IPSJ Transactions on Computer Vision and Applications Vol. 3 110–121 (Dec. 2011)

Research Paper

Using a Reference Point for Local Configuration of SIFT-like Features for Object Recognition with Serious Background Clutter

Martin Klinkigt[†1,†2] and Koichi Kise[†1]

Object recognition can be performed on local or global features. While local features are more robust against occlusions, global features are more powerful to distinguish among many objects. In this paper we propose a novel approach in construction of a shape model from local features aimed at achieving high discriminative power as global features have, while keeping the robustness of local features. We utilize a common reference point expressing the relative position of local features like in a star graph representation. This model is dynamically calculated during recognition which makes it flexible. With our approach we achieve an improved recognition performance of 2% compared to other shape models and even 6% compared to approaches that do not utilize shape information.

1. Introduction

Object recognition in computer vision has gained more and more interest. It is not surprising that this research has split into many different fields of more theoretical or practical research. However, the process to learn and recognize objects is mainly based on images. During learning information of the image is extracted and compared with information of the query image in which a certain object should be recognized.

In recent research for the task of content description in an image, local features are used. Here information is described only in a tiny part of the whole image which can be seen as a peephole. This procedure is often done at the so-called region of interest. The major benefit of this approach is its stability. Even if parts of the object are covered by other objects, the same information can be extracted from the remaining visible parts. This approach also has a drawback. By only looking at such small regions, the discriminative power is lower as compared to an approach which takes into account more information of the image.

In this paper we present an approach which has the stability of local features and increases the discriminative power. We achieve this by providing a shape model about the features and how they are arranged in the image. Our shape model is based on a common reference point constructing a star graph representation of the local features. By using the distance and the angle of the local features to this reference point, this star graph is verified during recognition. Our representation of the shape is *soft* in the sense that occlusions do not become a major problem. We achieved an improved performance of around 2% on a challenging image dataset compared to other recently proposed shape models and up to 6% compared to approaches only relying on local features without utilizing any shape information.

2. Related Work

Object recognition and detection based on shapes is an often applied approach. Some approaches use boundaries to define a shape. The approach of Toshev et al. [1] is one of the most recent. The general strategy is to learn a vocabulary of boundaries from a set of training images. The procedure to extract these boundaries can be pixel based [11] or as in the approach of Toshev et al. superpixel based. Superpixels are connected regions normally defined over similar color information. Pixel based approaches often tend to recognize many different objects in very structured regions. With superpixels this does not happen, since such structured regions can be detected. However, they are still not scale nor orientation invariant. These two properties are needed for many realistic use cases. The difficulty lies in a proper model of boundaries to make them scale invariant.

Working on local features rather than boundaries we can avoid the above mentioned problems of scale and orientation invariance. The constellation model proposed by Fei-Fei et al. [2] uses around 5 features which are most characteristic for the object and creates a connected graph from them. To achieve scale invariance one feature is selected as “landmark” and based on its spatial dimension the graph is normalized.

†1 Osaka Prefecture University
†2 German Research Center for Artificial Intelligence

Leibe et al. are using with their implicit shape model [9] also on local features. In contrast to Fei-Fei, they use the relative position of the features to a common reference point called the *centroid*. With this approach, Leibe et al. proposed a very flexible shape model.

However, we still have some unsolved problems. The constellation model is only applicable for training images with a well-segmented object from the background while the implicit shape model still needs a bounding box. Both requirements will be impractical in many use cases. Also, these systems are rather complex and require sophisticated environments such as Matlab.

The method we propose is of different nature. Systems based on SIFT-like features can be easily extended to use our proposed model. It is inspired by the implicit shape model and also uses one common reference point to define a shape for all features. In contrast to the implicit shape model, our system does not create this shape during training. Our utilization of such a reference point takes place during recognition. This results in the high flexibility of our system and a very simple implementation.

A recent approach which works on SIFT-like features is the weak geometric consistency (WGC) proposed by Jégou et al. [4] Other than our approach, shape information is verified based only on each individual feature without taking into account their relation to each other. This model is really simple and thus it is only applicable under "clean" conditions, e.g., on smaller databases and a low involvement of background clutter as we will point out with our evaluations.

3. Overview

With this section we give a rough overview of our system while focusing on the idea behind and the necessary elements needed for our proposed method.

Designing an object recognition system, the used features can be of local type by only looking at a small part of the image, or of global type by taking into account the whole image at once. While local features can address the problem of partially covered objects, global features have a higher discriminative power. Achieving both with one descriptor is not obvious. The goal of a highly discriminative descriptor stands in contrast with a flexible descriptor addressing occlusions.

A more discriminative descriptor can be achieved by being stricter concerning local features. However, slight changes in the illumination may result in an unrecognizable object, since the descriptor is too strict. A different and often applied way is to take into account more information of the image at once which results in global descriptors. A problem with these global descriptors is that occlusion can only be addressed in a limited manner.

In our model we are aiming to achieve both goals: addressing occlusions and achieving a higher discriminative power. We utilize a special type of local feature, providing an orientation and scale about this region. By using these properties we express the position of the features with respect to one common point comparable to a star-graph representation. This results in our global descriptor, the shape. This shape is verified during recognition and fragmented shapes are ignored. If some of the local features are missing due to occlusion, the remaining features still form a proper shape.

The further details in this section explain the basic concepts needed for our method. This includes an explanation about the local features and the properties needed for our shape model. After that we explain the voting scheme in object recognition. This voting scheme requires nearest neighbor search which can become expensive on large databases. With hashing we reduce these search costs significantly. The search space can be controlled by arranging the features into classes also called bins. For the query feature the corresponding bin is determined and only within this bin, possible nearest neighbors are searched.

3.1 SIFT Features

The Scale-Invariant Feature Transformation (SIFT) proposed by Lowe [10] is one example of local features. It utilizes a 128 dimensional feature descriptor vector and for its entries mathematical values like derivations are used. This descriptor has been used successfully in many fields in the current research of object recognition.

PCA-SIFT [6] uses an even higher feature descriptor (3,042 dimensions) and compresses these values by Principal Components Analysis [5] to the most significant eigenvalues. As Rahul et al. have also shown, the recognition performance can be even higher and most importantly, the used memory to store these features is reduced significantly. This is an important condition, since we apply object recognition via a voting scheme, which involves nearest neighbor search

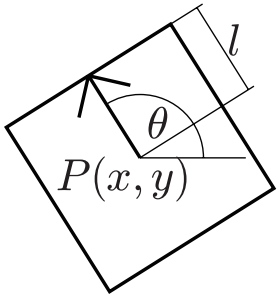


Fig. 1 SIFT feature.

and, therefore, features must be kept in memory. We explain this strategy in the next section.

These PCA-SIFT features are computed for regions of interest (ROI) in a certain location in the image. These types of features additionally provide the scale and orientation of the ROI as shown in **Fig. 1** where l is the scale, θ is the orientation and the position of the feature in the image is $P(x, y)$.

3.2 Object Recognition by Voting

For object recognition by voting normally local features are used to represent an image. 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 more votes an object has, the higher is the confidence for the object.

3.3 Hashing

A major problem with object recognition by voting is to find similar 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 large, while the number g can easily become several millions or more.

Hashing is an approach which reduces the complexity remarkably. A hash table consists of many bins which are accessed 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 2 bit 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

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

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.

4. Reference Point

After we discussed the advantages and disadvantages of the different features we can derive some properties a model should hold to describe the local configuration of features:

(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 and a reasonable runtime, the last one is to focus on

the problems some models like the WGC can have, even if 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.

4.1 Learning Phase

As mentioned earlier our implementation is based on PCA-SIFT features. However, any descriptor could be used which provides a scale l and orientation θ about the local region. These properties are shown in Fig. 1. The additional memory needed to store these values is marginal compared to the 36 dimensions for the PCA-SIFT descriptor. Additional computational cost does not occur, since these values are also needed to calculate the descriptor itself. We just utilize them. For all features we calculate its hash value as described in Section 3.3.

4.2 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 neighbors in the corresponding bin. Near features have to fulfill a certain threshold. The simple voting approach to which we compare our system, now cast a vote for this object.

At this point we create our shape model. We assume to have all possible matches between query features and features from the database. Let F_D be this set of matched features for one image I in the database. For this set 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:

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

From this reference point we calculate for every matched feature in the database image two new values shown in **Fig. 2**. These are the distance t of the feature to this reference point RP and the enclosed angle α.

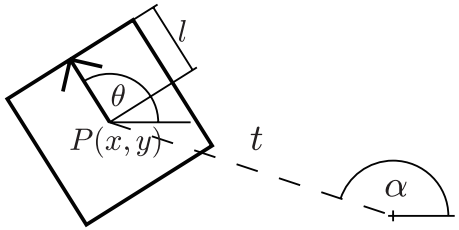
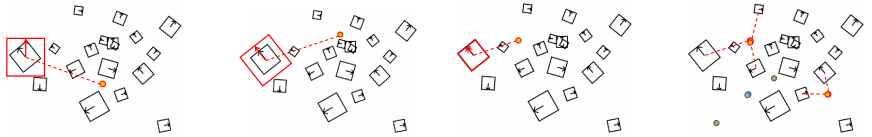


Fig. 2 Additionally calculated properties.



(a) Placing matching features. (b) Adapt the orientation. (c) Adapt scale. (d) Resulting reference points of all matches.

Fig. 3 Steps of reference point matching.

This selection of the reference point has some advantages concerning numerical errors, e.g., when the distance t to the reference point becomes large, small errors in the orientation θ will lead to misplaced reference points. However, any arbitrary selection of the reference point is possible.

After we have these two new values we proceed further with the query image. **Figure 3** illustrates these steps. Indicated are the features from the query image and a matching feature attached with the new properties.

(a) We place the matched feature from the database over the feature from the query image.
(b) We adjust 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 adjust 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) Performing these steps for all matches for all objects results in a map of reference point proposals.

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 objects 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 regions which will be described in the next section.

5. Clustering

Clustering of the proposed reference points is necessary to find dense regions. The k-means clustering algorithm which is often applied for this task, is not suited for our use-case since the k is nearly impossible to select automatically. Fortunately the number of reference points is not high so that we can choose an algorithm which can determine useful clusters without the need to provide a k.

In the standard clustering algorithm one would have to calculate for each cluster its nearest neighbors. If the distance of this nearest neighbor is below a certain threshold, the clusters are merged. In the next iteration again for each cluster its nearest neighbor must be calculated, since the merging of the two clusters one step before affects the distance among the clusters. This makes the application of the standard clustering algorithm a time-consuming task.

For the purpose of a faster calculation of these clusters we choose the *RNN algorithm for Average-Link clustering with nearest-neighbor chains* [9]. The idea of this algorithm is to select a random start point and search for this point its nearest neighbor. From this nearest neighbor again its nearest neighbor is searched by ignoring the former point. The idea of keeping all these nearest neighbors in a chain gives this algorithm its name *reciprocal nearest neighbor* (RNN) chain. If the next nearest neighbor is beyond the threshold the chain is canceled. All points within the chain are merged and the process starts in a new random seed. The major speed-up of this algorithm results from keeping the nearest neighbors in this chain instead of leaving this information behind.

A complete proof of the correctness of this strategy could not be given in this paper but in short it relies on Bruynooghe's reducibility property which means that before the agglomeration of two clusters and after, the nearest neighbors are still the same. Only the clusters might be closer to each other. This algorithm is summarized in the pseudo code in **Fig. 4**.

```
last ← 0
lastsim[0] ← 0
L[last] ← v ∈ V
R ← V\v
while R ≠ ∅ do
  (s, sim) ← getNearestNeighbor(L[last], R)
  if sim > lastsim[last] then
    last ← last + 1
    L[last] ← s
    R ← R\ {s}
    lastsim[last] ← sim
  else
    if lastsim[last] > t then
      s ← agglomerate(L[last], L[last − 1])
      R ← R ∪ {s}
      last ← last − 2
    else
      last ← −1
    end if
  end if
  if last < 0 then
    last ← last + 1
    L[last] ← v ∈ R
    R ← R\ {v}
  end if
end while
```

Fig. 4 The RNN algorithm for Average-Link clustering with nearest-neighbor chains.

For the clustering we make one slight adaption. We do not treat all clusters 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. 3 (c)) of the feature in the query image to RP' after the scale was adapted. 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.

In general the database can contain more than one image per object. However, clustering is always done for reference points from one image. 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 be normalized, since the object may be at different locations in different images. Currently we do not consider such a normalization.

6. Voting

After we have performed the clustering on the proposed reference points, the question raises how this information can be used to improve 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 individual points in each cluster usually increases. If many reference points could be agglomerated in one 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 remaining clusters containing more than one point, we calculate a score for the corresponding object. This is done by taking the number of containing reference points in a cluster.

We also considered a voting based on the feature size or a normalization based on the resulting area covered by the clusters. However, we were not able to improve the overall performance. Special voting based on the features could only improve the performance in some special cases. As the final score for the object, we take the highest achieved score from one image of this object. At the end we apply a normalization so that the scores lie within the interval $[0, 1]$, where 1 is then the highest value.

for $o \in O$ **do**
  for $f \in W(o, f) \neq 0$ **do**
    $S(o) += W(o, f)$
  end for
end for

(a) Sliding window

for $f \in F$ **do**
  for $o \in W(o, f) \neq 0$ **do**
    $S(o) += W(o, f)$
  end for
end for

(b) Hough transform

Fig. 5 Comparison of sliding window and generalized Hough transform.

7. Localization

Sometimes the localisation of the object of interest is request. Here the object is marked in the image to give the user additional feedback from the recognition system. To formalize the problem of localization let $o \in O$ be the set of trained objects, let $f \in F$ be the set of features extracted from the query image, $W(o, f)$ a scoring function defining a weight for a certain object given a feature and finally let $S(o)$ be the score for an object o.

7.1 Sliding Window vs. Generalized Hough Transform

To achieve the localization of the object of interest sliding window is a typically applied approach. The burden of this approach is its enormous calculation overhead. A different approach is the generalized Hough Transform to which our system has some similarities.

Figure 5 compares these two approaches in pseudo code. On the left hand side (Fig. 5 (a)) we see the sliding windows approach while on the right hand side (Fig. 5 (b)) the generalized Hough transform is summarized.

For the sliding window approach, a window smaller than the image is defined and in its area the object will be searched. If the confidence for the object is high, the region is often marked as occupied to prevent the system from searching for other objects in this region. Since the system normally has no prior information about the location of an object, this window is moved over certain places with changing size all over the image. This results in high computational costs and an often underestimated overhead, since for certain locations in the image the same calculations must be performed only for different windows.

The Generalized Hough Transform processes the whole image only once. The

overall idea is to find the object by voting for its center of gravity. In the case of a circle, this would mean that all points on the circle vote for the center-point. Since again the system does not have any prior information this involves a voting for all possible circles through this point. For often voted parameter setting the Hough space will show dense regions. These dense regions are a hint for correct settings of parameters and, therefore, a possible object. In the Generalized Hough Transform the parameters for voting are read from some kind of look-up table. In this table for all point locations in the image the corresponding votes in the Hough space are stored. With such a table any shape can be defined.

These two approaches look similar and in fact only the two loops in Fig. 5 at the beginning are interchanged. These approaches are just two sides of the same coin as Lehmann et al. [8] have shown. The reason that they are looking so different is only due to the fact of different voting strategies. While the sliding window approach wants to verify a hypothesis for a certain object in the image, the Hough transform analyse for which objects the information in the image could vote.

7.2 Proposed Method

Since our model has similarities to the Hough transform we can apply the same localization approach, which we shortly explain in this section. For our explanation we use **Fig. 6** which is an example query image of our evaluation.

In the first figure (Fig. 6 (a)) we indicated with holes in the building matching features of the correct object. Indicated are only the features for which the correct object has the closest match. For the voting strategy only these features would be taken into account. As we see, these features are concentrated on a smaller, more unique part of the object. For the localization with the reference point approach we also use matches of lower evidence. By doing so we want to achieve two goals. First, we have to challenge with numerical problems and by only using less features clustering would become fragile. Since incorrect matches will be eliminated in the shape verification, clustering can only benefit from using more features. Second, with only less features a meaningful localization of the object is hardly possible. The object may be cut into different parts.

Figure 6 (c) shows in striped regions the resulting proposed reference points. The larger the scale of these striped regions, the more proposed reference points could be clustered. As we can see these points do not necessarily have to be

(a) Correct matching nearest neighbor features.

(b) Final segmented object.

(c) Resulting proposed reference points and their clustering.

Fig. 6 Localization of the object.

placed over the object, which exemplary shows the higher generality of our model. The image in Fig. 6 (b) shows the segmented object. For the calculation of this image, we applied the segmentation algorithm of Felzenszwalb et al. [3] and blank out regions for which the extracted features have no similarity with any features from the database or have inconsistent matching in the reference points. The algorithm of Felzenszwalb et al. calculates such segments based on the color information of the image. With this algorithm one can control the size of the resulting patches to avoid too small regions as they would result from highly structured regions, e.g., trees.

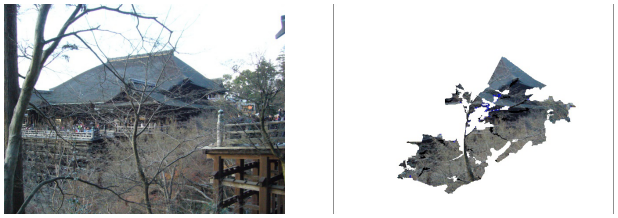


Fig. 7 Misleading object segmentation.

In the case of a reasonable visible object this works quite satisfyingly. If the object gets more and more covered, the object and the background become one, and the results of the segmentation algorithm are not accurate enough. Therefore, the localization also includes parts of the background as we can see in the example on the right side of **Fig. 7**.

For the Hough transform normally no prior knowledge about the shape is given. Therefore, for every point in the image plane multiple votes in the Hough space are generated. This corresponds with the second iteration as shown in the algorithm in Fig. 5 (b). To keep this multiple voting in one point below a reasonable limit, Leibe et al. require such a bounding box.

Our proposed method does not require such a bounding box since it has a different voting strategy. Our system calculates something like "prior" knowledge as explained in Section 4.2 with Fig. 2. From this knowledge we directly calculate *one* vote. Thus in the sense of algorithm in Fig. 5 (b) we do not have the second for-loop. This makes our model faster and release it from the need of a bounding box.

8. Experiments and Discussions

In our experiment we analysed the performance of a simple voting strategy as in Ref. 7), the weak geometric consistency proposed by Jégou et al. [4] and our approach utilizing shape information with the help of the reference point. As summarized in Section 6, the voting strategy only takes into account the matching local features by ignoring shape information. The WGC is a state-of-the-art model and verifies some simple shape information like the scale and orientation of the local features. In detail we used:

Vote We roughly summarized the voting scheme in Section 3.2. In our utilization we search for each feature from the query image, the feature from database with the smallest distance which is at least below a certain threshold. The object from which this nearest feature is extracted receives one vote. The number of votes defines the order in the ranked list. The more votes, the better the rank is.

WGC The Weak Geometric Consistency proposed by Jegou et al. is one of the most recent approaches verifying shape information on the base of SIFT features. The shape information in this model is based only on the local features. From the matching features in the database and query image, the differences in scale and orientation are calculated. These differences are the same for all matches of the correct object, since the object can only be transformed homogeneously in all its parts. In this method, the object which agglomerates its matches for similar parameter settings is assumed to be the correct object.

These differences are quantized and stored into two histograms, one for the differences in scale and one for the differences in orientation. Let $g(\delta_{l_j})$ be the score in scale difference histogram bin δ_{l_j} and respectively $h(\delta_{\theta_j})$ the score in a bin of orientation difference histogram. The score s resulting from the matches is:

$$s = \min\left(\max_{\delta_{l_j}\in\delta_l} g(\delta_{l_j}), \max_{\delta_{\theta_j}\in\delta_\theta} h(\delta_{\theta_j})\right). \tag{5}$$

8.1 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:

$$P_{\text{ave}} = \frac{\sum_{r=1}^{N}(P(r)\times \text{rel}(r))}{N_{rD}} \tag{6}$$

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:

$$rel(r) = \begin{cases} 1, & \text{if } r \text{ is relevant;} \\ 0, & \text{otherwise.} \end{cases}, P(r) = \frac{\sum_{i=1}^{r} rel(i)}{r} \tag{7}$$

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

This value is more reasonable, since 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.

8.2 Public Dataset

We evaluated our method on the INRIA holiday dataset for which results are available for weak geometric consistency. This dataset consists of images 1,491 from which 991 are training images and the remaining 500 are query images. The mAP for the weak geometric consistency reported on this dataset is 61.16%. By using more prior knowledge, like the images are always up-right, the authors could push this value to 75.07%. However, we do not use such prior knowledge so this is not our competitor. For this dataset our system achieved a mAP of 71.83%.

However, this dataset is not fitting for our motivation. It was created to find similar images in the database and, therefore, the impact of background clutter is limited. To simulate such a higher impact, we used the following temple dataset.

8.3 Temple Dataset

As a task for the temple dataset we let the systems recognize temples and shrines from 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 database 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 an unexpected ranking of the objects.

Fig. 8 Random example images of the distractor data set loaded from Flickr.

Fig. 9 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.

To analyse the stability of the systems we prepared a distractor image dataset. The only use of these images is to disturb the systems. These images are seen as *incorrect* 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 used a random number of them and defined them to be one object. This is done as an additional challenge, since also for each temple the system have more than one image.

Some of these distractor images are shown in **Fig. 8**.

As query images we prepared our own data set. Our objective is to realize 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 background. In **Fig. 9** 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 remains stable (e.g., trees), we still have to struggle with seasonal changes like red autumn leafs or

Table 1 Results for the temple data set. All values are in percentage. Shown is the mean average precision for the simple voting strategy (Vote), the weak geometric consistency (WGC) and our proposed method (RP). The table shows the results for a clean database without distractor images.

	Vote	WGC	RP
Ginkaku-ji	31.28	32.86	35.30
Kinkaku-ji	37.40	41.99	50.47
Kiyomizu-dera	17.15	24.27	18.79
average	28.61	33.04	34.85

snow.

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

8.4 Results and Discussion

In **Table 1** we can see the results of our approach compared to a simple voting strategy by only using the PCA-SIFT features and the weak geometric consistency. 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 compared to the simple voting approach. Remarkable are the results for Kinkaku-ji, where the improvement is over 13%. The weak geometric consistency (WGC) can also increase the performance. For Kiyomizu-dera this improvement is remarkable while our proposed approach can hardly achieve an improvement for this object. For this object our proposed method suffers from less matching features from the nearest neighbor search. When the number of available matching features is low, the space of proposed reference points is sparse and numerical problems can not be addressed easily and the reference points can hardly be clustered. The WGC has not to struggle from such numerical problems, if only a few matches are available. Since it is working with quantized bins these numerical errors do not effect the voting. From Table 1 we can also conclude an averaged improved performance of our proposed method of around 6% for each object compared to the simple voting approach and around 2% compared to the WGC.

The results for the database with distractor images are given in **Fig. 10**. Compared to the simple voting approach, the reference point improves the recognition performance even under such conditions. From this figure we can also see a significant drop for all approaches in the region of around 10,000 additional distractor images. For the WGC already after 2,500 distractor images, the first improved performance nearly does not exists anymore. After 10,000 distractor images the WGC has a lower recognition rate as the simple voting, while our proposed method still achieves a significant improvement. This behaviour of the WGC results from its only *weak* verification. With such a high amount of distractor images, the global histograms get disturbed and after the normalization no meaningful separation among the objects is possible since the scores are nearly equal.

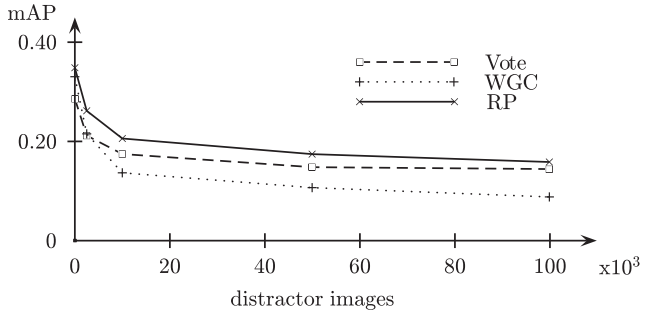


Fig. 10 Results for with distractor images increased database.

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. 9, 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 are unstable, i.e., only a few votes separate the correct object from an 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. Despite from this fact, our proposed method is still capable of improving the recognition performance as compared to the WGC which even has a lower recognition rate than the simple voting.

Table 2 Computational cost of the three main steps of the compared methods. Feature extraction refers to the cost of the PCA-SIFT feature extraction, feature search to the cost of finding suited nearest neighbors and shape calculation to the cost of calculation of the WGC or the reference point of our proposed method (PM).

	voting	WGC	PM
feature extraction		0.71	
feature search		0.13	
shape calculation		0.08	0.10
total	0.84	0.92	0.94

An overview of computational cost of the proposed method is given in **Table 2**. We can see that the main computational cost resulting from the calculation of the PCA-SIFT features. The costs for the nearest neighbor search and the shape calculation are minor.

The weak point is still to find a near neighbor based on the descriptor part of the PCA-SIFT vector. To keep the computational cost limited, approximations like the hashing approach should be applied. If the system fails in this step to provide the features of the correct object, every following step becomes 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 performance with the drawback of an increased recognition time. Different classifiers as for example support vector machines, are not suited for our task. By using SVM a shape construction during recognition becomes difficult. It would be possible to learn the shape during training which results in the requirements of a segmented object.

With the proposed method we can improve the recognition performance on a database without distractor images significantly by overall 6% and for some objects even improvements of more than 13% are achieved. If only a few matching features are available, e.g., in the case that the object of interest is only partial visible, the shape reconstruction fails and the performance is lower as compared to the WGC. However, for a database with distractor images the improvement is still noticeable and does not result in a reduced performance as compared to the WGC.

9. Conclusion

In this paper we have proposed a novel approach in calculation of a shape model from local features to obtain the stability of local features and the power to distinguish between many objects of global features. For this task we utilize a reference point which holds the relative position of the local SIFT features. The model is created and verified during recognition which results in a reduced calculation time compared to the generalized Hough transform. With our approach we are able to increase the recognition performance on a small database by 2% and for a large database even up to 6% compared to the WGC which has even a lower performance than a simple voting approach.

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 an even further increased database of one million distractor images will be analysed.

References

1) Toshev, A., Taskar, B. and Daniilidis, K.: Object Detection via Boundary Structure Segmentation, *CVPR* (2010).
2) Fei-Fei, L., Fergus, R. and Perona, P.: Learning generative visual models from few training examples, *Workshop on Generative-Model Based Vision, IEEE Proc. CVPR* (2004).
3) Felzenszwalb, P.F. and Huttenlocher, D.P.: Efficient Graph-Based Image Segmentation, *Int. J. Comput. Vision*, Vol.59, No.2, pp.167–181 (2004).
4) Jégou, H., Douze, M. and Schmid, C.: Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search, *Proc. ECCV*, pp.304–317 (2008).
5) Jolliffe, I.: *Principal Component Analysis*, Springer, New York (2010).
6) Ke, Y. and Sukthankar, R.: PCA-SIFT: A More Distinctive Representation for Local Image Descriptors, *Proc. IEEE CVPR*, pp.506–513 (2004).
7) Kise, K., Noguchi, K. and Iwamura, M.: Robust and Efficient Recognition of Low-quality Images by Cascaded Recognizers with Massive Local Features, *Proc. WS-LAVD2009*, pp.2125–2132 (2009).
8) Lehmann, A., Leibe, B., and Gool, L.V.: PRISM: PRincipled Implicit Shape Model, *British Machine Vision Conference* (*BMVC*) (2009).
9) Leibe, B., Leonardis, A. and Schiele, B.: Combined Object Categorization and Segmentation With An Implicit Shape Model, *ECCV Workshop on Statistical Learning in Computer Vision*, pp.17–32 (2004).

10) Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints, *Int. J. Comput. Vision*, Vol.60, pp.91–110 (2004).
11) Opelt, A., Pinz, A. and Zisserman, A.: A Boundary-Fragment-Model for Object Detection, *Proc. ECCV* (2006).

(Received November 10, 2010)
(Accepted May 17, 2011)
(Released December 28, 2011)

(Communicated by *Keiji Yanai*)

Martin Klinkigt was born in 1983. He received his M.Sc. degree in computer science from Technical University of Kaiserslautern, Germany in 2009. Now he is a Ph.D. student of the Graduate School of Engineering, Department of Computer Science and Intelligent Systems, Osaka Prefecture University. Since 2007 he is research assistant at the German Research Center for Artificial Intelligence (DFKI), Germany. His research interests are related with Semantic Web and object recognition and under unconstrained conditions.

Koichi Kise received his B.E., M.E., and Ph.D. degrees in communication engineering from Osaka University, Osaka, Japan, in 1986, 1988 and 1991, respectively. From 2000 to 2001, he was a visiting professor at German Research Center for Artificial Intelligence (DFKI), Germany. He is now a Professor of the Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Japan. His research interests include object recognition, document analysis and retrieval. He received the best paper award from IEICE in 2008, the IAPR/ICDAR best paper award of ICDAR2007, the IAPR Nakano award of DAS2010, and the best paper award of ICFHR2010. He is now the vice chair of IAPR TC11 (Reading Systems), IAPR Conference & Meetings Committee. He is a member of scientific societies including IEICE, IPSJ, IEEE and ACM.