HPAT indexing for fast object/scene recognition based on local appearance

Hao Shao[1], Tomáš Svoboda[1], Tinne Tuytelaars[2] and Luc Van Gool[1,2]

[1]Computer Vision Lab, Swiss Federal Institute of Technology, Zürich, Switzerland
{haoshao,svoboda,vangool}@vision.ee.ethz.ch
[2]ESAT, PSI, Katholieke Universiteit Leuven, Leuven, Belgium
{tinne.tuytelaars,luc.vangool}@esat.kuleuven.ac.be

Abstract. The paper describes a fast system for appearance based image recognition . It uses local invariant descriptors and efficient nearest neighbor search. First, local affine invariant regions are found nested at multiscale intensity extremas. These regions are characterized by nine generalized color moment invariants. An efficient novel method called HPAT (hyper-polyhedron with adaptive threshold) is introduced for efficient localization of the nearest neighbor in feature space.
The invariants make the method robust against changing illumination and viewpoint. The locality helps to resolve occlusions. The proposed indexing method overcomes the drawbacks of most binary tree-like indexing techniques, namely the high complexity in high dimensional data sets and the boundary problem. The database representation is very compact and the retrieval close to realtime on a standard PC. The performance of the proposed method is demonstrated on a public database containing 1005 images of urban scenes. Experiments with an image database containing objects are also presented.

1 Introduction

Most content-based image retrieval systems focus on the overall, qualitative similarity of scenes. Hence, global aspects like color gamuts and coarse spatial layout are among the features most often used. This paper deals with a somewhat different kind of retrieval, in the sense that images are sought that contain the same, prominent objects as the query image. One could e.g. look for images on the Internet with the same statue as contained in the query image, or the system could recognize one's location, by taking an image and looking for the most similar image in a large database of images taken all over a city. The latter type of application is given as an example further in the paper.

For the rather stringent type of similarity search propounded here, global and qualitative features no longer suffice. Hence, we propose the use of local color patches as features, which are compared rather precisely. In particular, we propose to extract so-called invariant regions, which have become popular recently [1, 5, 2, 7]. Here, we use the intensity-based regions proposed by Tuytelaars and Van Gool [7]. Invariant regions correspond to small image patches, constructed around special seed points (here intensity extrema). These regions are special in that they automatically adapt their shapes in the image to the viewpoint of the camera. The crux of the matter is that this adaptation takes place without any knowledge about the viewpoint and without any comparison with other images in which the same regions are visible. Thus, in principle, the same physical parts are extracted from two images of the same scene, even if these have been taken from different viewpoints (and possibly under different illumination as well). In section 2.1, we introduce a multiscale approach that improves the repeatability of these regions. Once such regions have been extracted, the color pattern that they enclose is described on the basis of invariant features. The feature vectors make it possible to match invariant regions very efficiently, using hashing techniques. Such matching lies at the heart of our approach, since the similarity between images is quantified as the number of matching features.

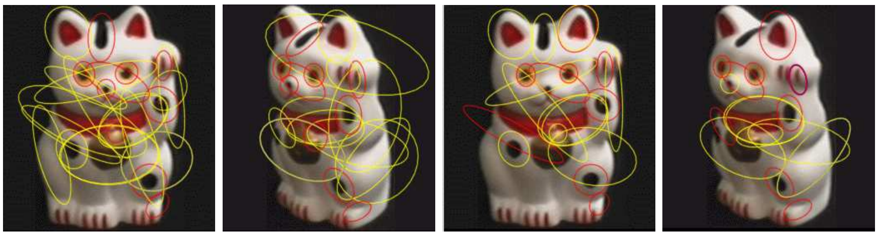
However, since one image is represented by a large set of local invariant regions, the amount of data in the feature space is much higher than for global methods. As a result, feature matching becomes more complex and is usually performed in a high-dimensional space. Under these conditions, finding the nearest neighbor can become quite time consuming. Nene and Nayar proposed an efficient method to overcome this problem [4]. We contribute to this method by proposing a hyper-polyhedron instead of a hyper-cube as a better approximation of the hyper-sphere. The hyper-polyhedron approximation reduces the number of feature vectors to be searched for the nearest neighbor, thus saving computation time.

The structure of the paper is as follows. Section 2 describes the overall system, with special attention to the way in which more stable invariant regions are extracted using a multiscale approach. Section 3 discusses the indexing structure based on the HPAT idea. Finally, section 4 demonstrates experiments on several public databases. Section 5 concludes the paper.

2 Retrieving Images

2.1 A more stable invariant region extractor

In [7], Tuytelaars and Van Gool describe a practical method to extract local invariant features. More precisely, they extract elliptical regions constructed around intensity extrema. Here, we present a slight modification to this method, which from our experiments seems to significantly improve the stability or repeatability of the method. More precisely, we select the intensity extrema used as seed points in a more stable way. To this end, we apply a Gaussian scale space. This can be constructed very efficiently thanks to the separability of the Gaussian kernel. We use five different scale level representations (the original image, plus four smoothed versions with a scale factor of $1.2, 1.4, 1.8$ and 2.2). Then, a non-maximum suppression algorithm finds the local intensity extrema at each scale level, as in the original method. However, only those local extrema which are repeated at different scales are considered as stable and are used to extract invariant regions. Fig. 1 shows an example of the regions extracted with



(a) (b) (c) (d)

Fig. 1. Two different views of the same object, with some invariant regions overlayed (red: regions that have been extracted in both images, yellow: regions that have been extracted in only one image). Note how the red regions cover the same physical parts of the scene. **(a, b)** regions extracted with the method of [7], **(c,d)** regions extracted with more stable intensity extrema.

the proposed method as well as those extracted with the original method. In total, the improved method extracts 20 and 16 regions on the two views, with 8 of them being repeated, while the original method extracts 26 and 21 regions, with the same number of repeated regions. The repeatability has clearly improved, which not only saves computation time during the retrieval process but also may improve the recognition accuracy.

One could argue that the extracted intensity extrema are not really invariant under affine transformations, due to the isotropic Gaussian smoothing. However, intensity extrema that are found over several scales will also be found after an affine transformation with anisotropic scaling smaller or comparable to the scale change between the scale levels. Moreover, the method does not strictly rely on the affine-invariance of the seed points, as the next step of the region extraction method is robust to their inaccurate localization[7].

2.2 Matching invariant regions

Finding correspondences between two views is then performed by means of a nearest region classification scheme, based on feature vectors of invariants computed over the affine invariant image regions. As in the region extraction step, we consider invariance both under affine geometric changes and linear photometric changes, with different offsets and different scale factors for each of the three color bands. More precisely, each region is described by a set of nine generalized color moment invariants, as described in [3].

The Mahalanobis distance is a good measure to compare such feature vectors, as it correctly takes into account the different variability of the elements of the feature vector as well as their correlation. During the offline construction of the database, the original space is rotated based on the covariance matrix, such that Mahalanobis distance is reduced to Euclidean distance.

2.3 Retrieving images

The actual recognition of objects or scenes then goes as follows. First, we build a database with representative images for all the known objects or scenes (with possibly more than one image per object or scene). Then, in the offline preprocessing, we extract invariant regions and invariant feature vectors for all the images in the database, compute the covariance matrix, and store the rotated feature vectors in the database as well, with pointers to the corresponding image. Then, during the online recognition phase, a query image is processed in exactly the same way (extraction of invariant regions, computation of feature vectors, and rotation according to the covariance matrix). For each feature vector, we look for the nearest neighbor in the database, using fast indexing methods as explained in section 3, and add a vote to the corresponding image. This way, images in the database showing the same object or scene as the query image will get a high number of votes. Simply ranking the database images based on the number of matches finishes the retrieval.

3 Indexing based on a hyper-polyhedron approximation of the hypersphere and with an adaptive threshold

Nene and Nayar [4] proposed an efficient algorithm for nearest neighbor search. It uses a hyper-cube as an approximation of the hyper-sphere. The algorithm begins with selecting the points that are in between a pair of parallel planes $X1$ and $X3$ perpendicular to the first coordinate axis (see Fig. 2) and adds them to a list, called the candidate list. Next, it trims the candidate list by discarding points that are not in between another pair of parallel planes $X2$ and $X4$, corresponding to the second dimension. This procedure is repeated for each dimension to end up with a hypercube of size $2r$ centered around the query point. Once we have this trimmed candidate list, the closest point is found using an exhaustive search. The method can be implemented very efficiently. However, the hyper-cube approximation of the hyper-sphere deteriorates very quickly with increasing dimensions. For instance, in case of 9 dimensions, the volume of the hyper-sphere is only 0.7% of the hyper-cube volume. This results in unnecessary computation during the exhaustive search phase. Here, we propose a more accurate approximation of the hyper-sphere based on a hyper-polyhedron.

3.1 Hyper-polyhedron

We explain the proposed hyper-polyhedron approximation in 2D space, although the ideas hold for higher dimensional spaces as well. As can be seen in Fig. 2, left, the point P_c is not within the circle even though it falls inside the square. For higher dimensions, there will be many more such points. As a result, using the hyper-cube method, the candidate list includes many useless corner points such as P_c. Instead of using two pairs of parallel planes, we propose to use four, see Fig. 3. The additional four planes are perpendicular to two new *lifted* coordinates

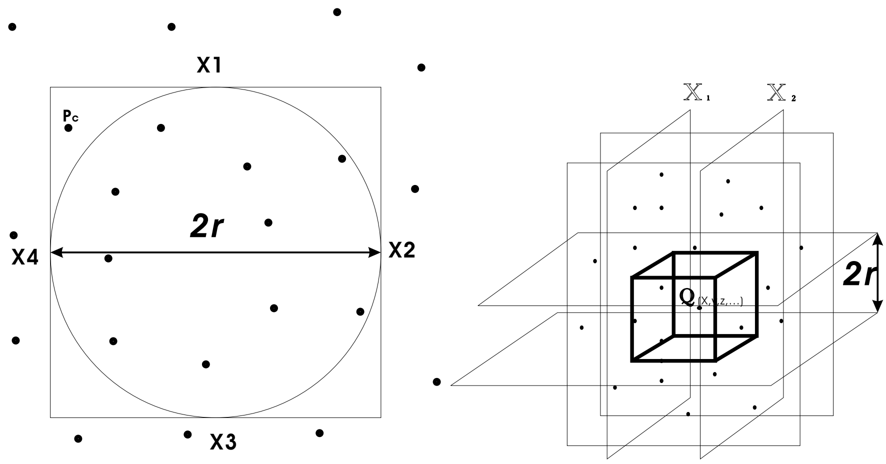


Fig. 2. Two and three dimensions case of Hypercube. The corner points like p_c fall within the hyper-cube but not within the hyper-sphere

x_{12} and x_{21}. These lifted coordinates are projections of the x_1, x_2 coordinates onto a rotated coordinate axes frame:

$$x_{12} = (x_1 + x_2)/\sqrt{2}\,, \text{ and } x_{21} = (x_1 - x_2)/\sqrt{2}\,. \tag{1}$$

In general, for the n-dimensional case we need $n(n-1)$ auxiliary dimensions. The lifted coordinates are used only for trimming the space, i.e., to construct the hyperpolyhedron. This phase is very efficient. Only the "true" n coordinates are used during the (time-consuming) exhaustive search.

It is clear that the hyper-polyhedron is a much better approximation of the hypersphere than the hypercube. The volume of the hypercube is given by

$$V_{hc} = (2r)^n \tag{2}$$

with r the radius of the inscribed hypershere, while the volume of the hyper-polyhedron is given by (see the appendix for a derivation of this formula)

$$V_{hp} = (2r)^n(\sqrt{2}-1)^n \left[1 + \sum_{k=1}^{n} \binom{n}{k} 2^{1-k/2}\right] \tag{3}$$

The volume of the hyperpolyhedron grows much slower with increasing number of dimensions than the hypercube, which grows exponentially. The volume of the hypersphere, on the other hand, remains more or less constant and even decreases for high dimensional spaces. For the 9-dimensional case, the hyper-polyhedron takes only 8.8% of the hypercube volume. Of course, constructing the hyper-polyhedron has its price: it needs $n(n-1)$ more candidate list trimmings (scalar comparisons) than the original hyper-cube method. In return, we save

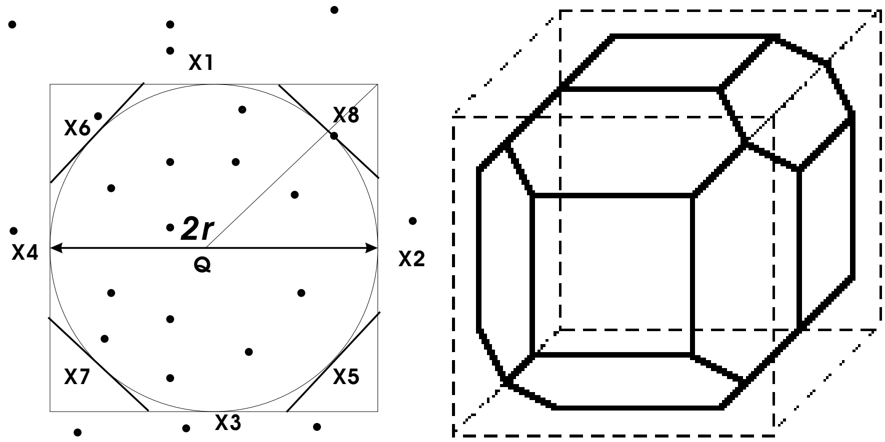


Fig. 3. Reduction the volume by using more planes in two dimensions and the illustration of three dimensions

$(1 - \frac{V_{hp}}{V_{hc}}) * m$ distance computations, with m the average number of feature vectors inside the hypercube after trimming. Approximately, our method is more efficient if

$$n(n-1) < \frac{m(1 - \frac{V_{hp}}{V_{hc}})\mathcal{O}_d(n)}{2\mathcal{O}_c} . \tag{4}$$

with $\mathcal{O}_d(n)$ the computational complexity of the distance computation and $\mathcal{O}_c$ complexity of the scalar comparisons. The ratio $\mathcal{O}_d(n)/\mathcal{O}_c$ varies on different computer platforms. Our experiments on PC showed that $\mathcal{O}_d(n)/\mathcal{O}_c \approx 150$ for $n = 9$. On a PDA, without a floating point unit, this number will be even higher and our method even more desirable.

3.2 Ranking the dimensions

The order of the dimensions in the trimming procedure may be arbitrary. We always end up with the same hyper-polyhedron. However, proper ranking may save some computation time. We use the ranking suggested by Nene and Nayar [4]. More precisely, we start slicing in the dimension that discards the highest number of points from the trimmed list.

3.3 Adaptive threshold

It is apparent that the cost of the proposed algorithm depends critically on the radius r. Setting r too high results in a huge increase in cost, while setting r too small may result in an empty candidate list [4]. Nene and Nayar proposed a solution for two special cases — normal and uniform distribution of the point sets. Here, we propose a general solution for any distribution $p(x_i)$. The computation of distributions is expensive, but it can be computed beforehand. Based on the above analysis, our goal is to find the smallest r for which the candidate list is non-empty. Let us assume a specific query point $Q(x_1, x_2, x_3, ..., x_n)$. Assuming the different dimensions are uncorrelated, the joint probability is defined as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i) \; . \tag{5}$$

If we select r such that

$$\prod_{i=1}^{n} \int_{x_i - r}^{x_i + r} p(x_i) dx_i \times N \geq 1 \; , \tag{6}$$

with N the total number of points in the database, the expected number of points inside the hyperpolyhedron ≥ 1. We can approximate the integral for a small neighborhood around Q by a rectangle which yields

$$\prod_{i=1}^{n} 2rp(x_i) \times N \geq 1 \; . \tag{7}$$

To overcome the problem of hypercube corner point mentioned in [4], we use $r/\sqrt{2}$ to replace r and do exhaustive search in the hyper-polyhedron illustrated in the Fig. 3 right. To ensure the hyper-polyhedron is not empty, a safe factor k is introduced into Eq. 7. The desired function which selects the right r is

$$r \geq \sqrt[n]{\frac{k\sqrt{2}}{2N \times \prod_{i=1}^{n} p(x_i)}} \; . \tag{8}$$

4 Experiments

We experimented with our system on several databases. First, we use the often used coil-100 object database. For each object, 5 views(view 0, 100, 215, 270, 325) were included in our database, and another 3 views (view 25, 250, 255) were used as query images. The recognition rate is about 59.5%. The main reason for such a poor performance is that some objects are too simple to extract enough affine regions. If the simple objects for which less than four regions can be extracted are not included, the recognition rate goes up to 77.3%. We designed the proposed method with respect to real world conditions. It is robust to affine transforms, partial occlusions, background changes and (to some extent) illumination changes.

We are mainly motivated by a virtual tourist guide application. We created a rather representative database containing 201 buildings in Zürich [6]. Each of the buildings has been photographed from five different viewpoint. These 1005 images have been captured in different seasons and different weather. The images are subsampled to the resolution 320×240. In total, 57095 affine regions

Fig. 4. Retrieval results on ZuBuD database. First one is query image, others are returned images with descending order of matches. Top row: Best match is the correct one. Mid row: Correct image is not at top fist position but second. Bottom row: Failure. The query image which is highly occluded by a tree.

have been located in the database images. The 115 query images are not included in the database. They have been captured with a different camera and under different photometric conditions. Our method returns the correct match for 99 images. For an additional 10 images, it gives the correct match within the top five. Only 6 images have not been recognized among the first five and are considered as failures. The complete recognition process needs about 1.5 - 1.7 seconds for one query image on PIII at 1GHz. Most of the time is spent on the invariant region extraction. It should be noted that our code is not yet well optimized, so there is still room for improvement. The top row in Fig.4 shows one recognition example in which the query image was occluded by a tram. The mid row illustrates one retrieval result in which the correct result is not at the top first position but at the second. The bottom row demonstrates a failure because of significant occlusions in the query image.

5 Conclusions

In this contribution, a method to retrieve images from a database based on local features was proposed. It makes it possible to search in a database for images containing the same object or scene as shown in a query image, even in case of large changes in viewpoint, occlusions, partial visibility, changes in the illumination conditions, etc. The use of invariance together with efficient indexing allow close to realtime performance. By approximating a hypersphere by a hyperpolyhedron and including an adaptive threshold, the amount of expensive distance computations can be reduced, at the cost of extra scalar comparisons. This is particulary useful in the foreseen application on PDA which typically has no floating point unit. The effective indexing technology provides acceptable searching speed for most applications. It is suitable for two types of queries: range search and k-nearest-neighbor search. When the database becomes bigger, correct recognition rates decline because too many similar regions are found in other images, especially since most buildings have similar colors and features. Adding a re-ranking procedure as a postprocessing step that would include more expensive operations like epipolar geometry verification or cross-correlation could probably further improve the retrieval accuracy.

Acknowledgement

This work is supported by Polyproject-Wearable computing of Swiss Federal Institute at Zurich and the National Fund for Scientific Research Flanders (Belgium).

Appendix: Volume of the hyperpolyhedron

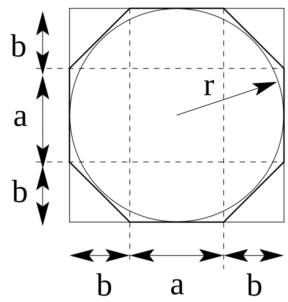


Fig. 5. Two-dimensional hyperpolehydron

To find the total volume of the n-dimensional hyperpolyhedron, we subdivide the n-dimensional hypercube into a set of rectangular boxes as illustrated in figure 5 for the two-dimensional case, and compute for each of these boxes how much of its volume is occupied by the hyperpolyhedron. More precisely, the hypercube is divided by intersecting it with $2n$ hyperplanes Π_i given by the equation

$$x_i = \pm a/2, i = 1..n \ ,$$
$$a = 2(\sqrt{2} - 1)r \ ,$$
$$b = \sqrt{2}(\sqrt{2} - 1)r \ .$$

This way, the volume of the n-dimensional hyperpolyhedron can be written as

$$V_n(r) = \sum_{k=0}^{n} N_k v_k a^{n-k} b^k$$

with N_k the number of boxes with dimensions $a^{n-k}b^k$ and v_k the percentage of the subvolume occupied by the hyperpolyhedron. Indeed, due to symmetry reasons all the subvolumes with the same dimensions will have similar positions relative to the hyperpolyhedron and hence will have a similar percentage of their volume occupied by the hyperpolyhedron.

The number N_k of boxes with dimensions $a^{n-k}b^k$ is given by

$$N_k = \binom{n}{k} 2^k, k = 1..n$$

with $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ the number of combinations of k out of n.

The percentage of each subvolume occupied by the hyperpolyhedron v_k can be shown to be

$$\begin{aligned} v_0 &= 1 \ , \\ v_k &= (1/2)^{k-1}, k = 1..n \ . \end{aligned}$$

Combining all these equations, the volume of a n-dimensional hyperpolyhedron can be written as

$$\begin{aligned} V_n(r) &= a^n + \sum_{k=1}^{n} \binom{n}{k} 2a^{n-k}b^k \\ &= 2^n r^n (\sqrt{2}-1)^n \left[1 + \sum_{k=1}^{n} \binom{n}{k} 2^{1-k/2} \right] \ . \end{aligned}$$

References

1. A. Baumberg. Reliable feature matching across widely separated views. In *Computer Vision and Pattern Recognition*, pages 774–781, 2000.
2. Krystian Mikolajczyk and Cordelia Schmid. An affune invariant interest points detector. In *European Conference on Computer Vision*, pages 128–142, 2002.
3. F. Mindru, T. Moons, and L. Van Gool. Recognizing color patterns irrespective of viewpoint and illumination. In *Computer Vision and Pattern Recognition*, pages 368–373, 1999.
4. Sameer A. Nene and Shree K. Nayar. A simple algorithm for nearest neighbor search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 1997.
5. J. Matas O. Chum, M. Urban and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, 2002.
6. Hao Shao, Tomáš Svoboda, and Luc Van Gool. ZuBuD — Zürich buildings database for image based recognition. Technical Report 260, Computer Vision Laboratory, Swiss Federal Institute of Technology, March 2003. Database downloadable from http://www.vision.ee.ethz.ch/showroom/.
7. T. Tuytelaars and Van Gool. Wide baseline stero based on local affinely invariant regions. In *British Machine Vision Conference*, 2000.