for our approach. Much more matches are below the 20m threshold which results in more links drawn. Fig. 5(c) shows the corresponding distance histograms. Fig. 6 shows some correct example matches.

We have implemented a location recognition tool using the proposed technique. By specifying a query image, a database query is performed and the corresponding part of the 3D model is shown (see Fig. 7). We also queried the database with images from a camera phone with good results (see Fig. 7(b) as an example).

9 Conclusion

We proposed a novel distance augmented weighting scheme for visual word based image search schemes. The proposed method significantly improves the retrieval performance compared to the standard approach. To maintain the scalability of the vocabulary tree approach we introduced PCA compressed features. Each vocabulary tree cell is compressed individually to exploit the sparseness introduced by the vocabulary tree clustering. Our experiments showed a significant retrieval quality improvement using our new method that comes with only a small increase in memory requirement and computation time especially when using the proposed 2-pass method. And we successfully applied this method for location recognition in 3D models.

10 Discussion

Finally, we would like to draw your attention to an interesting and quite surprising observation. To obtain the best retrieval results the σ parameter needed to be varied more than anticipated with the changing number of eigenspace dimensions used (see Table 1). The σ parameter determines basically the separating function between matches and mismatches or between intra-features (matching features under image transformations, e.g. noise, scale change, viewpoint change, ...) or inter-features (visually different, not-matching features). We initially thought that we would have

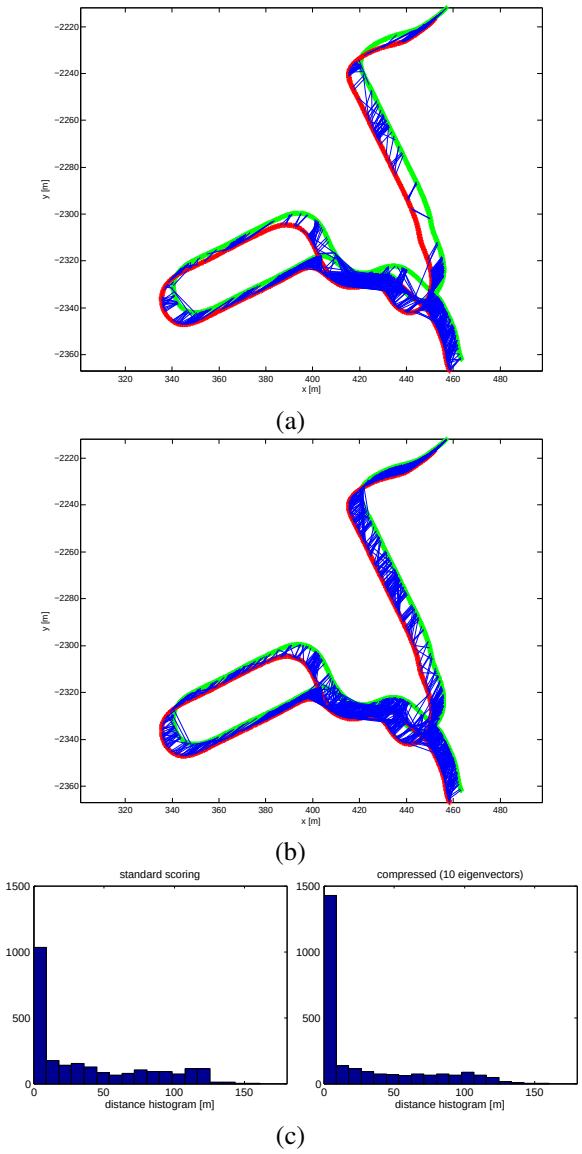
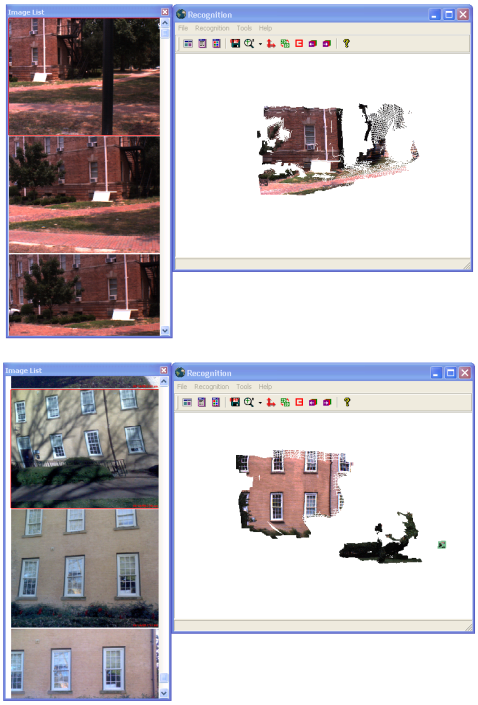


Figure 5. Location recognition experiment. Lines show matches (below a 20 m distance threshold) between the forward and side camera. Our approach (b) shows much more matches than the standard approach (a). (c) Distance histograms show that more matches have lower distances.

to vary σ according to the total variation captured by the k-best eigenvectors (see Fig. 3), e.g. if 10 eigenvectors capture 65% of the total variation we intended to set the corresponding σ to 65% of the exact distance σ. The values used however do not follow the total variation, instead it looks like that σ^2 roughly follows linearly the number of dimension of the eigenspace, i.e. $\sigma^2 = 1600, 3025, 4225$ and 12100 for

Figure 6. Example matches from the location recognition experiment. Left column: Forward camera image (query). Right column: Side camera image (database).



(a)

(b)

Figure 7. (a) Screenshot of our location recognition tool. (b) Query with an image from a camera phone.

dimensions 10, 20, 40, 128. While we don't have proof, a possible explanation could be, that the inter-feature distribution and intra-feature distribution are not correlated. In that case the k-best eigenvectors from our PCA analysis (which was mainly computed from inter-features) might only represent unordered random directions for the intra-feature distribution. Thus while the k-best eigenvectors capture most of the variation of the inter-features they might only capture a small part of the intra-feature variation. If we assume that the k-best inter-feature eigenvectors only sample random directions from the intra-feature distribution then the k-best eigenvectors will, at average, only capture a $1/k^{th}$ part of the total variance, which would explain the almost linear relation of σ^2 with the number of dimensions.

References

[1] A. Akbarzadeh, J. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys. Towards urban 3d reconstruction from video. In *3D Data Processing, Visualization and Transmission*, pages 1–8, 2006. 1

[2] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007. 1, 2

[3] D. Lowe. Object recognition from local scale-invariant features. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1150–1157, 1999. 1, 2

[4] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1

[5] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proc. 8th International Conference on Computer Vision, Vancouver, Canada*, pages I: 525–531, 2001. 1

[6] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, New York City, New York*, pages 2161–2168, 2006. 1, 2, 3, 5, 6

[7] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, Minnesota*, pages 1–7, 2007. 2

[8] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. 9th International Conference on Computer Vision, Nice, France*, pages 1470–1477, 2003. 1, 2

[9] T. Yeh, K. Tollmar, and T. Darrell. Searching the web with mobile images for location recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, pages II: 76–81, 2004. 1