IMAGE LOCALIZATION IN SATELLITE IMAGERY WITH FEATURE-BASED INDEXING

Changchang Wu[1],Friedrich Fraundorfer[2], Jan-Michael Frahm[1], Jack Snoeyink[1] and Marc Pollefeys[1,2]

[1]Department of Computer Science
The University of North Carolina at Chapel Hill, NC, USA
{ccwu,jmf,snoeyink}@cs.unc.edu

[2]Department of Computer Science
ETH Zurich, Switzerland
{fraundorfer, marc.pollefeys}@inf.ethz.ch

Commission III/4

KEY WORDS: orthogonal satellite imagery, image localization, content-based image retrieval, image feature, SIFT, visual word, vocabulary tree, map database indexing

ABSTRACT:

This paper presents a method to index ortho-map databases with image-based features and search a map database for regions that match query images of unknown scales and rotations. The proposed method uses image-based features to index the 2D map locations. Image feature extractors normally generate features with location, orientation, shape, and a descriptor for normalized image patch. In a map database, the geographical location, orientation and shape of the image features can be recovered with a reasonable local planarity assumptions. The paper makes use of a visual word based recognition scheme and extends it by adding geographical dimensions to the visual words and use them to index 2D locations in a map grid.An indexing-friendly scoring system is defined to measure the similarity of query and database images which represent unit tiles of the complete map. The implemented scoring algorithm can efficiently give the matching scores between a query image and all possible database images. Upon searching a new approximately orthogonal image, a set of scaling and rotations are first selected, and the visual words are transformed and matched against the database. The best locations along with scales and rotations are determined from the query results of the different set of transformed visual words. Experiments show a high success rate and high speed in searching map databases for aerial images from different datasets.

1 INTRODUCTION

Nowadays, satellite imagery has become an important part of our information source. The amount of high resolution satellite imagery is growing rapidly, and much of it is now available to the public through various map services, such as Google Maps, etc. In this paper, we are interested in the problem of matching aerial images (with unknown scales and rotations) to a map database. Given a particular aerial image, we propose a method to find the locations of similar map data along with the relative scales and rotations, and provide a confidence measurement for the similarity. Temporal changes, repetitive structures, varying illumination conditions, and varying cameras lead to appearance variances that make the problem very difficult to resolve. Our approach efficiently handles the challenges that come with the complexity and large scale of satellite imagery.

One of the main contributions of this paper is to propose a new feature indexing for geo-located features in a map. Our method adds geometric dimensions to existing visual word (Sivic and Zisserman, 2003, Nistér and Stewénius, 2006), and uses the extended visual words on map to index 2D location grids. Unlike the general image retrieval problem, geographical size and geographical rotation of features in map database can be recovered. Visual words with sizes and rotations differentiate features at different scales and different orientations, which leads to a more efficient indexing and retrieval system.

A significant difference, between a map database and an image database in normal image retrieval problem, is that unit tile images in a map database are fully geographically connected, while normal image database contains independent images. Satellite imagery can be viewed as a grid of unit tile images of a same size and the neighboring tile images in a same map database can not be treated independently. Correspondingly, query images of different sizes need to take varying number of unit tile images as a group to match. This paper introduces a scoring scheme that can be applied to matching with such image groups.

The remainder of the paper is organized as follows: Related work is discussed in Section 2. Afterwards Section 3 introduces our new visual word and the associated feature indexing system. An efficient image localization based on the indexing system is given in Section 4. Experimental results are presented in Section 5. Conclusions and future work are discussed in Section 6.

2 RELATED WORK

Our localization task is a problem of looking for small parts out of a much larger image, which is essentially a content-based image retrieval problem. This section will discuss some related techniques in image retrieval including invariant image features, visual word indexing and vocabulary tree method.

2.1 Invariant Image Feature

In recent years, there has been an extensive investigation in local invariant image features to achieve robustness to viewpoint changes. Lowe's Scale Invariant Feature Transform (SIFT) extracts distinctive scale and rotation invariant features from the DOG (difference of Gaussian) scale space of images, and describes the corresponding normalized image patches with SIFT descriptors, which are 128D vectors constructed from the local gradient histograms (Lowe, 2004). Figure 2 shows an example image with SIFT features. While SIFT detector handles only 2D similarity transformation, some other feature detectors (e.g. MSER) go beyond to achieve invariance to affine changes. A good overview and evaluation of such affine invariant features can be found in (Mikolajczyk et al., 2005). SIFT descriptor, as a

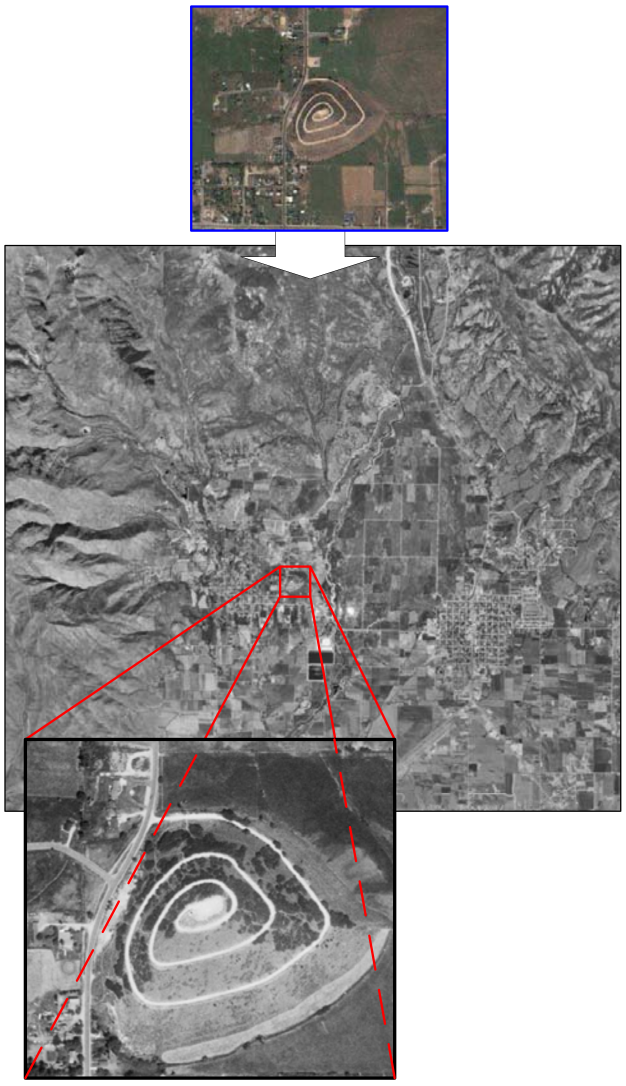

Figure 1: Illustration of the image localization problem. The background is a zoomed view of part of the satellite map in Park City, UT, USA, and the smaller image on the left bottom and the small red box in the map gives a detailed view of a small area. The color image at the top is a query example from a different dataset taken about 6 years later.

general method to describe normalize image patch, is also used in those features. Those local invariant features also partially handle illumination changes by using normalized SIFT descriptors.

Methods like nearest neighbor matching of SIFT descriptors can then be used to establish the feature point matches between images. Figure 3 shows an example of SIFT feature matching. Feature point matches are also used in sparse 3D reconstructions (Snavely et al., 2006). In this paper, we also use the local invariant features to build the image-based localization system for map databases.

2.2 Image Search with Visual Words

We will give a short outline of the visual word based image search scheme that is described in (Sivic and Zisserman, 2003, Nistér and Stewénius, 2006, Fraundorfer et al., 2007) to introduce the terminology, which is used in this paper. The image search proceeds by a local approach, by detecting local features, computing a description (feature) vector and matching with a database of

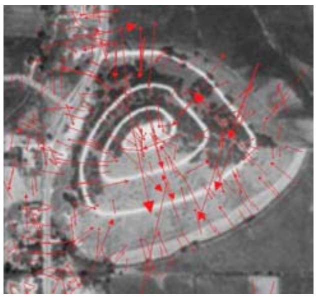

Figure 2: SIFT features shown as arrows. Location, size and orientation of features are corresponding to the starting point, length, and direction of the arrows.

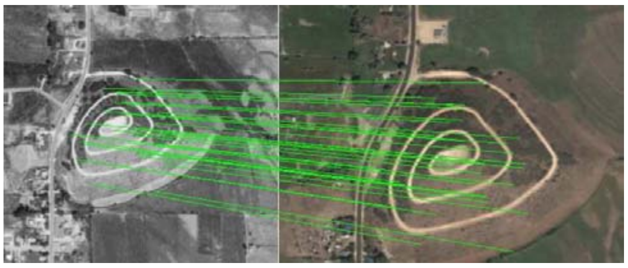

Figure 3: SIFT matches shown as lines connecting the corresponding feature points in two images.

feature vectors (see illustration in Figure 4). Each local detection is described by a SIFT feature vector. Each SIFT feature vector is then quantized with the so-called vocabulary tree. It assigns a single integer value, denoted visual word (VW) to a SIFT feature vector. This eases matching a lot. Instead of computing distances between SIFT feature vectors only integer values have to be compared. Each image is then represented as a set of visual words. The set is denoted as a document vector which is a v-dimensional vector where v is the number of possible visual words. It is usually extremely sparse which creates a very compact image representation leading to a highly scalable and efficient approach. For an image query the similarity between the query document vector to all document vectors in a database is computed. As similarity score the distance (e.g. L1, L2) between document vectors can be used. The vector with the lowest distance is reported as the most similar match.

The organization of the database as an inverted file and the sparseness of the document vectors allows very efficient scoring. The inverted file is not storing the VW's itself, instead it stores an image identifier for the image in which the VW was detected. This creates a list of image identifiers for each individual VW. When using an inverted file the computation of similarity scores between the query document vector and the database document vectors reduces to simple voting. For each VW that occurs in the query document vector the corresponding inverted file list is processed and a vote is cast for the corresponding images. By selecting proper values for the votes the distance can be computed (Nistér and Stewénius, 2006). In most cases TF-IDF weighting is used in the scoring, so that rare VW's count more than frequently occurring VW's. The scoring process is illustrated in Figure 5.

A crucial step is the quantization of the SIFT feature vectors into visual words using the vocabulary tree. The vocabulary tree defines a partitioning of the 128-dimensional SIFT feature space. The partitioning is computed off-line by hierarchical k-means clustering of a large amount of training SIFT feature vectors which creates a tree of cluster centers. For an image query SIFT feature quantization works by determining in which cluster cell the feature falls. The corresponding visual word is simply the cell identifier (a simple integer value). The method used in this paper is an extension of the just described vocabulary tree method.

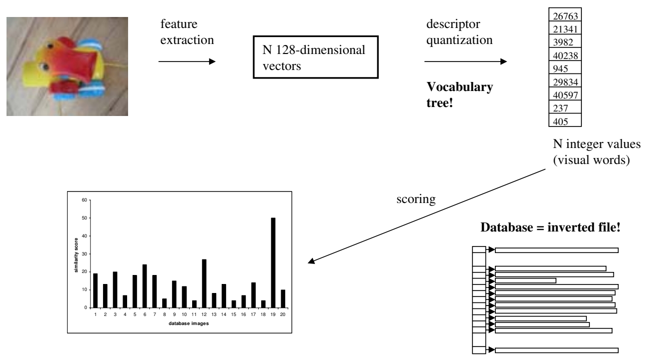

Figure 4: Illustration of the vocabulary tree based image search scheme: It proceeds by a local approach, by detecting local features, quantizing the SIFT descriptors with a vocabulary tree into visual words and computing the similarity between the query image and the database images (using an inverted file structure). The database image with highest similarity is reported as match.

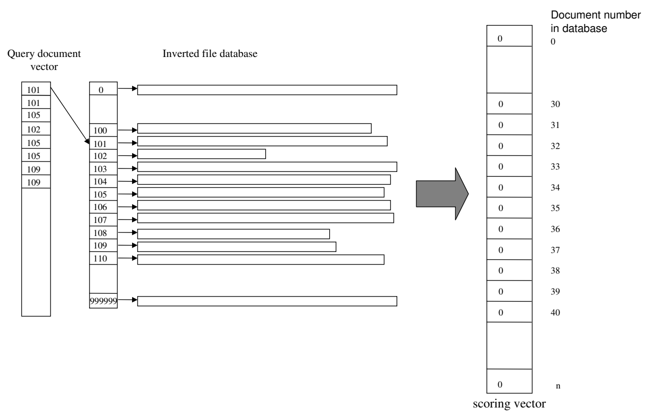

Figure 5: Scoring with an inverted file: For each VW that occurs in the query document vector the corresponding inverted file list is processed and a vote is cast for the corresponding images in the scoring vector. Proper vote values ensure that the scoring vector contains the used norm distances (e.g. L1, L2) between the query and database vectors.

3 MAP DATABASE INDEXING

This section discusses the proposed new visual word for indexing the map databases, and explains the weighting of the new visual words.

3.1 The Extended Visual Word

A good indexing of features needs a vocabulary of discriminative visual words. Although quantization through vocabulary tree makes SIFT descriptors easily indexable, pure descriptor-based indexing still results in a lot of ambiguity. The reason for this is that the feature descriptor itself does not carry much information about the size, orientation and shape of the real 3D geometry. Invariant feature detectors can transform different image patches of the same 3D structure to a similar normalized patch. On the other hand, it may also generate similar normalized patches for different 3D surfaces. Accordingly, the same visual word is very likely to represent a set of features from different 3D structures with different projective transformations.

In the context of satellite imagery, local 3D structure corresponding to image features can be approximately recovered by assuming the corresponding part of the ground to be planar. This assumption is reasonable because the variation of elevation is much smaller than the distance between the earth surface and the camera. Then every image feature can be seen as a geographic image feature. The additional information about the geographical size and orientation are discriminative properties of image features, which can resolve the ambiguity between many different 3D structures on the ground. Therefore, the geographical size and orientation can be used along with the descriptor for indexing. Additionally the geographic properties of affine covariant regions can also be recovered. This paper only conducts experiments on SIFT features, but the proposed method can be extended to use these affine invariant features.

Satellite imagery can be ortho-rectified with an affine transformation because the cameras are very far away from the ground. Then, the only remaining unknown image transform between rectified images taken by different camera or camera at different time is approximately a 2D similarity transformation. To cope with the remaining similarity transformation between query image and database images we chose the similarity invariant SIFT features.

The visual words stored in our database are triplets consisting of the quantized SIFT descriptor, the geographical size, and the geographical orientation. Feature descriptors are quantized with a vocabulary tree that is trained from millions of features extracted from the satellite map. Then the geographical orientation and the logarithm of geographical size are also discretized to integers for convenient indexing. Hence, each visual word is a 3D integer vector. Like other indexing-based techniques, inverted files are constructed for the visual words of each map database.

In detail, for a feature $(x,\ y,\ \zeta,\ \theta,\ sift)$, where $(x,\ y)$ denotes its UTM coordinate in meters, ζ the geographical size in meters, θ the angle within the ground plane in degrees, and $sift$ the SIFT descriptor, it is transformed to an indexing pointer as below:

$$(\mathrm{f_s}\lfloor \log_2 \zeta \rfloor,\ \lfloor \theta * N_\theta/360 \rfloor,\ \mathrm{f_v}(sift)) \Rightarrow (\lfloor x/W_t \rfloor,\ \lfloor y/H_t \rfloor)$$

where function $\mathrm{f_s}$ maps the logarithm of feature size to a smaller set of N_s integers, N_θ is the number of rotation to use, function $\mathrm{f_v}$ uses a pre-trained vocabulary tree to quantize 128D SIFT descriptors to integers, $W_t \times H_t$ is the tile size for the database. $(\mathrm{f_s}\lfloor \log_2 \zeta \rfloor,\ \lfloor \theta * N_\theta/360 \rfloor,\ \mathrm{f_v}(sift))$ on the left is the visual word from the feature, and on the right side$(\lfloor x/W_t \rfloor,\ \lfloor y/H_t \rfloor)$ is a location (visual document of a $W_t \times H_t$ tile) to index. Our proposed method is indexing 2D connected location grid other than independent images in other image retrieval problems. The map database is stored on disk as tile images with associated features corresponding each indexed locations. A tile size of 200×200 pixel is chosen in our experiment because the original image data we downloaded from TerraUSA (Microsoft TerraServer, n.d.) server are organized as 200×200 pixel tiles.

An advantage of this new visual word is that scaling transformation and rotation transformation can be applied to the visual words in the vocabulary, and the transformed visual words will still be in the vocabulary. Similarly, scaling and rotation can be applied to any query image to get a new group of visual words for querying. For convenience, given a function g of some scaling plus some rotation, the transformation of a visual word t is denoted as $g(t)$, and the transformation of a visual word group $\{t\}$ as $\{g(t)\}$.

Given some scaling and rotation threshold, the set of visual words that have similar scale, similar rotation and the same descriptor with a visual word is defined. Then the inverted files of those similar visual words can also be used in matching. This enables us to match query images to the image database with some transformation threshold. For convenience, given a threshold T of some scaling and some rotation, the set of visual words within such a range of a visual word t is called $T(t)$.

3.2 Visual Word Weighting

It is important to weight the visual words in the vocabulary so that visual words are treated differently according to their frequency. For example, if some visual words show up in all most all the locations, a small number of occurrence of such feature won't provide sufficient information for localization, which means a small weight should be assigned. A standard IDF (inverse document frequency) weighting is used in this paper to weight visual words. Such an IDF weight for a visual word t_i is defined

$$w_i = \text{idf}_i = \log \frac{|XY|}{|xy : t_i \in d_{xy}|}$$

where $|XY|$ is the total number of locations to index and $|xy : t_i \in d_{xy}|$ is the number of locations n which this visual word occurs. IDF is set to zero when the number of occurrence is zero. IDF weighting of the new visual words now gives different weights to features that have different scales and orientations. A a result, the large features now become more important than the smaller ones since there are less of them. This is also consistent with our intuition about the features. Without the utilization of the geographical information, the IDF weighing will lose a powerful dimension for discriminating features. However, due to the dependency on the real 3D structures, the proposed visual word will not be applicable to general image retrieval problem except for the cases where all 3D dense structure and relative sizes can be recovered.

Figure 6 demonstrates the change of IDF distribution when feature sizes are used in the location indexing. In this proposed scheme, more visual words in the vocabulary have zero occurrences, which is more efficient for lookup of inverted files because more lookups are skipped. The proposed visual words also emphasize the importance of large feature sizes.

4 MAP DATABASE SEARCH

This section will explain the image localization algorithm with the proposed feature-based indexing. The definition of our scoring system and the associated implementation is introduced, which are the main contributions of the paper. Specifically handling of querying sets of database tiles is discussed afterwards. The querying algorithm for recovering locations along with scales and rotations is given in the end.

An image with known geographical size and rotation or an hypothesis of the geographical size and rotation can be seen as a vector d where d_i gives the number of occurrence of visual world i. The correlation between any two images is defined as

$$\text{corr}(q,\ d) \quad = \quad \sum_i q_i d_i w_i^2$$

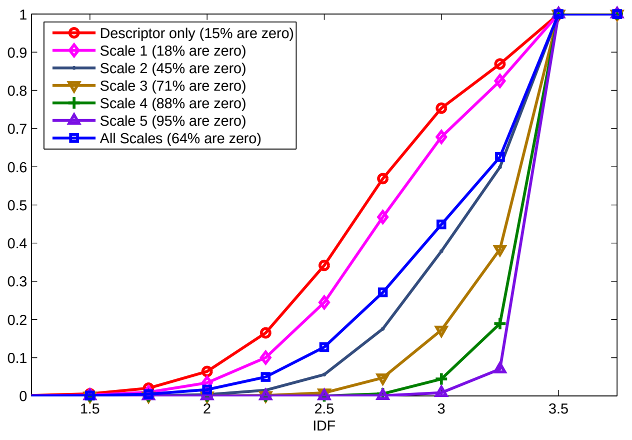

Figure 6: CDF plot of non-zero IDF for a database. In this experiment, 5 scale levels are used, rotation is ignored. The first red curve is the IDF for the descriptor-only visual word. The rest are the curves for the proposed visual words at five different scales and at all scales. With scales taken into account, much larger portion of visual words have zero occurrence in the database. visual words of larger scales will have more zeros occurrences and larger IDF weights because of less occurrence.

The above definition leads to an easy implementation that does not require the counting of visual word occurrences as follows:

```
function ComputeCorrelation(q)
1 set $\text{corr}(*,\ q)$ to zero
2 for each $i$ in image $q$
3   for each $d$ in the inverted file of $i$
4     $\text{corr}(d,\ q)\ +=\ w_i^2$
```

When some threshold is provided, a set of visual words need to be checked for each visual word in the query document. The computation will be as follows:

```
function ComputeCorrelationWithThreshold(q, T)
1 set $\text{corr}(*,\ q)$ to zero
2 for each $i$ in image $q$
3   for each $j \in T(i)$
4     for each $d$ in the inverted file of $j$
5       $\text{corr}(d,\ q)\ +=\ w_j^2$
```

The final normalized matching score between any two sets of visual words is defined as

$$\text{score}(q,\ d) = \frac{\text{corr}(q,\ d)}{\sqrt{\text{corr}(q,\ q)\text{corr}(d,\ d)}}$$

which measures the confidence of two images being the same scene.

When querying an image q, its matching score $\text{score}(q,\ d)$ with all d in the database are computed as above, then the best matches can be obtained by comparing the scores to determine the largest ones. To realize efficient query, the self-correlation of database image $\text{corr}(d,\ d)$ is should be pre-computed when building the database. $\text{corr}(q,\ q)$, the self-correlation of query image, only need to be computed once for each possible transformation on a query image, and it is constant for all database documents.

Our problem is to locate images with unknown scale and rotation in a map database. It needs to compute not only the best possible locations, but also the corresponding scales and rotations. Although the actual stored images in our map database are 200×200 pixel tiles, it is not enough only computing the matching score between the query image and each database tile, because the query image could match to a region of multiple tiles. We need to match query images with any map images with some reasonable size of $W_t \times H_t$ tiles, where W_t and H_t are the number of tiles along x and y direction. An image made of any $W_t \times H_t$ tiles is also called a tile group in this paper.

The correlation between a query image q and a tile group D can be obtained from the query's correlation with the tile units as

$$\mathrm{corr}(q, D) = \sum_{d \in D} corr(q, d)$$

Once computing the correlation of a query with all the database tiles, the correlation between the query and any tile groups can obtained by summing up the correlation of all the tiles in the group.

The self-correlation of the tile groups in the map database needs to be specifically handled. For any group of tiles $D = \{d\}$, its self-correlation is

$$\begin{aligned}\mathrm{corr}(D,\ D) &= \sum_i D_i D_i w_i^2 \\ &= \sum_i w_i^2 (\sum_{d \in D} d_i)^2 \\ &= \sum_{d1 \in D} \sum_{d2 \in D} \mathrm{corr}(d1, d2)\end{aligned}$$

The self-correlation of a tile group is the sum of the correlations of any two tiles in the tile group. Therefore, only the correlations between any two tiles need to be pre-computed for fast computation of self-correlations, and the self-correlation of the tile groups can then be easily obtained.

To recover the scale and rotation between query and database, we first choose a set of reasonable scaling and rotation transformations as hypotheses. A set of thresholds are also selected accordingly to cover the gaps between them. Each transformation then generates a new set of visual words to match with the database. The corresponding map sizes for a query image under each transformation is determined according to the scale, and the matching scores between the query image and all possible tile groups are then computed. Finally, by finding the maximum matching scores among all considered transformations, scale, and rotation are recovered along with the location. While choosing the top N scores, to avoid choosing locations that are too close to each other, each time when a maximum score is selected, its neighboring locations will be suppressed as non-maximum.

```
function Query($q$)
1 Get the set of geometric transformation $G$ to test
2 for each $g \in G$
3   Compute the tile group size $W_g \times H_g$
4   Compute correlation of g(q) with database tiles
5   Compute the summed correlation for tile groups
6   Compute the self correlation for tile groups
7   Compute the matching scores
8 Select tile groups with the $N_q$ largest scores
9 Verify 2D transformation for selected tile groups
```

The final optional step of matching is the geometric verification of some top scoring images as tile groups. Putative feature matches can be established form the inverted files of visual words. A simple histogram method is then used to get the inliers for 2D similarity transformation. Since each feature contains information about scale, rotation and translation, each putative match can establish a 2D similarity transformation. A histogram of number of supporting matches can be constructed, and the largest number corresponds to the best possible geometric transformation for this image. Then the best matches among all candidate images are the ones with largest number of inliers. Geometric verification not only filters out false positives, but also recovers more accurate scale, rotation, and location.

5 EXPERIMENTS

This section first explains how the feature extraction is adapted for map database, then talks about the database and query images that are used in the experiments. Experiment with querying images that are from different datasets with the map database are presented.

5.1 The Map Database

To extract SIFT features for satellite map database, the huge map image need to be divided into small pieces on which feature detection is run. As it is not feasible to detect features for the entire map at once. Enough overlap between sub-images is very important for keeping the features that are close the boundary. Otherwise, features of large scale on the boundary will be lost. In our experiments, an 800 pixel overlap is used. It means the feature detection can approximately keep all features of sizes up to 800 pixels. Additionally, GPU-based SIFT implementation is used to speed up the processing (Sinha et al., 2006, Wu, 2007). The fast processing with GPU is also another reason that we choose SIFT.

In our experiment, a gray-level satellite map of 16000 meters by 12000 meters in Park city of Utah is used in our experiments. The background image in Figure 1 is a small part of our database, where both mountains and cities are covered. It contains 4800 200×200 tile images that are downloaded form the Aerial Photo data at (Microsoft TerraServer, n.d.). The downloaded map is dated 1997, and the resolution of the map is 1 pixel per square meter. With the GPU-based SIFT, 831084 features are detected from the entire downloaded map.

After feature extraction, a vocabulary tree is trained off-line using the extracted SIFT descriptors. With this tree and some manually chosen scale set and rotation set, image query tasks can be performed after loading features from disk, converting them to visual words, and adding location pointers to inverted files of visual words. We chose a vocabulary tree with branching factor 10 and 5 levels (i.e. resulting in 100000 leaves) together with 100 different scales and 360 different rotations to establish the vocabulary. A set of geometric transformations of 5 scales and 9 rotations are used in the query. The thresholds for scaling and rotations are selected accordingly to cover the entire geometric transformation space. Our implementation of indexing is using the method in (Fraundorfer et al., 2007).

5.2 The Park City (USGS Seamless) Experiment

Query images were generated by taking screen shots from online databases. We first took a set of 20 screen shots from the DOQQ 1.0m B&W (west) data at (The National Map Seamless Server, U.S. Geological Survey, n.d.). The experiment of searching those images have a 100% success rate at the top match. We believe the DOQQ dataset and the TerraServer Aerial Photo dataset are just rectified differently from the same original data and the query images in this experiment differ from the database images only by a small affine transformation. Experiments prove that the proposed visual word is working properly under such image transformation which is approximately a 2D similarity transformation.

5.3 The Google Maps Experiment

For the second experiment, color images from Google Maps are selected as query images (see Figure 7). 40 images are chosen from Google Maps in the same area. Variances of scales in the query images are maintained for verifying how well the proposed visual word can handle scale changes. More pictures with at least a few roads are chosen on purpose because those regions are close to places where people can reach. Considering that the color image dataset in TerraServer is dated 2003, those query images might be taken no earlier than 2003, which means a 6 year gap from our database which is dated 1997. The regions with too much temporal changes are avoided. Figure 8 demonstrates the experimental results, which shows a recognition rate of 90% for top 6 matches and 40% for the top match is obtained. Note that 6 is only .13% of the entire database. Although this is not a general success rate because the query set is not randomly sampled, the result is still promising because normally those parts with salient features are what we are interested in.

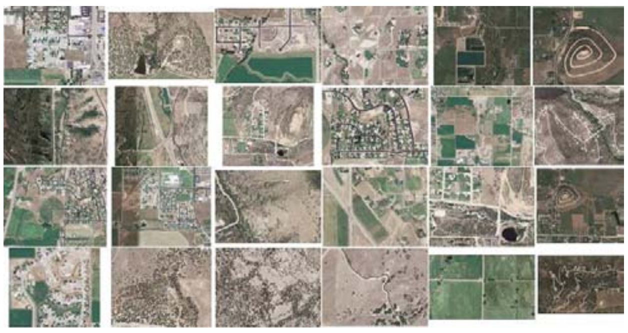
Figure 7: Thumbnails of some Sample query images in our experiments.

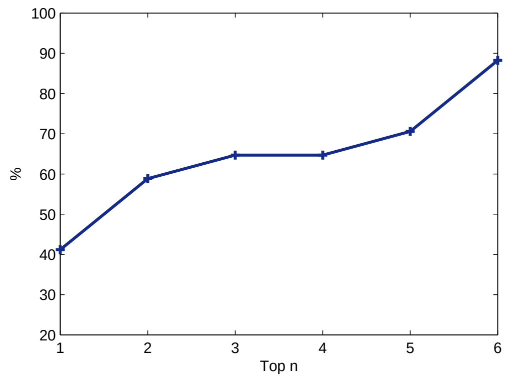

Figure 8: Percentage of ground truth locations that make in to the top X matches without geometric verification. Considering more top matches gives higher probability of having a correct query.

The query speed without geometric verification is 2hz on a machine with 3Ghz CPU and 1GB RAM. Geometric verification can take charge of finding the most likely location together with the scale and rotation from the top matches. This step takes about 1 to 4 seconds to verify the top 10 putative locations depending on the number of features.

6 CONCLUSION AND FUTURE WORK

The paper proposed a new visual word for indexing orthogonal satellite imagery and the associated method for image-based localization. The proposed visual word incorporates the geographic information of image features, and gives stronger discriminability for indexing images. A scoring implementation is designed to match a query image to a part of a large image (represented as multiple tiles), which is significantly different to standard image retrieval. Scale and rotations are recovered together with location by matching the proposed visual words. The proposed method can be used to for information mining from map databases, for example, searching for interesting patterns on a map. The future work of this paper includes extending the work to other image features and experimenting on larger database.

7 ACKNOWLEDGEMENTS

This work has been supported in part by NGA under the Geo* program.

REFERENCES

Fraundorfer, F., Stewénius, H. and Nistér, D., 2007. A binning scheme for fast hard drive based image search. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, Minnesota, pp. 1–6.

Lowe, D., 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), pp. 91–110.

Microsoft TerraServer, n.d. http://terraserver.microsoft.com/.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T. and Van Gool, L., 2005. A comparison of affine region detectors. International Journal of Computer Vision 65(1-2), pp. 43–72.

The National Map Seamless Server, U.S. Geological Survey, n.d. http://seamleass.usgs.gov.

Nistér, D. and Stewénius, H., 2006. Scalable recognition with a vocabulary tree. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, New York City, New York, Vol. 2, pp. 2161–2168.

Sinha, S., Frahm, J.-M., Pollefeys, M. and Genc, Y., 2006. Gpu-based video feature tracking and matching. In: Workshop on Edge Computing Using New Commodity Architectures, Chapel Hill, US.

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

Snavely, N., Seitz, S. M. and Szeliski, R., 2006. Photo tourism: Exploring photo collections in 3d. In: SIGGRAPH Conference Proceedings, ACM Press, New York, NY, USA, pp. 835–846.

Wu, C., 2007. Siftgpu: A gpu implementation of lowe's sift. http://cs.unc.edu/ ccwu/siftgpu.