Object Identification by enhanced Local SIFT Features

Dongbo Wang (dongbo@cs.brown.edu)

Abstract: The project is to design and implement a framework for our smart camera network, to effectively identify an object when given a certain amount of continuous frames of such object. An enhanced local feature descriptor is used to represent detected objects in our smart camera network. We select stable and distinctive SIFT features from an object as well as maintain the viewpoint relations existing amongst different features. The matching algorithm takes into consideration not only the appearance of SIFT features but also those viewpoint relations. Such descriptors are constructed in in-camera scenarios with the support from tracking algorithms, and can provide useful information about the object matching when solving tracking in cross-camera scenarios.

1. Project Introduction

This project aims to provide the searching functionality to our smart camera network so that the user can query by example. Therefore, we need to figure out the effective way to represent detected objects in our network and do object matching, which will also help get better results in cross-camera tracking. There are three local features being used in object matching in our platform: location, color histogram, and SIFT feature [1, 2]. Location and color histogram are very effective to match objects in single camera scenario, but due to the variance in factors such as the illumination and view angle amongst different cameras, these two features are less useful when it comes to match object that are captured from different cameras. Contrasted with the location and color histogram, SIFT features are robust to changes in image scale, noise and illumination, so we try to represent objects by using SIFT features, which make objects from different cameras comparable.

2. Related Work

Detecting distinctive local image features is a noticeable progress occurred in computer vision. The local SIFT features has been applied in image retrieval, object recognition, and autonomous vehicle navigation tasks. In [3], indexing descriptors are extracted from local regions to retrieve the right CD-covers from a database of 40000 images of popular music CD's. In [4], local features are used in vehicle navigation to estimate the global uncertainty of epipolar geometry. The Locality Sensitive Hashing algorithm [5, 6] aims at searching for the nearest points in a high dimensional space, so it is also used in some image retrieval applications such as [7]. In our project, both the local SIFT feature and the LSH algorithm are used to find an effective way to do object matching and searching in our camera network.

3. “Bag-of-features” approach

Initially, we try to make the searching functionality able to do the query based on single frame taken in our camera network. The idea is to use the “bag-of-features” approach: when detecting an object from frames, we collect all fore-ground SIFT features of this object, and use the SIFT feature collection to represent this object. SIFT features are actually 128-dimension pointers, so when it comes to searching, we use Locality Sensitive Hashing algorithm to construct the index of all SIFT features from all detected objects, and each SIFT feature points to the object ID it belongs to. LSH is used to search for the nearest point in a high dimensional space. Given an example frame, we first extract all fore-ground SIFT features from the frame, and then for each of those features, use the LSH index to find the most similar one, and add one score to the object it points to. In the end, we compare the score each object gets, and the one with the highest score is supposed to be the query result.

Unfortunately, the experiment result shows this approach is not reliable. When the example frame in a query is one of the frames the object is detected from (this means the fore-ground SIFT features on this example frame are already included in the SIFT feature collection representing the object, and are taken into account when constructing the LSH index), this approach usually give the correct query result. But when the example frame is not in the frames the object is detected from, the query result seems to be totally random.

Due to the limitation in image resolution and sharpness, usually there are not too many SIFT features can be extracted from a frame taken in our camera network. Thus the useful information provided by only one example frame may not be sufficient enough to help us get the correct query result. So, we adjusted the initial goal, instead of basing on only one example frame, we try to make the query base on a certain amount of continuous frames, so as to get more information to do the matching in the candidate objects. However, more example frames in the query does not help out in our “bag-of-features” approach: when the example frames are not from those where the object is detected, the query result still seems to be random. This “bag-of-features” approach is later given up for three reasons:

(a) Unreliable query result. We cannot guarantee that the SIFT feature collection representing an object covers all features about this object from all frames taken in our camera network, and the query result is unacceptable if the SIFT features from the example frame is not already covered.
(b) LSH index does not support incremental update. Every time there are new objects, we need to rebuild the LSH index, and this will be time consuming when the number of detected objects in our camera network goes up.
(c) LSH index does not help when matching is done in a limited number of

candidates. In tracking, sometimes it's necessary to do object matching in a limited number of candidates, for example in three or four candidates. In such cases, we cannot afford to build an index only for several candidates, so we have to do object matching solely by SIFT feature matching between the SIFT feature collections that represent the objects.

4. Enhanced local SIFT feature approach

The enhanced local SIFT feature approach is based on Yong Zhao's paper about object identification [8]. We use a novel SIFT feature based descriptor to represent an object, which consists of a set of stable on-body SIFT features of the object, and the view point relation information between any two features in the set.

Instead of collecting all fore-ground SIFT features for an object, we filter the features to get stable ones. The SIFT features we collect from frames usually contain many instable ones, which probably only appears once and never shows up again. Instable features are useless for object matching since they do not reflect actual characteristics of the object, so that it's barely possible to have SIFT features of the same object from other frames to be matched with it. Worse still, these instable features may even increase the inaccuracy of the matching result, because when we do SIFT feature matching between two objects, the existence of instable features will cause more error matching. The movement of objects may cause the instability of features extracted, and when features are extracted near the edge of the fore-ground object, they are also very possible to be instable ones.

There are two ways to filter SIFT features. First, we only want to keep the features that are fully on the body of the fore-ground object. Every feature contains its position and scale information, denoting where this feature locates on the image, and also the size of this feature. We regard the position of the feature as the center of a circle, using six times of the feature scale as the radius of the circle, and then we check eight pointers on the circle as shown in Figure 1. If all those eight pointers are on the body of the fore-ground object, we believe this feature is valid, otherwise the feature is discarded.

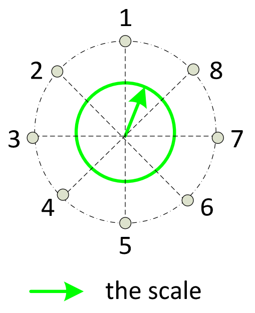


Figure 1. checking pointers

Second, we only want to keep the stable features, which tend to show up again later in the neighboring continuous frames. When collecting SIFT features for an object in a mount of continuous frames, for each frame, we only keep the fully on-body features. At the same time, a buffer is maintained which holds all on-body features from the last five or ten frames (the exact number is influenced by the total number of continuous frames provided). A stable feature will usually be repeatedly detected only in several continuous frames, so it's useless to make the buffer hold features from too many previous frames. Normally, holding features from the last five or ten frames will be sufficient enough, and thus the number of frames the buffer can contain is fixed. When we obtain the fully on-body features from a new coming frame, we try to match those new features with the ones in the buffer, and only the new features that can be matched for a fixed number of times are regarded as stable features. Then all on-body features from this new frame are inserted in the buffer, and if the buffer was already full, the on-body features from the most previous frame will be removed.

The process to obtain stable SIFT features from a frame is as depicted in Figure 2. As shown in the Figure 2-a, the buffer presently has already contained all on-body features from frame 1, frame 2, frame 3 and frame 4. Then in Figure 2-b, there comes the frame 5, and the on-body SIFT features are extracted from the frame 5 for the object. In order to get stable on-body SIFT features from the new coming frame, we do SIFT matching between the new features we get and features from each of the previous frames kept in the buffer. While doing the matching, we keep track of how many times a new feature is matched with a feature from the previous frames. In this case, a new feature can be matched 4 times at most, because the buffer only contains the sets of on-body SIFT features from the last 4 frames. A feature being matched 4 times means that we can find a similar on-body feature matching this new feature from each of the last 4 frames. Now we have the statistics about how many times each new feature is matched. Suppose we set the threshold that a new feature is regarded as a stable one if it got matched for at least two times. Then we can filter the new on-body features based on the times each new feature got matched, and obtain the set of stable on-body features of the object from the new coming frame. The last step is to insert all new on-body features (all new on-body features, not only those stable ones) to head of the buffer. If the buffer was already full, the on-body features from the oldest frame will be removed from tail of the buffer.

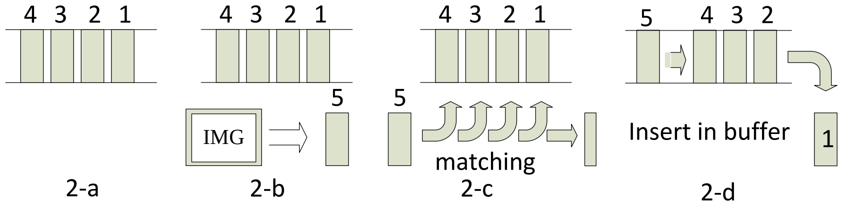


Figure 2. Obtain stable SIFT features

What if the buffer is empty or not full yet when a new frame is under processing? In such a case, we keep all on-body features from the new frame as stable features, and insert all of them into head of the buffer. In this way, we can obtain stable on-body features from every frame we process. Since they are comparatively stable, they are more possible to show up again in new frames from other scenarios, and thus more accurate for the object matching. The stable feature tends to show up in continuous frames, so stable features from a new coming frame may contain repeated features that are already included in the stable features from the past several frames. Repeated stable features should be removed to get distinct stable features for an object. So, after we get the stable features from a frame, we do a SIFT matching between the new ones and all stable features of the same object that are already obtained. If a new stable feature is matched, we believe it is a repeated one. Therefore, we finally can get all distinct stable features of an object from the continuous frames we processed.

Compared to the "bag-of-features" approach, when we do object matching solely by SIFT matching between the stable features from two potentially identical object, we always can get better results, because we avoid the interference from the instable, bad one-time SIFT features. However, representing an object solely by the stable feature collection is still not good enough. There is a problem about SIFT matching: the more features we do matching on, the more features will be matched. So if a candidate object has more stable features than another candidate object, it's possible that the former will have more matched features than the latter when doing object matching for a new object, even if the new object should actually match to the latter one. Due to the image resolution, the number of stable features we can get from a frame is heavily influenced by the contrast of the color on the fore-ground object, and the more drastic the color contrast is, the more SIFT features can be extracted. Therefore, it is unavoidable that some objects may have more stable SIFT features in their collections than others even if equal number of continuous frames is processed for them. When doing the object matching by solely SIFT matching between the collections of stable features that represents objects, candidate objects with more stable features will get extra benefits. This will greatly affect the accuracy of the object matching, especially when the candidate objects have large difference in the number of stable features in their collections. To solve this problem, we introduce the viewpoint relations of the stable features.

The view point relation is defined as the closeness of the view points from which two features can be detected. In fact, we further simplify this relation to a binary form: whether or not two features can be observed at the same time. If two features are obtained from the same frame, we say they can be observed at the same time. We create a "viewpoint relation map" to represent this viewpoint relationship. It is implemented as a 2-dimensional sparse binary matrix V. The size of this matrix is the number of distinct stable features we obtained for the object. The matrix entry V[*i*, *j*] = 1 if feature *i* and *j* have been detected simultaneously for at least one time. If feature *i* and *j* have never been detected from the same frame, we set 0 to V[*i*, *j*].

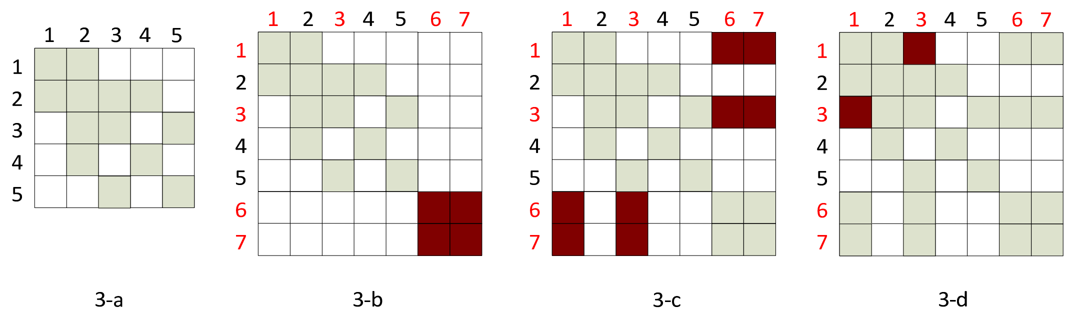


Figure 3. View Point relation Map Update

We illustrate the creation and update of a viewpoint relation map in Figure 3. As shown in Figure 3-a, suppose currently we have 5 stable on-body SIFT features for an object. The black and while are used to represent 1 and 0 respectively. When a new frame comes, 4 new stable on-body SIFT features are detected from the new frame. Two of the new features are matched to the 1st feature and 3rd feature from the existing stable features, and the rest two new features are brand new ones. Then these two brand new features are added to the stable feature set as the 6th and 7th features, and the corresponding slot in the viewpoint relation map is set to be black because they are obtained from the same frame, as is shown in Figure 3-b. As for the existing features, only the 1st feature and 3rd feature can be observed with the two new features at the same time, so we set the slot [1, 6], [1, 7], [6, 1], [7, 1], [3, 6], [3, 7], [6, 3] and [7, 3] to be black, as shown in Figure 3-c. The most interesting part here is the 1st feature and 3rd feature. Before the new frame is processed, the slot [1, 3] and [3, 1] are while, because they are not detected in the same frame. But those two features show up together in the new frame, so slot [1, 3] and [3, 1] are updated to be black, as indicated in Figure 3-d.

By using the view point relation map and the set of stable on-body SIFT features, the new descriptor that represent an object contains much more information contrasted with the "bag-of-features" approach. Suppose the descriptor for object A and B are $\{F_A, V_A\}$ and $\{F_B, V_B\}$ respectively. When doing object matching between A and B, we first do SIFT matching between F_A and F_B. Suppose the indices of matched features F_A and F_B are $\{id_{A1} - id_{B1}, id_{A2} - id_{B2}, id_{A3} - id_{B3}, \ldots, id_{AN} - id_{BN}\}$. Then, the likelihood of object A and B is calculated as:

$$\mathrm{Score}_{AB} = N + \sum_{x=1}^{N} \sum_{y=1}^{N} \{VA[idAx, idAy] \cdot VB[idBx, idAy]\}$$

According to the equation, the likelihood score of object A and B includes the number of features get matched and the similarity of the view point relations of those matched features. In another word, the evaluation not only counts the number of individual local stable on-body features that are matched, but also how similar the global view point of these matched features. Therefore, the matching result would be more accurate than solely using SIFT matching between two sets of stable on-body features, even if one contains more amount of features than the other.

5. Design and the Implementation

We designed and implemented the “SIFT_DETECTOR” module. Once an object is under tracking, the SIFT_DETECTOR can accumulatively update the descriptor of the object as more frames are provided. The process of searching is simply finding the most similar candidate by comparing the example descriptor with descriptors of all candidate objects.

The Figure 4 indicates the diagram of SIFT_DETECTOR. Once a new frame comes, the underlying in-camera tracking will detect the fore-ground object, and match it with an existing object or decide it to be a new object based on the color histogram and location information. The underlying in-camera tracking will provide the binary blobs for each fore-ground object detected or newly recognized. Then we do SIFT feature extraction from the whole frame. The raw SIFT features and the binary blobs (an object blob may split into small pieces during background subtraction) of an fore-ground object detected or recognized from the same frame will be passed to the SIFT_DETECTOR, along with the existing descriptor of the object, or a fresh descriptor if the object is newly recognized.

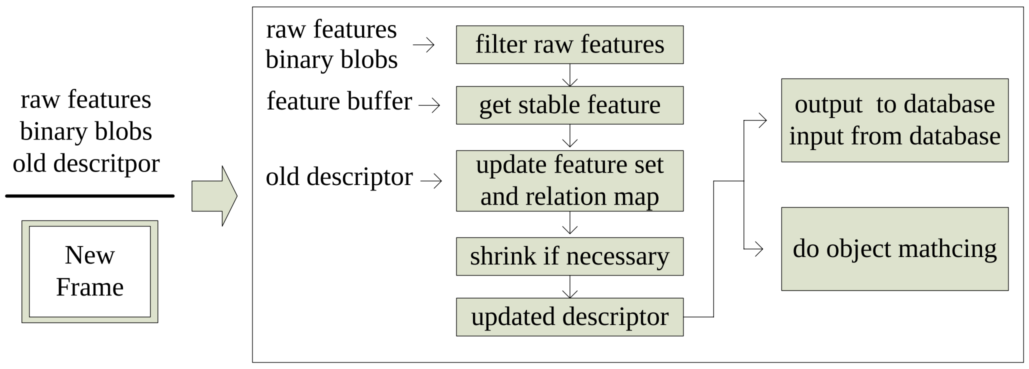
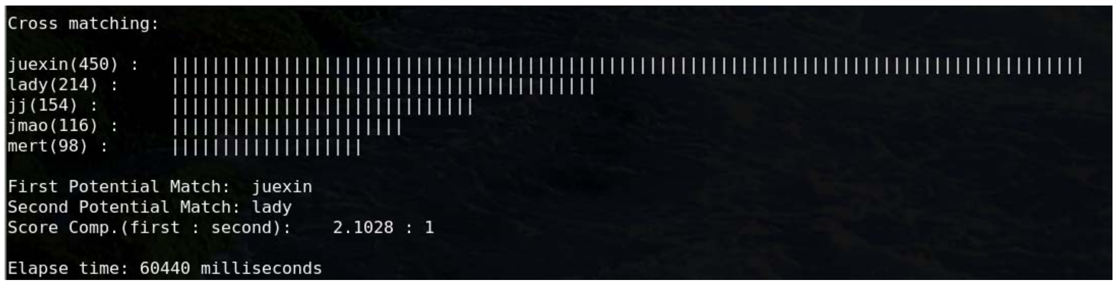


Figure 4. the Diagram of SIFT_DETECTOR

Based on the blobs, we filter the raw features according to their positions, to get all on-body features. The descriptor passed in mainly consists of three components: a set of existing on-body SIFT features of the object that this descriptor represents; a view point relation map of existing features; a buffer that contain on-body features from the past 10 frame, or an empty buffer if it’s a fresh descriptor. The new on-body features will be compared with the previous on-body features in the buffer, to get the stable ones. The threshold to decide a feature is stable or not we are using in our camera network is 4. In another word, after comparing with the on-body features from the previous 10 frames, any new on-body features that have been matched for at least 4 times are regarded as stable features. After we get the stable features, we compare them with the set of existing stable features, to get the distinctive ones, and then insert them into the set and update the view point relation map.



```
Cross matching:

juexin(450) :   ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
lady(214) :     ||||||||||||||||||||||||||||||||||||||||||
jj(154) :       ||||||||||||||||||||||||||||||
jmao(116) :     |||||||||||||||||||||||
mert(98) :      |||||||||||||||||||

First Potential Match:  juexin
Second Potential Match: lady
Score Comp.(first : second):    2.1028 : 1

Elapse time: 60440 milliseconds
```

Figure 5. An example of searching result

The SIFT_DETECTOR also needs to shrink the descriptor when it's necessary. We set a threshold for the size of the descriptor. If it exceeds the threshold, the set of existing stable features will be scanned, to find out those less stable ones. Every stable feature in the set labeled with the number of times it get repeated by stable features from the later frames. The larger number it is labeled, the more stable it is. Once we get the less stable features, they are removed from the set, and the view point relation map is updated. The SIFT_DETECTOR also provides interfaces to output a descriptor into the database and input an existing descriptor from the database, as well as the interface to do matching to two descriptors, which return the likelihood scores of these two descriptors.

Figure 5 shows an example of object searching by using the SIFT_DETECTOR. There are five candidate objects, and object "juexin" is one of them. The example descriptor is also "juexin", but it is created and updated by continuous frames taken in a different scenario. The result shows that the correct candidate is found out. And the ratio between the first candidate's score and the second candidate's score is 2.1028 : 1. The larger the ratio is, the more confident the result is. But the SIFT_DETECTOR is also not perfect, sometimes we also get 1.4 : 1 when doing the matching, which means the result is not very confident. Actually, the configuration of thresholds is important, and has a lot influence on the matching result. Currently we set the thresholds by experiences, and try to make the matching result as confident as possible. The future work would be set up a scientific way to select the more appropriate value for the threshold in our framework, to further improve the matching result.

6. Conclusion

We have worked on finding an effective way to represent objects and do matching and searching in candidate objects. The enhanced local SIFT feature approach shows better results compared to the "bag-of-features" approach, but still need further polish on the configuration of thresholds. The framework is implemented and being used in the cross-camera tracking.

Acknowledgement

I would like to thank my advisor Professor Ugur Cetintemel for all his valuable help and support during my past one-and-a-half-year study at Brown University. I would also like to thank Yong Zhao and Jie Mao with whom I worked in the team. Their attitudes to work and life as well as their broad knowledge are always what I am learning from.

Reference

[1] Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. 2. pp. 1150 – 1157.
[2] Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision. 60, 2. pp. 91-110.
[3] Nistr D, Stewnius H. (2006). "Scalable recognition with a vocabulary tree". Proceedings of Conference on Computer Vision and Pattern Recognition.
[4] Nistr D, Engels C. (2006). "Visually estimated motion of vehicle-mounted cameras with global uncertainty". SPIE Defense and Security Symposium, Unmanned Systems Technology VIII.
[5] Mayur Datar, Nicole Immorlica, Piotr Indyk and Vahab S. Mirronkni. (2004). "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions". Proceedings of the twentieth annual symposium on Computational geometry. pp. 253 – 262.
[6] Wei Dong, Zhe Wang, William Josephson, Moses Charikar and Kai Li. (2008). "Modeling LSH for Performance Tuning". Proceeding of the 17th ACM conference on Information and knowledge management. pp. 669 – 678.
[7] Piotr Indyk, Nitin Thaper. (2003). "fast image retrieval via embeddings". The 3rd International Workshop on Statistical and Computational Theories of Vision (at ICCV).
[8] Yong Zhao and Gabriel Taubin. "Stereo Enhanced Local Features for Object Identification".