GEOMETRY-BASED RANKING FOR MOBILE 3D VISUAL SEARCH USING HIERARCHICALLY STRUCTURED MULTI-VIEW FEATURES

David Ebri Mars, Hanwei Wu, Haopeng Li and Markus Flierl

School of Electrical Engineering
KTH Royal Institute of Technology, Stockholm
{dem, hanwei, haopeng, mflierl}@kth.se

ABSTRACT

This paper proposes geometry-based ranking for mobile 3D visual search. It utilizes the underlying geometry of the 3D objects as well as the appearance to improve the ranking results. A double hierarchy has been embedded in the data structure, namely the hierarchically structured multi-view features for each object and a tree hierarchy from multi-view vocabulary trees. As the 3D geometry information is incorporated in the multi-view vocabulary tree, it allows us to evaluate the consistency of the 3D geometry at low computational complexity. Thus, a cost function is proposed for object ranking using geometric consistency. With that, we devise an iterative algorithm that accomplishes 3D geometry-based ranking. The experimental results show that our 3D geometry-based ranking improves the recall-datarate performance as well as the subjective ranking results for mobile 3D visual search.

1. INTRODUCTION

In recent years, the development of wireless mobile devices and virtual reality has raised the interest in mobile visual search [1] [2] [3]. It aims to provide an augmented reality in a real-world environment by utilizing the methods of image based information retrieval. Current mobile visual search solutions achieve search results based on the appearance of objects in images captured by mobile devices. These solutions fail if different real objects appear similar in the captured images. To solve this problem, mobile 3D visual search captures not only the visual appearance of query objects, but uses also the underlying 3D geometry [3].

To obtain the 3D geometric information of an object, multi-view imagery may be used. A hierarchical structure of multi-view features allows the selection of image features which is based on feature correspondences across multiple views [4]. Features with well-established correspondences are more robust for matching with query features. Further, by utilizing relevant multi-view feature correspondences, it is possible to achieve an improved ranking performance while using a smaller number of image features.

For robust image retrieval, multi-view scalable vocabulary trees have been used in [5]. Multiple vocabulary trees may be generated with respect to different view perspectives. This method measures the distortion of the descriptors when there is a change of perspective. It is suitable for planar objects such as CD cover images where all feature points are located on the same plane. It has been shown that using multi-view scalable vocabulary trees can significantly increase the image retrieval performance when compared to using a single view approach. This motivates us to utilize multi-view scalable vocabulary trees to handle the 3D geometry of objects.

On the other hand, geometric consistency verification is usually used to evaluate the matches of the query in the database vocabulary tree. RANSAC [6] based methods may be used to distinguish the outliers from the inliers by exploiting statistical consensus. However, the speed of the algorithm depends heavily on the computational complexity of RANSAC. A fast geometric ranking method has been proposed in [7]. It evaluates the geometric consistency by exploiting the geometric similarity between matched features. With 3D geometric information of the object, we can efficiently extend this method to features in 3D space, which are more robust to perspective transformation of objects.

2. MULTI-VIEW VOCABULARY TREES

Large-scale objects such as buildings are usually hard to match due to significant change of viewpoint and lighting conditions between query and server. Thus, an image database at the server with a considerable perspective diversity will improve the performance of mobile 3D visual search. To match a query with the corresponding object at the server, well-known vocabulary tree (VT) methods [8] [9] have been widely used. It can efficiently index image features from a large number of different objects. However, the flexible adaptation of vocabulary tree to handle different perspectives of the same object becomes a crucial problem.

2.1. Hierarchically Structured Multi-View Features

To manage image features from different perspectives, we aim at aligning multi-view feature correspondences. Each of our multi-view feature correspondences is associated with images from different perspectives. To efficiently implement that, we utilize our earlier work [4] to generate a set of hierarchically structured multi-view features for each object.

As promoted multi-view features, which are associated with more images, are more reliable and representative for robust matching, we structure sets of feature correspondences in a hierarchical manner. In particular, sets of feature correspondences with l images are placed on level l. Usually, the number of such sets on higher levels is smaller than that on lower levels. An example for hierarchical sets of features with four levels from four views is shown in Fig. 1. For each set of corresponding features among view images, we take the median of their descriptors as the representative descriptor. More details of our hierarchical multi-view features are discussed in [4].

We see three advantages of using our multi-view features. First, it has been shown in [4] that increasing the number of view images can improve the recall-datarate performance of mobile visual search. Second, the geometric information of the features can be exploited for geometric consistency verification. Third, the use of the representative descriptors reduces the redundancy and improves memory efficiency.

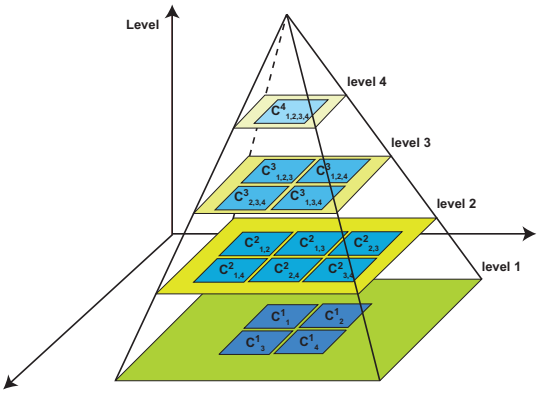


Fig. 1. Hierarchical sets of features with four levels from four views. $C^l_{i,j,\ldots,k}$ defines a set of feature correspondences using l images with image indices $i, j, \ldots, k$.

2.2. Construction of Vocabulary Trees

With hierarchically structured multi-view features, we are able to generate vocabulary trees [8] which are based on multi-view feature descriptors. Both hierarchical and tree structures provide us an advantage when constructing a memory-efficient representation.

To construct the tree, we apply recursively the *k-means* clustering method [10] on the representative descriptors. Note that we use the Euclidean distance for representative descriptors. We generate a D level tree with K branches at each level. Each node of the tree is known as a visual word and the nodes at bottom level are called leaf nodes. The visual vocabulary is composed of the visual words in the leaf nodes. It has been shown in [8] that the memory occupation of the vocabulary tree is linear in the number of leaf nodes K^D. If the N-dimensional visual word is represented by single precision in MATLAB, the size of the tree is approximately $4NK^D$ bytes.

The combination of D and K is given and determined by the requirement of memory usage. Let the total number of descriptors which are used to generate the tree be denoted as M. Generally speaking, the more descriptors are involved in constructing the tree, the higher is the chance that the query features can be matched correctly. However, the number of descriptors which are associated with the leaf nodes is monotonically increasing with respect to the number M. Thus, the risk of outliers becomes relatively high due to the characteristic of the tree matching algorithm, as explained in Sec. 3. In this situation, it is desirable to choose more discriminative descriptors to generate the tree with a limited number of M.

To achieve this, we efficiently utilize the hierarchical structure of the multi-view features. We structure the features from top to down and use the most reliable descriptors to construct the vocabulary trees. On one hand, the descriptors on the top levels are more robust for matching against the query features. On the other hand, this limits the number of descriptors associated with each leaf node and reduces the risk of outliers.

2.3. Object-Based TF-IDF Weighting

We use the concept of *term frequency-inverse document frequency (tf-idf)* to assign the weighting to each leaf node [11]. Different from conventional tf-idf weighting based on images, our weighting is based on the objects associated with the multi-view images.

For the i-th leaf node, let the number of objects associated with leaf node i be denoted as N_i. For the features of the k-th object associated with this node, the tf-idf weighting factor is given by

$$w_{i,k} = \frac{n_{i,k}}{n_k} \log \frac{N}{N_i}, \tag{1}$$

where $n_{i,k}$ is the number of multi-view features of the k-th object associated with the leaf node i, n_k the total number of multi-view feature of the k-th object and N the number of objects in the whole database. The part $n_{i,k}/n_k$ describes the term frequency which indicates the importance of the associated feature in the k-th object. The part $\log \frac{N}{N_i}$ describes the inverse document frequency which penalizes features that appear often among leaf nodes in the database.

3. 3D GEOMETRY-BASED OBJECT RANKING

3.1. Cost Function

Vocabulary tree matching and geometric consistency verification are two important steps for a successful query. For vocabulary tree matching, a short list of candidate objects is generated by ranking the tf-idf score of each object. For geometric consistency verification, we evaluate the geometric consistency of the candidate objects which we have obtained from the vocabulary tree matching.

Conventional methods usually apply these two steps sequentially on the input query features. This means that the procedure of geometric verification is applied after vocabulary tree matching. Therefore, we develop a new method which combines both steps and outputs an object ranking that considers geometric information.

We consider a constrained problem. If candidate objects have very similar tf-idf scores, then the ranking of these candidate objects is determined by geometric consistency verification. The constrained optimization problem reads

$$\begin{aligned} &\min_k \quad J_k \\ &s.t. \quad |s_k - s_j| \leq \delta, \quad k \neq j, \end{aligned} \tag{2}$$

where J_k is the cost of geometric inconsistency, s_k the tf-idf scoring function of the k-th object, and $\Omega = \{k | |s_k - s_j| \leq \delta, k \neq j\}$ is a set of object indexes associated with objects on a similar score level as defined by the small threshold δ. By solving this problem, the ranking of a set of objects is determined by sorting according to the geometric inconsistency.

3.1.1. Scoring Function

We use our multi-view vocabulary tree to serve each incoming query feature. The best-bin-first strategy is used recursively, it compares the query only to the children nodes of the best parent node [12]. With this approach, each incoming query feature can reach the expected leaf node efficiently.

For each candidate object in the database, we can obtain a set of leaf nodes which the query features have reached. Let this set for the k-th object be denoted as $\mathcal{I}_k$. Conventional methods define the score of the k-th object as the sum of the weighting factors of the leaf nodes in $\mathcal{I}_k$

$$s_k = \sum_{i \in \mathcal{I}_k} w_{i,k}, \tag{3}$$

where $w_{i,k}$ is the tf-idf weighting as shown in (1).

However, it ignores the distance between query and matched leaf node which reflects the confidence of the query matching. Therefore, we introduce a credibility value to each correspondence between query and leaf node. It measures the distance ratio between

first and second closest leaf node for each incoming query feature

$$c_{i,k} = 1 - d_l(1)/d_l(2),\ i \in \mathcal{I}_k, \tag{4}$$

where $d_l(1)$ is the Euclidean distance between query descriptor and centroid of the closet leaf node and $d_l(2)$ that of the second closest leaf node. The credibility value reflects the confidence of the query matching. It is close to 1 when the query feature can be clearly distinguished between first and second closest leaf nodes. It is close to 0 when the query feature lies close to the border of two leaf nodes.

Moreover, the performance of classification depends on the size of the tree as well as the selection of the scoring function. Using the sum of the weighting factors as a scoring function is suitable for scenarios with small vocabulary trees. In such scenarios, the number of descriptors associated with each leaf node is usually large. Then, the td-idf weighting in (1) can discriminate the importance of different features. However, for large vocabulary trees, the number of descriptors associated with each leaf node is usually small. This leads to td-idf weighting that is less discriminative. Thus, a counter $|\mathcal{I}_k|$ which indicates the number of query descriptors that are matched with the k-th object in the database will be helpful for a scoring function.

Considering above scenarios, we define the score for the k-th object candidate as a product of the counter with the weighted sum of the td-idf values

$$s_k = |\mathcal{I}_k| \sum_{i \in \mathcal{I}_k} w_{i,k} c_{i,k}^2. \tag{5}$$

Empirical results suggest to use the square of the credibility values for weighting. Due to above counter, the scoring function grows incrementally for all candidate objects with the incoming query features. Thus, an iterative algorithm can be implemented by monitoring the constrains on the scoring function.

3.1.2. 3D Geometric Consistency

Vocabulary tree matching provides us a set of feature correspondences Q_k between the query and database features of the k-th candidate object. However, purely vocabulary tree matching contains outliers in Q_k, which influence geometric consistency verification significantly. The first step is to choose a subset of reliable correspondences from Q_k, i.e., $\tilde{Q}_k \subset Q_k$. With this step, we can avoid using the outliers in Q_k as well as limit the computational complexity. To do this, we borrow the idea from [7]. We sort the correspondences by counting the number of descriptors under the associated leaf-nodes. Usually, the leaf node which contains less descriptors is more unique and reliable than that with more descriptors. Therefore, we can generate a set of reliable correspondences $\tilde{Q}_k$ where $|\tilde{Q}_k| \leq |Q_k|$. Note, the computational complexity of geometric verification depends only on $|\tilde{Q}_k|$.

Each feature correspondence in $\tilde{Q}_k$ contains a query feature q and a database feature p. Combining the location data of the query features and the intrinsic camera information, we obtain the set of 3D world coordinates W^c by using the method of two-view self-calibration. Similar, we are able to obtain the set of 3D world coordinates W^s for the database features.

For correct feature matching in $\tilde{Q}_k$, we use the seven-parameter Helmert transformation [13] to describe the relationship between the 3D world coordinates of query and database object points

$$\vec{w^c} = \kappa\, \Phi\, \vec{w^s} + \vec{t}, \tag{6}$$

with $\vec{w^c} \in W^c$ and $\vec{w^s} \in W^s$, where κ is the scale parameter in $\mathbb{R}^+$, Φ the rotation matrix in $\mathbb{R}^3$ and $\vec{t}$ the translation parameter in $\mathbb{R}^3$. In our earlier work [3], we use the Helmert-constrained RANSAC to estimate the parameters. However, it results in high computational complexity due to the RANSAC iterations.

Hence, we are not going to estimate the parameters in (6). It is sufficient that we only calculate the 3D misalignment between 3D world coordinates of each query and database correspondence

$$g(q,p) = \log_2 \left[1 + \|\vec{w^c}(q) - \vec{w^s}(p)\|\right], \tag{7}$$

where g is the function of calculating the 3D misalignment. The $\log_2$ operation allows us to limit the dynamic range of the misalignment. Note that the 3D misalignment only depends on the transformation between two coordinate systems (i.e., query and database) and is invariant to the absolute position of individual 3D points. It remains a constant value for all correct matches.

Thus, we define the cost of 3D geometric inconsistency in (2) by the variance of the 3D misalignment

$$J_k = \mathrm{var}\,\{g(q,p)\}, \quad q,p \in \tilde{Q}_k. \tag{8}$$

For a set of correspondences which contains consistent matches, the variance J is usually small. Therefore, the object candidates can be ranked by sorting the variances of the 3D misalignment.

3.2. Iterative Ranking Algorithm

We exploit the incremental properties of the scoring function and develop an iterative algorithm for 3D geometry-based ranking. For each incoming query feature, the scoring function of the candidate objects is updated according to (5). Thus, the scores of the candidate objects are growing incrementally with the incoming query features.

Once two or more candidates reach a similar score level, we are able to define a set of object indexes Ω which contains indexes of objects with similar scores. Then, we evaluate the geometric consistency by using the method as proposed in Sec. 3.1.2. The objects in set Ω are re-ranked by the geometric consistency. A description of the algorithm is shown in Fig. 2.

1. Initialize: Set the score $s_k = 0,\ k = 1 \ldots N$ for all objects and the set of object indexes $\Omega = \emptyset$;
do Update the scores by matching the incoming query features against the vocabulary tree;
a) Update Ω for which $|s_k - s_j| \leq \delta,\ k \neq j$;
b) Calculate the cost of 3D misalignment in (8) for all objects in Ω;
c) Update the ranking of objects in Ω by using above costs;
until All incoming query features have been used.
2. Output the results of object ranking.

Fig. 2. Algorithm for 3D geometry-based ranking.

With this algorithm, we are able to re-rank groups of objects when they satisfy the constraint of similar score level. Due to the vocabulary tree, we can update the scores easily. This approach allows us 3D geometry-based ranking. The top-ranked object is on the highest score level and has the best 3D geometric consistency.

4. EXPERIMENTAL RESULTS

We evaluate our 3D geometry-based ranking for the multi-view image dataset *Stockholm Buildings*[1] which comprises 50 buildings of that city. The server holds 254 images of the 50 buildings. At least 2 views have been recorded for each building. The client may use up to 100 additional test images of the 50 buildings. We acquired

[1] `http://people.kth.se/~haopeng/sthlmbuildings/`

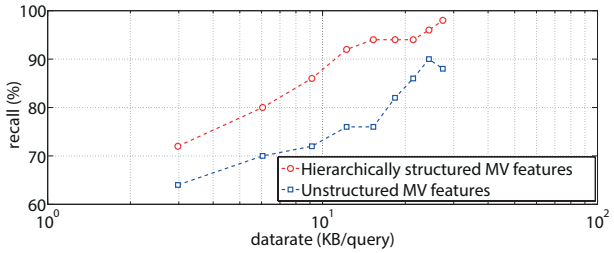


Fig. 3. Comparison of the recall-datarate using hierarchically structured and unstructured multi-view features.

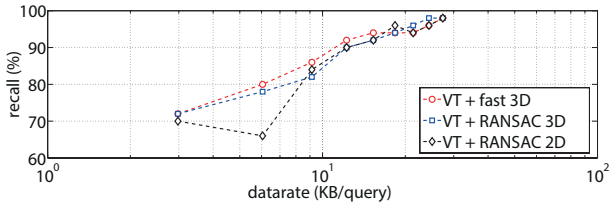


Fig. 4. Comparison of the recall-datarate using different geometric verification methods.

server and test images at different viewpoints and at different times. The images have been recorded by a Cannon IXUS50 digital camera at a resolution of 2592×1944 pixels.

The query features are selected and encoded with the rate-constrained feature selection method from our earlier work [3]. It utilizes stereo features to obtain more reliable query features. Note, this rate-constrained feature selection differs from single-view feature-based methods as discussed in prior frameworks [1]. The advantage of stereo features is explained in [3].

4.1. Structuring of Multi-View Features for Memory-Constrained Multi-View Vocabulary Trees

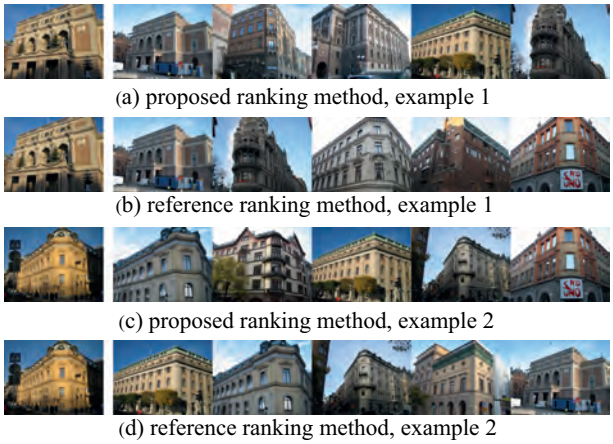
We use two approaches to construct vocabulary trees and compare their performance. The first approach uses all available multi-view features of the data set in an unstructured way. The second approach uses the top of the hierarchically structured multi-view features. For a fair comparison, we use the same memory constraint for the vocabulary trees. Both trees have the same number of tree levels and branches.

We set $D = 5$ for the number of tree levels and $K = 8$ for the branches. The total memory usage of such vocabulary trees is about 70 MB. Note that the original view by view feature database uses 5.3 GB, and the multi-view feature database uses 400 MB. For the hierarchically structured multi-view features, the top three levels have been used on average for the construction of the vocabulary tree. We use our iterative algorithm to obtain the recall-datarate trade-off. As shown in Fig. 3, using the hierarchically structured multi-view features improves the performance significantly. As expected, this approach selects the most reliable features to construct the vocabulary tree as well as limits the number of outliers at each leaf node.

4.2. Comparison of Different Geometric Verification Methods

We evaluate the performance of fast 3D geometric verification by comparing it to other two RANSAC based methods, namely the epipolar-constrained 2D RANSAC and the Helmert-constrained 3D RANSAC [3]. As shown in Fig. 4, we investigate the trade-off between recall and datarate for these methods. Note that we use the same vocabulary tree for all methods. In general, the methods based on 3D geometry information outperform the 2D RANSAC as the underlying 3D geometry is more discriminative than the 2D geometry.

We also investigate the average execution time by using different geometric verification methods on a MATLAB platform. To obtain the top five ranked images, the fast 3D geometric verification method needs only 0.16 seconds to achieve an average recall of 90%. The 2D RANSAC algorithm needs 3 seconds, and the 3D RANSAC algorithm needs 13 seconds to achieve the same recall level.



(a) proposed ranking method, example 1

(b) reference ranking method, example 1

(c) proposed ranking method, example 2

(d) reference ranking method, example 2

Fig. 5. Comparison of the ranking results using proposed and reference methods.

4.3. Performance of Ranking

We evaluate the performance of ranking by comparing our method to a reference method. The reference method uses the scoring function in (3) and the epipolar-constrained 2D RANSAC for the geometric verification after vocabulary tree matching. Note that we use the same multi-view vocabulary tree for both methods.

As shown in Fig. 5, the left most image is the query image from the client side while the other five images in the same row are the ranking results. With the improved scoring function, the proposed method can efficiently retrieve objects with similar visual elements such as window patterns and eaves. On the other hand, the fast 3D geometric verification improves the ranking by considering the underlying 3D geometry of the objects such as round corners, sharp edges, or planar surfaces.

5. CONCLUSIONS

We discussed mobile 3D visual search and proposed an algorithm for 3D geometry-based ranking. We use multi-view vocabulary trees and construct them by utilizing hierarchically structured multi-view features. Object-based tf-idf weighting and scoring functions have been designed for the mobile 3D visual search scenario. With 3D geometric information associated with the vocabulary tree, we developed an iterative algorithm for 3D geometry-based ranking. The experimental results show that our tree design as well as 3D geometry-based ranking improve the recall-datarate performance significantly.

6. ACKNOWLEDGMENTS

This work has been supported in part by EIT ICT Labs in the context of the project 14452 “Mobile 3D Visual Search in 3D Environments”.

7. REFERENCES

[1] B. Girod, V. Chandrasekhar, R. Grzeszczuk, and Y. Reznik, "Mobile visual search: Architectures, technologies, and the emerging MPEG standard," *IEEE Trans. on Multimedia*, vol. 18, no. 3, pp. 86 –94, Mar. 2011.

[2] D. Chen, G. Baatz, K. Koser, S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, C. Xin, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk, "City-scale landmark identification on mobile devices," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2011.

[3] H. Li and M. Flierl, "Mobile 3D visual search using the Helmert transformation of stereo features," in *Proc. of the IEEE International Conference on Image Processing*, Sept. 2013.

[4] X. Lyu, H. Li, and M. Flierl, "Hierarchically structured multiview features for mobile visual search," in *Proc. of the IEEE Data Compression Conference*, Mar. 2014.

[5] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, J. Singh, and B. Girod, "Robust image retrieval using multiview scalable vocabulary trees," in *Proc. SPIE, Visual Communications and Image Processing*, Jan. 2009.

[6] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[7] S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, R. Vedantham, R. Grzeszczuk, and B. Girod, "Fast geometric re-ranking for image-based retrieval," in *Proc. of the IEEE International Conference on Image Processing*, 2010.

[8] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2006.

[9] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod, "Inverted index compression for scalable image matching," in *Proc. of the IEEE Data Compression Conference*, Mar. 2010.

[10] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 881 –892, July 2002.

[11] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. of the International Conference on Computer Vision*, Oct. 2003.

[12] J. Beis and D. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[13] G. Watson, "Computing Helmert transformations," *Journal of Computational and Applied Mathematics*, vol. 197, no. 2, pp. 387 –394, 2006.