Using a Reference Point for Local Configuration of SIFT-like Features

Martin KLINKIGT[†] and Koichi KISE[†]

† Graduate School of Engineering, Osaka Prefecture University
E-mail: †klinkigt@m.cs.osakafu-u.ac.jp, ††kise@cs.osakafu-u.ac.jp

Abstract In this paper we propose a model to describe the local configuration of SIFT-like features. This information helps to overcome the problem with background clutter in images, i.e., information extracted from areas in the image, which do not belong to the object. Our model is not static as many other proposed models. Instead we calculate during detection a reference point of the object and verify this with the features extracted from the query image. This additional step increases the detection performance around 6% even under hard conditions, for exmaple, if training images and query images contain background clutter.

Key words object recognition, SIFT, Bag-of-Features, Constellation Model, Implicit Shape Model, Generalized Hough-transform

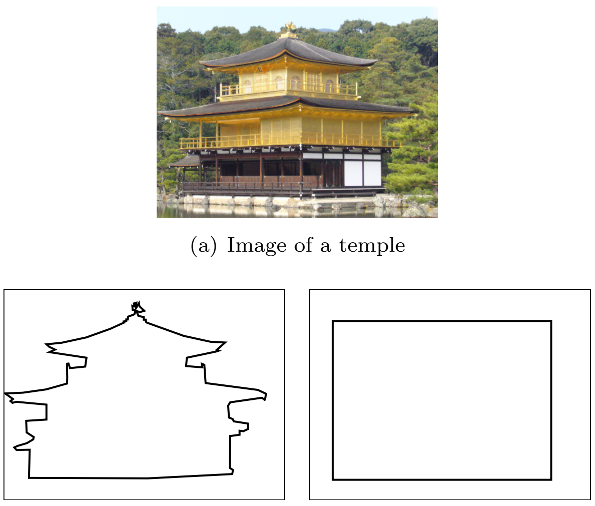
1. Introduction

In computer vision object recognition is the task to categorize images by their containing objects. For the application of such a task, the image is described with the help of local features. Often the local configuration of these features is just ignored by a performing a Bag-of-Feature strategy. If one has to struggle with background clutter i.e., information extracted from image areas, which do not belong to the object, becomes problems. For background clutter it can easily happen that it has a high similarity even to wrong objects. If the amount of such clutter is too high, the results of the system become meaningless.

To utilize the local configuration of the features seems to be a suited solution to address this problem. It is not reasonable to assume that also the configuration or in other words the shape, of the background matches perfectly. This would be contradictory. Current models to hold information about the local configuration often can not be used without human interaction. We propose a model which needs no human interaction and is very simple to apply. We utilize a reference point and verify the configuration of matching features. With our proposed method we have achieved an increased recognition rate of more than 6%. Even under hard conditions, if the training and query images contain background clutter, our approach still performs better than a Bag-of-Features strategy.

2. Background Clutter

Background clutter is one of the major problems in object recognition. By background clutter we mean the


(a) Image of a temple

(b) Well segmented object

(c) Bounding box around the object

Fig. 1 Object and possible cutting approaches

part of the image which does not directly belong to the object of interest. Information in any form of these parts can confuse a recognition system. If the amount of information extracted from background overweigh the available information about the object itself, the recognition system may not be able anymore to detect this object. With this section we briefly explain frequently applied solutions for this problem.

2.1 Segmentation

One solution to solve the problem of background clutter is to segment the object from its background. This normally gives the best results, since only information form the object is used. Such a segmentation is of a high quality. However, it is hard to obtain it. Even nowadays a computer is not able to directly provide such a segmentation which in the end means that a human has to do this task. This can be unacceptable in some environments, since the time needed to segment an object often becomes long. Figure 1(a) shows such an object, here a temple, and its segmentation in Fig. 1(b).

2. 2 Bounding Box

Another solution which would need less time to be prepared by a human user is the bounding box. Here the object is not segmented perfectly from its background, instead a less precise box around the object of interest is used. The disadvantages of this solution are that the inner of the box still can contain much background if the object is not quadrangular and/or some parts of the objects laying outside of the box as we can see in Fig. 1(c). So a bounding box can not solve the problem of background clutter completely but it can reduce its influence. Still a computer is not able to perform such a task, it is again up to the user to provide such a bounding box around the object of interest. This task seems to be less work than a full segmentation of the object, if the user would not have to do so for several images per object

2. 3 Image Clustering

To cluster many images is a fundamentally different approach. Here the image is not cut in any sense. The system extracts information from independent, small regions of interest in the image. This is done for many images of the object which should show the object in front of different backgrounds. With the help of clustering the system can determine, which regions of interest are frequent in all images, and therefore, should belong to the object. This approach is not less work, since sometimes several hundreds of images must be provided. Also it may be even impossible to show the object in front of different backgrounds, as it would be the case for buildings.

3. Related Work

After the discussion about the different solutions in solving the problem of background clutter, we will discover one possible root of this problem and give an overview of related work.

For object recognition often the local configuration of features is ignored by applying a Bag-of-Feature approach. Here only the description in the point of interest is used during detection. Such an approach is simple and can be easily implemented. However, the local configuration of the features also consists of useful information.

3. 1 Constellation Model

In a constellation model [1] the local information of the features from an object is stored as positions in a 2-dimensional probability space. To keep the resulting graph computable only a few features (normally around 5) are used to create the model. It could also be made scale invariant through normalization concerning one feature. As Fergus et al. have already pointed out, this approach highly depends on the feature detector [2]. If it fails to detect these features defined over large regions of the image (e.g. the complete wheels of a bicycle), the results are not useful anymore. To apply this model the image must be background free, which is similar to a segmented object from Section 2. 1.

3. 2 Implicit Shape Model

Leibe et al. have proposed the implicit shape model [3]. The shape is no representation of the local configuration of the feature to each other as it was the case for the constellation model. For every feature its relative position to a predefined centroid point is used. During detection the features extracted from the query image are compared and the algorithm proposes a possible centroid position. Dense regions of centroids are a hint for a possible object. In [4] this model was made scale invariant. This model is flexible enough to address the problems of a high intra-class variation of the object. This is achieved by sharing the features of the object learned from different images. This model does not require a perfect segmentation. The declaration of a bounding box around the object itself is necessary (Sec. 2. 2). To learn a centriod many such images must be prepared. This model is, as well as the constellation model, not rotation invariant.

3. 3 Weak Geometric Consistency

The idea of the Weak Geometric Consistency (WGC) as provided by Jegou et al. [5] is not to create a model in that sense. The used local features, which describe the object, also hold some general weak information about their individual local configuration, as an orientation and a size or scale. During detection these properties are used to calculate differences in the values of matching features. These differences are categorized into different classes. The object which concentrates its matched features in similar classes has a better matching local configuration. This approach becomes clear, if we consider that an object is normally rotated or scaled unique as a whole and not separated parts are rotated or scaled differentially.

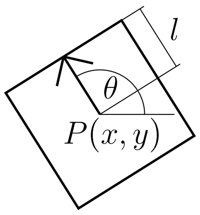


Fig. 2 SIFT Feature

Here we do not have a direct condition concerning the background. A condition concerning the background lies in the definition of the model. The root of the problem is the use of these global histograms which of course also contain the background. This implies that such an information is only useful, if the amount of clutter does not dominate over the amount of information of the object itself.

3.4 Generalized Hough Transform

With the generalized Hough transform arbitrary shapes can be detected. This is often achieved with the help of look-up tables which defines the shape. Interations about different settings in the position, orientation and shape are typical for this approach which leads to high computational costs. The shape is also not learned from images, instead its predefined by a human.

4. Necessary Elements of the Proposed Method

In this section we briefly explain necessary elements of our proposed method. These are SIFT and PCA-SIFT features, object recognition by voting and hashing.

4.1 SIFT Features

The Scale-Invariant Feature Transformation (SIFT) was provided by Lowe in [6]. It consists of a high dimensional feature descriptor vector of dimension 128. For the calculation of the entries of the vector mathematical values like derivations are used. This descriptor has been used successfully in many fields in the current research of object recognition.

PCA-SIFT [7] uses even a higher feature descriptor (1024 dimensions) and compresses these values by Principal Components Analysis to the most significant eigenvalues. The resulting dimension is 36. As Rahul et al. also have shown, the detection performance can be even better but most importantly, the used memory to store these features is reduced significantly.

These features are computed for regions of interest (ROI) in a certain location in the image. These types of feature additionally provide the scale and orientation of the ROI as shown in Fig. 2 where l is the scale, θ is the orientation and the position of the feature in the image is $P(x, y)$. These values provide a weak information about the "shape".

4.2 Object Recognition by Voting

Voting is frequently used in object detection. For voting, local features are used to represent an image, and therefore, an object. These features together with the object ID are stored in a database. The number of features may vary for different objects and images. From the query image, in which an object should be recognized, the same type of features are extracted. For each feature similar matches in the database are searched. If features are found which fulfill a certain threshold in similarity, the match casts a vote for the corresponding object. The voting can also be varied in the way that only the best matching feature from the database cast a vote for the object. The object with the most votes is supposed to be the correct result.

4.3 Hashing

A major trouble one has with object recognition by voting is to find near features even in a large database. A naive approach to just compute the distance to every feature is of complexity $\mathcal{O}(g \cdot h)$ where h is the number of features from the query image and g the number of features in the database. The number h normally is not so large, while the number g can easily become several millions and more.

Hashing is an approach which reduces the complexity remarkably. A hash table consists of many bins, which are addressed via a hash value. Such hash values are calculated from the features. The original features are then stored in the corresponding bin. To calculate such a hash value for our used PCA-SIFT features, the original real-valued vectors are transformed into their scalar quantized form with 2bit per dimension. Let $\boldsymbol{p} = (p_1, \ldots, p_n)$ be the scalar quantized feature vector. The bit-vector representation $\boldsymbol{u} = (u_1, \ldots, u_d)$, where $d < n$ is the desired number of values, which should be taken into account, is calculated as

$$u_j = \begin{cases} 1, & \text{if } p_j - \theta_j \geqq 0; \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where θ_j is the median of the original vector values of each dimension j. Finally the hash value itself is calculated from $\boldsymbol{u}$ as

$$H_{\text{index}} = \left(\sum_{j=1}^{d} u_j 2^{(j-1)} \right) \mod H_{\text{size}} \tag{2}$$

where H_{size} donates the size of the hash table. In the case of a collision a specified number c of elements can be kept by a chaining method. If this c is exceeded, all entries for this hash value are marked as invalid.

5. Reference Point

After we discussed the advantages and disadvantages of current approaches and gave some elements in the last sections, we explain our proposed method based on a reference point in detail. From the discussion we can derive some properties a suited model for the local configuration of features ideally should have:

(1) Learning aptitude: The model should be adaptive from images, without any additional information, like a segmented object, bounding box or other specific information.

(2) Computation time: The computational cost for the model should be within reasonable time constraints.

(3) Stability: The model should be stable and not only work under certain conditions.

While with the first two constraints we intend a decreased interaction time of the user with such a system, the later one is to focus on the problems some models like the WGC (Sec. 3. 3) can have, if even only a small amount of background clutter must be considered. In such an environment one can ask, whether it is possible to provide a practical approach.

Our proposed method increases the recognition performance with respect to the three above mentioned conditions. The idea is that we do not generate one fixed model during the learning phase of the system. Instead the model is generated during detection, which gives us a higher flexibility.

5. 1 Learning Phase

Beside the PCA-SIFT descriptor we store for all images in the database the scale l, orientation θ and the position (x, y) of the feature as shown in Fig. 2. The additional memory needed for these 4 values is marginal compared to the 36 dimensions of the PCA-SIFT descriptor. Additional computational cost does not occur, since these values are also needed to calculate the descriptor.

For all features we calculate its hash value as described in Section 4. 3, and store only their occurrence together with the additional values.

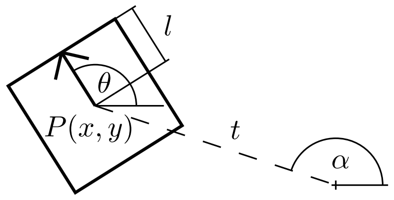


Fig. 3 Additionally calculated properties

5. 2 Recognition Phase

Assuming an image shows an object of interest, which is stored in the database, we extract the same type of PCA-SIFT features. Again we calculate the hash value of these features and search for near features concerning the descriptor part in the corresponding bin. Near features have to fulfill a certain threshold. The simple scoring approach as proposed in [8] now cast a vote for this object.

In our extension we do not stop the scoring at this point. Assuming we have all possible matches between query features and features from the database, we now create our model. Let F_D be the set matched features for one image I in the database. For these features we estimate one point which we call *reference point* or in short RP. For this point we take the mean (RP_x, RP_y) of the positions of the PCA-SIFT features. Let K_x be the x-coordinate and K_y the y-coordinate of a point K:

$$RP_x = \frac{\sum_{K \in F_D} K_x}{|F_D|}, RP_y = \frac{\sum_{K \in F_D} K_y}{|F_D|} \qquad (3)$$

From this reference point we calculate for every matched feature in the database image two new values which are shown in Fig. 3. These are the distance t of the feature to this reference point RP and the enclosed angle α.

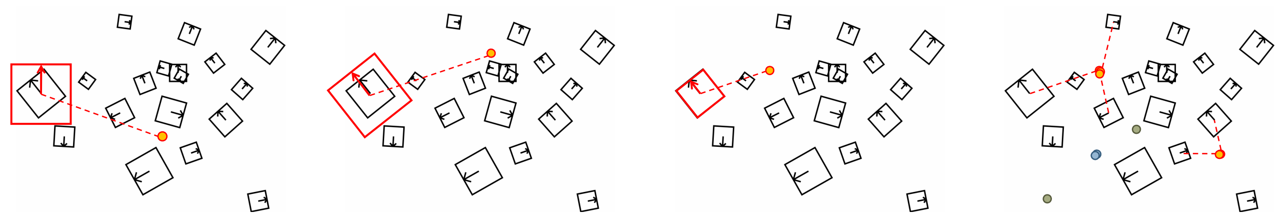
After we have these two new values we proceed further with the query image. Figure 4 illustrates the following steps.

(a) We place the matched feature from the database with its attached new properties l and α over the feature from the query image.

(b) After that we correct the orientation of the database feature by letting it show in the same direction as the query feature and update the location of the reference point.

(c) Finally we correct the scale of the database feature to be equal to the scale of the query feature and again update the location of the reference point.

(d) We perform these steps for all matches of the features, while one query feature may have more than one matching feature in the database.



(a) Placing mathcing featues. (b) Adapt the orientation. (c) Adapt scale. (d) Resulting reference points.

Fig. 4 Steps of reference point matching.

In some sense this looks like a generalized Hough transform from Sec. 3. 4. However, the major difference is that we do not need any iteration over several possible shapes in every point and that we learn the shape from images. Also the proposed method is based on a comparable idea as the implicit shape model from Sec 3. 2. Here the difference is that we do not need a bounding box of the object, which means that we can apply our method also for training images containing background clutter.

After these steps we have locations of proposed reference points in the image plane for the different objects. In the ideal case only the reference points of the correct object would be agglomerated in compact regions, while the proposed reference points of the incorrect object would be spread over the whole image plane. This assumption is oversimplified since we are always confronted with numerical inaccuracy. Therefore, we have to apply some clustering to find such small and compact region, which will be described in the next section.

6. Clustering

A clustering of the proposed reference points is necessary to find dense regions. For such a task approximations like a k-means clustering can hardly be applied, since the suited k is nearly impossible to select automatically. We have selected *The RNN algorithm for Average-Link clustering with nearest-neighbor chains* [3]. The main idea is to create *reciprocal nearest neighbor* pairs (RNN pairs) in a randomly selected start point, storing them in a chain and cluster them after the chain is canceled. A complete proof of this strategy could not be given in this paper but in short it relies on Bruynooghe's reducibility property which means that the agglomeration of two clusters may only decrease to a third cluster. Algorithm 1 summarizes this algorithm in the pseudo code.

For the clustering we make one slight adaption. We do not treat all cluster points equally. If the distance t of a matching feature to its proposed reference point becomes large, then even small errors in the adaption of the orientation can lead to significant misplaced reference points. To solve this problem we estimate a weight from this value t. Let RP' be a reference point and t' the final distance (Fig. 4(c)) of the feature in the query image to RP'. The weight $w_{RP'}$ for reference point RP' is:

$$w_{RP'} = \sqrt{t} \tag{4}$$

The weight of the agglomerated cluster is simply the addition of the weights of the original clusters.

One object possibly can be represented in the database by more than one image. If features of more than one image match with features from the query image, we treat these images separately. Reference points of different images are not clustered together. If one would do so, clustering becomes easy, if two near duplicates of the same image are stored in the database. Additionally, the position of the reference point must

Algorithm 1 The RNN algorithm for Average-Link clustering with nearest-neighbor chains.

```
$last \leftarrow 0$
$lastsim[0] \leftarrow 0$
$L[last] \leftarrow v \in V$
$\mathcal{R} \leftarrow V \backslash v$
while $\mathcal{R} \neq \emptyset$ do
  $(s, sim) \leftarrow$ getNearestNeighbor$(L[last], \mathcal{R})$
  if $sim > lastsim[last]$ then
    $last \leftarrow last + 1$
    $L[last] \leftarrow s$
    $\mathcal{R} \leftarrow \mathcal{R} \backslash \{s\}$
    $lastsim[last] \leftarrow sim$
  else
    if $lastsim[last] > t$ then
      $s \leftarrow$ agglomerate$(L[last], L[last-1])$
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{s\}$
      $last \leftarrow last - 2$
    else
      $last \leftarrow -1$
    end if
  end if
  if $last < 0$ then
    $last \leftarrow last + 1$
    $L[last] \leftarrow v \in \mathcal{R}$
    $\mathcal{R} \leftarrow \mathcal{R} \backslash \{v\}$
  end if
end while
```

be somehow normalized, since the object may be at different location in different images. Currently we do not apply such a normalization.

7. Voting

After we have performed the clustering on the proposed reference points, the question is how this information can be used to increase the recognition performance of the system. This will be described in detail in this section.

Every cluster will have at the end at least one reference point, the point it initially started with. During the clustering the number of points usually increases. If many reference points could be agglomerated in on cluster, then the local configuration of the features is reliable. This means that the configuration of the features, as they were placed in the training image, is in a similar way, as in the query image. At this point we take a special care of clusters which only have one point. These clusters come from matching features which do not have any meaningful local configuration. Such clusters are discarded and we do not analyse them further.

For the clusters which contain more than one point, we calculate a score for the corresponding object. Taking the sum of containing points leads to excellent results and its implementation is straightforward. As the final score for the object, we take the highest achieved score from one image of this object. At the end we just apply a simple normalization at the scores lie within the interval $[0, 1]$, where 1 is then the highest value.

8. Experiment and Discussion

In our experiment we analysed the performance of a simple voting strategy as provided by [8] and our proposed method to use shape information with the help of the reference point.

8.1 Dataset

As a task we let the systems detect temples and shrines from all over Japan. As training images we used all images from Wikipedia provided for these buildings. In detail we learned all buildings, which belong to the classes "temple in Kyoto Prefecture" and "treasure of Japan". The number of objects is 84 while the number of provided images is 819.

By using such a "public" data set the difficulties for the system are already high. The objects have a high similarity among all classes, since they are all temples or shrines. On the other hand, the objects are not segmented from the background and so the database contains much information extracted from these regions. This would not be a serious problem, if the background would be different for the objects. However, this is not the case. In the background we often have trees or persons. These trees have a high similarity in all images. Voting based on these features will result in a random ranking of the objects.

To analyse the stability of the systems we prepared a distractor image data set. The only use of these images is to disturb the systems. These images are just seen as *wrong* results if they are returned as a result of the systems. For this task we downloaded images from Flickr. To simulate more than one image per object in the database, we defined a random number of them to be one object. This is done as additional challenge, since also for each temple the systems have more than one image. Some of these images are shown in Fig. 5.

As query images we prepared our own data set. Our objective is to test object recognition under difficult conditions. So we prepared the images of the dataset to show the objects from many different viewpoint and containing various amount of clutter in the fore- and back-ground. In Fig. 6 a short sequence of these images is shown. As we can see, in some images the object is covered by trees and other objects. Even if for the buildings the background reminds somehow stable (e.g. trees), we still have to struggle with seasonable changes like red autumn leafs or snow.

We performed the experiments in four steps. First the database only contained the "correct" temples, as we receive them from Wikipedia. This database is somehow clean, since no "wrong" objects disturb the systems. In the next three experiments, we increased stepwise the amount of distractor images.

8.2 Mean Average Precision

As performance measurement we used the mean average precision in short mAP. For this value, the rank of the correct object in a sorted result list is taken into account. In detail, let N be the number of retrieved proposed objects, then the average precision P_{ave} is:

$$P_{\text{ave}} = \frac{\sum_{r=1}^{N}(P(r) \times rel(r))}{N_{rD}} \qquad (5)$$

where r is the rank, $rel()$ a binary function on the relevance of a given rank, and $P(r)$ precision at a given cut-off rank:

$$rel(r) = \begin{cases} 1, & \text{if } r \text{ is relevant;} \\ 0, & \text{otherwise.} \end{cases}, P(r) = \frac{\sum_{i=1}^{r} rel(i)}{r} \qquad (6)$$

Fig. 5 Random example images of the distractor data set loaded from Flickr.

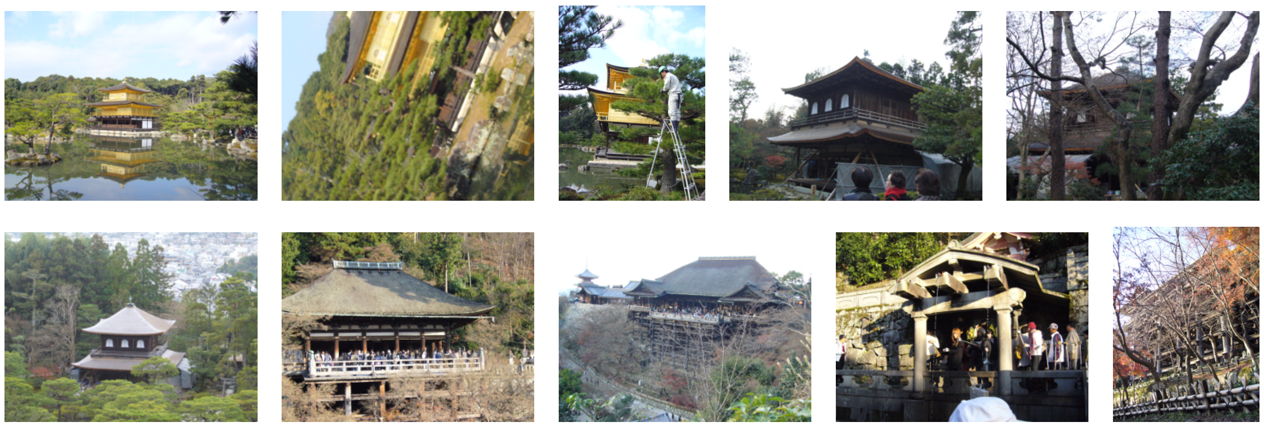


Fig. 6 Example query images from the temple data set. From left to right, top to bottom 3 images of Kinkaku-ji, 3 images of Ginkaku-ji and 4 images of Kiyomizu-dera are shown.

We get the mean average precision by finally taking the mean over all queries.

This value is more reasonable. In previous work [9] we counted the number of correct results at the first rank and within the top 10. Such a counting makes it hard to keep the overview or judge the results. However, still the better the rank, the higher is the mAP. If the correct object is at the first rank, we have a mAP of 1, at the second $\frac{1}{2}$, then $\frac{1}{3}$, $\frac{1}{4}$ and so on.

8.3 Results and Discussion

In Tbl. 2 we can see the results of our approach compared to a simple voting strategy by only using the PCA-SIFT features. For the discussion we split the results for the different objects. As we see, our proposed method can increase the recognition performance for all objects. Remarkable are the results for Kinkaku-ji, where the improvement is over 13%. For Kiyomizu-dera the improvement is only less. This may be due to the less uniqueness in the shape of this temple. Of course, below the temple is a characteristic structure but this is only visible in a few images.

Interesting are also the results for Ginkaku-ji. Here we can see that the recognition performance decreases with an increasing amount of distractor images in the database, but on the other hand, the improvement of our method compared to the simple scoring approach increases. In a clean database without any distractor

Tbl. 1 Computational cost of the three main steps of our proposed method. Feature extraction refers to the cost of the feature calculation, feature search to the cost of searching nearest neighbors and clustering to the cost of clustering the proposed reference points.

	time [sec.]
feature extraction	0.71
feature search	0.13
clustering	0.10
total	0.94

images, the improvement is only a bit more than 4% while for 50,000 distractor images the improvement is over 7%.

Overall our proposed method can always increase the recognition performance. However, that drop even after 2,500 distractor images is significant. Here our chosen way to disturb the system is maybe too hard. Additionally as we can see from the images in Fig. 6, many query images do not show the object perfectly. Often only small parts are visible or the image is taken from far away. Without distractor images, the results of the systems are unstable, i.e., only a few votes separate the correct object from a incorrect. If only a few votes are missing, as it happens after we add the 2,500 distractor images, the correct results descend in the ranking.

The weak point is still to find near a neighbor based on the descriptor part of the PCA-SIFT vector. To keep the computational cost limited, approximations like the

Tbl. 2 Results for the temple data set. All values are in percentage. Shown is the mean average precision for the simple voting strategy (Vote) and our proposed method (RP). The table includes the results for a clean database without distractor images, 2,500, 10,000 and 50,000 distractor images.

	no distractor images		2.5k distractor images		10k distractor images		50k distractor images	
	Vote	RP	Vote	RP	Vote	RP	Vote	RP
Ginkaku-ji	31.28	35.30	24.59	27.83	22.48	24.02	13.01	20.48
Kinkaku-ji	37.40	50.47	26.26	37.02	19.43	26.86	21.68	22.03
Kiyomizu-dera	17.15	18.79	12.70	13.66	10.49	10.91	9.75	9.86

hashing approach should be applied. If the system fails in this step to provide the features of the correct object, every following steps become meaningless. In this evaluation our main purpose was the analysis of the stability under very hard conditions. With changed parameters of the hash table we are able to improve the detection performance with the drawback of an increased recognition time. The clustering of the proposed reference points can be implemented very efficiently. The overview of computational cost is given in Tbl. 1. We can see that they are mainly coming from the calculation of the PCA-SIFT features. The cost for the nearest neighbor search and the clustering of the proposed reference points are minor.

Due to the nature of this dataset we still can have some correct matching based on the background like trees, if they occur in an image stored in the database and the query image. However, with the help of this simple reference point this effect is mainly limited to the correct object, while for incorrect objects these matches are discarded due to their geometrical structure. We achieve an significantly increased recognition performance on a database without distractor images and even for high disturbed databases the improvement is still noticeable.

9. Conclusion

In this paper we have addressed the common problem of background clutter in images for object recognition. This background clutter comprises, for example, trees or persons, which do not directly belong to the object. Our idea is to hold information about the local configuration of SIFT like features. For this task we keep the orientation, scale and position of these features in the image. During the recognition phase, we calculate a reference point from matching feature. This reference point is used to verify the local configuration of these matches. Matches with too different configurations are discarded. With our approach we are able to increase the recognition performance by more then 6% even under hard conditions. Our proposed method learns the shape even from training images containing a high amount of background clutter. This is the major advantage to other models which need a segment object.

Further research will focus on better setting of parameters for the reference point. Also additional information about local configuration to the next surrounding features will be investigated. The results of a even further increased database of one million distractor images will be analysed.

Acknowledgment

This work was supported in part by the Grant-in-Aid for Scientific Research (B)(19300062) from Japan Society for the Promotion of Science (JSPS).

References

[1] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories.," Workshop on Generative-Model Based Vision, 2004.

[2] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol.2, pp.264–271, June 2003. http://www.robots.ox.ac.uk/ vgg

[3] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," ECCV workshop on statistical learning in computer vision, pp.17–32, 2004.

[4] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," Int. J. Comput. Vision, vol.77, no.1-3, pp.259–289, 2008.

[5] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," ECCV, pp.304–317, 2008.

[6] D.G. Lowe, "Object recognition from local scale-invariant features," ICCV '99, p.1150, 1999.

[7] Y.K. Rahul, Y. Ke, and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," Proc. of IEEE CVPR, pp.506–513, 2004.

[8] K. Kise, K. Noguchi, and M. Iwamura, "Robust and efficient recognition of low-quality images by cascaded recognizers with massive local features," Proc. of WS-LAVD2009, pp.2125–2132, Oct. 2009.

[9] M. Klinkigt, K. Kise, H. Maus, and A. Dengel, "Object detection in images with cluttered background by using local features and their configuration," IEICE Tech. Report, PRMU2009-65, pp.75–80, Aug. 2009.