EFFICIENT MATCHINGS IN AUGMENTED REALITY APPLICATION

Wei Guan        Suya You        Ulrich Neumann
University of Southern California

ABSTRACT
With fast growing popularity of smart phones in recent years, augmented reality (AR) becomes more demanding than ever before. However, one of main challenges is that while features like SIFT or SURF are robust in matchings, they are not computationally efficient. In this paper, we propose an efficient matching method for robust features. A distinctive descriptor is also proposed for performance improvements. Besides, we have developed an outdoor augmented reality system that is based on our proposed methods. The system demonstrates that not only it can achieve robust matchings efficiently, it is also capable to handle large occlusions such as passengers and moving vehicles.

Index Terms— scale space, efficient image matching

1. INTRODUCTION

One of the most challenging problems in the vision community as well as in augmented reality area is to conduct image matchings efficiently with sufficient robustness. While there are many robust features that have good image matching performance, these features are not computationally efficient for realtime applications. Many fast features have been designed in recent years, however, most of these features sacrifice the robustness to meet with the realtime requirements. While the tradeoffs do exist for certain features, there are other ways to improve both of them. In this paper, we will cope with such a challenging problem.

We know that SIFT [1] and SURF [2] are two of the most popular robust features. Both of them first detect keypoints and then calculate the descriptors. Among the two phases, detections consume much more time than forming descriptors. One important reason with the inefficiency of detections is that it conducts scale space analysis and detect keypoints across all the scales. If we can avoid the process of scale space analysis, the performance will be significantly improved. In this work, we are employing the retrieval techniques to achieve such goal. However, a series of other challenges need to be handled to maintain the robustness of the proposed method.

Another requirement of many AR system is the ability to handle occlusions and dynamics. A hierarchical partitioning method is proposed to conquer this problem efficiently.

We talk about the details of our method in the remainder of this paper. After talking about some related work in section 2, we present the way to use retrieval techniques to replace scale space analysis in section 3. We propose a hierarchical partitioning scheme and a distinctive descriptor for both efficiency and robustness in section 4 and section 5 respectively. In section 6, we present an augmented reality system that is based on the proposed method. We show our experimental results in section 7 and conclude the paper in the last section.

2. RELATED WORK

In augmented reality, the fundamental problems are image matching and tracking with specific features. Over the past decade, many algorithms are proposed to handle such problems. Depending on the features they use to describe images and the method they exploit to do the matchings, these algorithms can be divided into two categories. In the first category [3, 4], simple features are used, but sophisticated techniques are required in the matching process. In [3], the vertical lines are extracted from images and combined with distance information which is obtained from ultrasound sensors. A Bayesian network is used for estimation of the robot location. Dodds and Hager [4] use a color interest operator consisting of a weighted combination of heuristic scores to identify landmarks. The operator can select regions that are robust representations for scenes recognition. In the second category, more sophisticated features are used [5, 6]. In [5], Se et al. propose a vision-based simultaneous localization and mapping (SLAM) system by tracking the SIFT features. In [6], Wolf et al. use local scale-invariant features and combine with Monte-Carlo localization to estimate robot positions. The system is robust against occlusions and dynamics such as passengers.

The techniques described above either use robust features with more computational time or sacrifice the robustness to achieve fast speed. Most of the works are not analyzing the bottlenecks in robust features and trying to make improvements on those aspects. While these techniques work effectively for their application uses, they are not suitable or sufficient for our system. In the following sections, we will dis-
cuss about our proposed matching schemes.

3. RETRIEVAL IN SCALE SPACE

Scale space theory is a framework for multi-scale signal representation. It is usually used to handle image structures at different scales by representing an image as a family of image pyramids. However, detecting interest points across scales is the most costly part in this process, so it is computationally inefficient especially for portable devices with less computing powers.

**Fig. 1.** Image retrieval in scale space. The image with closest scale has highest score in the rank (thickest arrow).

The workload can be largely reduced if we put scale space analysis for database images in offline process, and detect image scale of the query image by retrieval techniques (as in Fig. 1). In this way, the whole detection process can be significantly speeded up. Take working on a 1.5GHz processor as an example, the scale-invariant detections on 640 by 480 images usually take about 900ms for SIFT detectors and 300ms for SURF detectors, but single scale detection only takes 90ms and 25ms respectively, and retrieval process costs about 5ms to 10ms. Therefore, due to shift of scale space analysis, we have

\[
T_{\text{one-scale}} + T_{\text{retrieval}} < \leq T_{\text{scale-invariant}}.
\]

where \(T_{\text{one-scale}}\) and \(T_{\text{scale-invariant}}\) are detection time for single scale and multiple scales respectively, and \(T_{\text{retrieval}}\) denotes the time used for retrieval.

To analyze interest points in scale space, Fast-Hessian detector as in SURF is used. Instead of changing box filter size, we keep the filter size constant and apply it to image pyramids for up to 9 scales with step \(\sigma = 1.2\). The image at each scale is built into the database for retrieval. Note that in this case, we are not analyzing the scale for a single feature point, we analyze the scale of feature points as a whole in the image.

4. HIERARCHICAL GRID RETRIEVAL

4.1. Partitioning into Grids

In most cases, it is parts of the query image rather than the whole image that have corresponding matches in the database. To better refine the retrieval results, we divide the query image into \(4 \times 4\) grids. As shown in Fig. 2-(a), there are sixteen \(1 \times 1\) grids, nine \(2 \times 2\) grids, four \(3 \times 3\) grids and one \(4 \times 4\) grid (image itself). So there are totally 16+9+4+1=30 hierarchical grids for one image, and any grid with number of features greater than \(N_0\) will be constructed in the database. This is to make sure that every grid in retrieval database has sufficient features to be identified with less ambiguity. In our experiments, we set \(N_0=100\). For most images, there will be around 50 hierarchical grids across all scales.

**Fig. 2.** An image is hierarchically partitioned into 4 by 4 grids for both query image and database image. Four different sizes are used, they are \(1 \times 1\), \(2 \times 2\), \(3 \times 3\) and \(4 \times 4\) (whole image).

Since many grids have overlapping parts, we need some strategy to query the database. Our query strategy is designed in following three steps. Firstly, we use top three \(1 \times 1\) grids with the largest number of features to query the database. A query is considered successful if there are sufficient matches. Secondly, if all three queries fail, one \(2 \times 2\) grid and one \(3 \times 3\) grid is used to query the database. Thirdly, if the queries fail, the \(4 \times 4\) (whole image) grid is used.

<table>
<thead>
<tr>
<th>Up to Step 1</th>
<th>(G_{1 \times 1}^{(1)})</th>
<th>(G_{1 \times 1}^{(2)})</th>
<th>(G_{1 \times 1}^{(3)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>64.4%</td>
<td>83.4%</td>
<td>89.8%</td>
</tr>
<tr>
<td>Up to Step 2</td>
<td>(G_{2 \times 2})</td>
<td>(G_{3 \times 3})</td>
<td>(G_{4 \times 4})</td>
</tr>
<tr>
<td>Accuracy</td>
<td>93.2%</td>
<td>96.6%</td>
<td>98.3%</td>
</tr>
<tr>
<td>Up to Step 3</td>
<td>(G_{4 \times 4})</td>
<td>(G_{5 \times 5})</td>
<td>(G_{6 \times 6})</td>
</tr>
<tr>
<td>Accuracy</td>
<td>98.3%</td>
<td>99.9%</td>
<td>99.9%</td>
</tr>
</tbody>
</table>

**Table 1.** The accuracy rate up to each step. In most cases, step 1 can return the correct query results.

With step three, the query results are as good as querying without partitions in the worst case. Table 1 shows the accuracy rate of image retrieval within a database with over 200 different images. On average, 1.6 queries are needed to
achieve accuracy up to 98.3%.

4.2. Scoring Scheme for Grid Retrieval

Every hierarchical grid is constructed into the retrieval database with its bag of features going through a vocabulary tree. In our implementation, a tree of 6 levels with branching factor 10 is used. A scoring scheme is designed so that vocabulary tree is adaptive to handle hierarchical grids.

We assign a 2-dimension weight \( w_i = (w^1_i, w^2_i) \) to each node \( i \) in the vocabulary tree based on its entropy \( e_i \) and level \( l_i \). The entropy is also 2-dimensional and defined as,

\[
e_i = (e^1_i, e^2_i) = (\ln N_i, \ln M_i),
\]

where \( e^1_i \) is the image entropy and \( e^2_i \) is the grid entropy. \( N \) is the number of 4×4 grids (image itself) and \( M \) is the number of all hierarchical grids in the database. \( N_i \) and \( M_i \) are the number of images and grids with at least one descriptor vector path through node \( i \). The performance is further improved by assigning the lower level nodes with higher weights. In our case, we double the weight for each lower layer node. Therefore, the weight \( w_i \) is defined as,

\[
w_i = e_i \cdot \frac{1}{2^{l_i}},
\]

with the root node has \( l_i = 0 \) and a leaf node has \( l_i = 6 \). Then the query vector \( q_i \) and database vector \( d_i \) are \( q_i = n_i w_i, d_i = m_i w_i \), where \( n_i \) and \( m_i \) are the number of descriptor vectors of query grid and database grid respectively with a path going through node \( i \). The 2-dimensional score is given based on the normalized differences between the two vectors,

\[
s(q, d) = (s^1, s^2) = ((q^1/d^1, d^1), (q^2/d^2, d^2)).
\]

The ranking will be based on \( s^1 \). If multiple grids have the same \( s^1 \), which is usually the case when there are fewer images, they will be ranked according to the values of \( s^2 \).

5. MATCHING WITH EFFICIENT DESCRIPTORS

Descriptor generations usually have much less computational cost compared to keypoint detections. Due to the robustness and efficiency of SURF descriptors, it seems promising if we can use it directly. However, since we do keypoint detections on a single scale, for a SURF descriptor, with no changes on its repeatedness property, the distinctiveness has been reduced. We proposed a new descriptor that is based on colors to compensate for its distinctiveness.

We use the normalized 64 dimension SURF descriptor as the first 64 elements in our 128 dimension descriptor. The next 64 elements will be two 32-D descriptors that are based on \( r \) and \( g \) values, where \( r = R/(R+G+B) \) and \( g = G/(R+G+B) \). When computing the two color descriptors, same orientation as SURF descriptor but a smaller neighborhood of pixels are used. As shown in Fig. 3, four 4×4 pixel neighborhoods around the keypoint are considered in forming the \( R \) or \( G \) color descriptor. A histogram with 8 bins is generated for each 4×4 neighborhood, which will give us 32 elements in total for both \( r \) and \( g \) descriptor. The processing time for the proposed descriptor is about 1.4 to 1.6 times of that for SURF descriptor, which is a trivial overhead.

6. AUGMENTED REALITY SYSTEM

We use LIDAR data and aerial images as the input to reconstruct 3D virtual environment which mainly includes buildings and the ground. The method [7] is used to convert LIDAR data to triangular meshes and [8] is used to automatically map the textures. Fig. 4-(a) shows one portion of the reconstructed virtual environment.

![Fig. 3. The proposed descriptor based on normalized R or G colors. (a) The four 4×4 pixel neighborhoods around a keypoint (b) Each 4×4 has 8 bins, giving a 32 dimension vector for both r and g descriptor.](image)

![Fig. 4. (a) The reconstructed virtual environment (b) An example of captured image in real world and its corresponding view in virtual world with calculated camera pose.](image)
the real scene, parts of the scene are occluded by the hands so the recognition results are seriously affected. The proposed hierarchical grids can solve this problem effectively.

7. EXPERIMENTAL RESULTS

7.1. Computational Performance

We have tested the proposed algorithms on a portable device with a 1.5GHz processor. Fig. 5-(a) shows our comparisons of four different matching schemes on 500 frames. As we can see from the comparisons, by shifting the scale space analysis from detection process to offline process and replace scale detections with retrieval techniques, we can almost double the frame rate. Moreover, with hierarchical partitioning method, the matching time can be further reduced significantly with frame rate up to 4 to 6 times.

![Image](image_url)

Fig. 5. (a) comparisons of four different matching schemes on 500 frames. (b) grid retrieval compared with image retrieval.

7.2. Retrieval Accuracy for Hierarchical Grids

In our augmented reality system, we have to handle the occluding cases that are either caused by moving vehicles or passengers, or by users’ hand during interactions. The latter case is more challenging because the occlusions are usually larger, moving faster and lasting longer. With the hierarchical partitioning grids, we can effectively handle these occlusions without affecting the performance. Fig. 5-(b) shows the performance of grid retrievals compared to image retrievals. All the retrievals are conducted with occluded images in a database with over 200 different images.

8. CONCLUSION

We propose an efficient matching method for robust features. We use retrieval to replace the scale space analysis in feature detection process. An adaptive scoring scheme and a more distinctive descriptor are also proposed for performance improvements. Besides, we have developed an outdoor augmented reality system that is based on our proposed methods. The system demonstrates that not only it can achieve robust matchings efficiently, it is also capable to handle large occlusions such as passengers and moving vehicles.

9. REFERENCES