Reducing Keypoint Database Size

Shahar Jamshy[1], Eyal Krupka[2], and Yehezkel Yeshurun[1]

[1] School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
{weasel,hezy}@post.tau.ac.il
[2] Israel Innovation Labs, Microsoft Israel R&D Center
eyalk@microsoft.com

Abstract. Keypoints are high dimensional descriptors for local features of an image or an object. Keypoint extraction is the first task in various computer vision algorithms, where the keypoints are then stored in a database used as the basis for comparing images or image features. Keypoints may be based on image features extracted by feature detection operators or on a dense grid of features. Both ways produce a large number of features per image, causing both time and space performance challenges when upscaling the problem.

We propose a novel framework for reducing the size of the keypoint database by learning which keypoints are beneficial for a specific application and using this knowledge to filter out a large portion of the keypoints. We demonstrate this approach on an object recognition application that uses a keypoint database. By using leave one out K nearest neighbor regression we significantly reduce the number of keypoints with relatively small reduction in performance.

Keywords: Keypoints, Saliency, Recognition, ALOI.

1 Introduction

Many fields in computer vision, such as object recognition [1,2], model based recognition [3,4], object tracking [5], and matching of stereo pairs [6,7] may benefit from the selection of salient areas in the image as the first step of their processing. The selection of salient areas focuses the task on similar areas in different images thus reducing computational complexity and increasing accuracy. These salient areas are often referred to as keypoints.

Various algorithms have been suggested in order to achieve this goal. These algorithms use different techniques such as corner detection [8], local symmetry detection [9,10], convexity estimation [11,12] or blob detection [7,13,14,15] in order to detect interest points in an image.

Since most applications compare keypoints from different images, where the same object may appear with different illumination, scale, orientation, or background, keypoints must be represented in a way that will be invariant to these differences. This representation is called keypoint descriptor (see [15,16,17] and [18] for comparison). For example, SIFT [15] describes the keypoint using a weighted histogram of the orientation of the gradient in the area of the keypoint.

P. Foggia, C. Sansone, and M. Vento (Eds.): ICIAP 2009, LNCS 5716, pp. 113–122, 2009.

In order to compare keypoints from different objects and images the keypoints are stored in a labeled database and are then used as the basis for comparing, recognizing and tracking objects.

Even though there are many saliency operators intended to focus an image processing algorithm on salient areas in the image, state of the art operators (when used with parameters recommended by the authors) produce hundreds of keypoint for a single image which does not simplify the problem enough. Since different operators have different strengths and weaknesses (see [19] for comparison), it is common practice to use a combination of two or more operators [20], yielding more keypoints. Furthermore, it is another common practice to use a dense grid of keypoints (see [21,22]) instead of using a saliency operator, yielding an even larger number of keypoints.

When the number of images and objects grow the keypoints database becomes very large, which causes both time and space performance problems. In practice, a large number of the keypoints discovered are not actually helpful to the actual application (For example, if they belong to the background or to features common to many images and objects). Filtering the database and removing these redundant features we will reduce both time and space complexity of the application.

The rest of the paper is organized as follows: in Section 2 we give a general framework for filtering a keypoint database according to the specific requirements of the application. In Section 3 we show how to implement this framework for an object recognition application. Section 4 compares the results of filtering using our framework and random baseline filtering. We conclude in Section 5 with some further research directions.

1.1 Related Work

Most applications deal with the problems of large keypoint databases either by using a small scale implementation (order of hundreds of objects) to demonstrate their approach [2,4], or by reducing the complexity of the keypoint itself. A common approach (see [3]) uses Vector Quantizations and K-Means in order to reduce each keypoint to a single word in relatively small dictionary.

Another approach, described in [15], uses a hash function to approximate nearest-neighbor lookup. While this approach improves the time performance of the nearest neighbor search it does not reduce the memory required for a large database of keypoints.

Despite the large amount of literature on finding and describing keypoints little care has yet been given to the problem of directly reducing the number of keypoints or to working with databases that contain thousands of images.

2 Our Approach

We introduce a framework for filtering keypoints which is suitable for many computer vision tasks. The main notion of this framework is that an application

can rank individual keypoints based on their usefulness. We use these ranks in order to learn the characteristics of keypoints useful to the application. Fig. 1 shows a general scheme of an application that uses keypoints. First, keypoints are extracted from the image, usually by using a saliency operator. The keypoints are then coded into descriptors, and then some application specific processing is done.

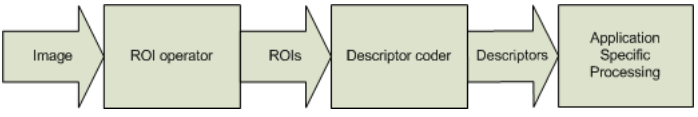

Fig. 1. General scheme of an application that uses keypoints

Our framework works in two stages: a training stage and a filtering stage. In the training stage the target application ranks each keypoint according to its usefulness. The ranks and the keypoints are used in order to train a keypoint filter, as shown in Fig. 2. For example, in an object recognition application, the application can rank the keypoints according to their ability to distinguish between the different objects. Highly distinctive keypoints will receive high grades and less distinctive keypoints will receive lower grades.

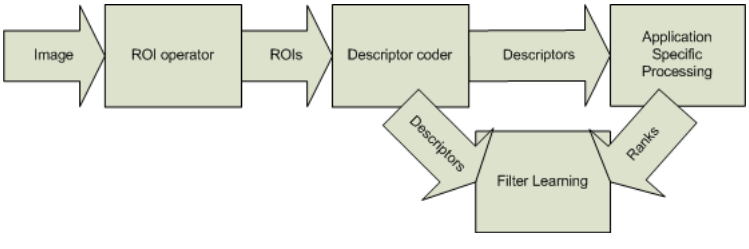

Fig. 2. Training Stage

In the filtering stage we use the rank based keypoint filter we built in the training stage in order to filter out less useful keypoints which reduces the number of keypoints the application needs to process, as shown in Fig. 3.

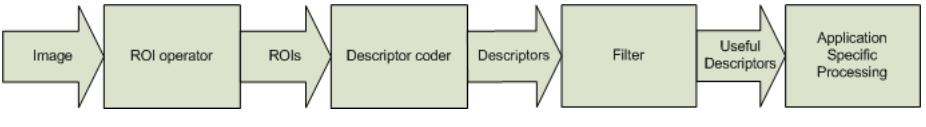

Fig. 3. Filtering Stage

3 Implementation

In order to demonstrate our framework we selected the SIFT based object recognition application described by Lowe in [15]. In this application keypoints are matched using a nearest neighbor database of keypoint descriptors, where the ratio between the distance from nearest neighbor and the first nearest neighbor from any other class is used to asses the distinctiveness of the match.

We used ALOI dataset [23] in order to train a database of labeled descriptors. We used four training images for each object, taken at 90 degrees difference as the training set.

We then gave each descriptor in the database a mark in the following way: Let $I_{i,j}$ be the j training image of object i. We denote $D\big(I_{i,j}\big)$ the descriptors extracted from image $I_{i,j}$. For each descriptor $d \in D\big(I_{i,j}\big)$ in the training set we define the correct label of the keypoint $l(d)$ by

$$l(d) = i \iff d \in D\big(I_{i,j}\big) \text{ for some } j \tag{1}$$

Let n be the number of objects and k be the number of training images for each object we define

$$T = \bigcup_{i=1..n, j=1..k} D\big(I_{i,j}\big)$$

to be a training set, and

$$DB_T = \big\{(d, l(d))\big|d \in T\big\}$$

to be the training database, a database of keypoint descriptors and their respective labels.

For every descriptor d we denote $l(T, d)$ the label of the nearest neighbor of d in T (we find $l(T, d)$ by searching DB_T) and define the correctness measure:

$$m^c(d) = \begin{cases} 1 & l(T,d) = l(d) \\ -1 & l(T,d) \neq l(d) \end{cases} \tag{2}$$

For each j we then break T into two parts: R_j - the descriptors extraced from the j training image for each object, and T_j - the rest of the descriptors in the training set. Formally:

$$R_j = \bigcup_{i=1..n} D\big(I_{i,j}\big) \text{ and } T_j = T \setminus R_j$$

We calculate $m^c_j(d)$ for each $d \in R_j$ using T_j for training set and DB_{T_j} as the training database the same way described in Eq. 2. At this stage we use the set $\big\{(d, m^c_j(d))\big\}$ of descriptors in R_j and their correctness measure as a basis for a K nearest neighbor regression database (we tested the algorithm with 3 and 5 neighbors) and find the regression value $m_j(d)$ for each $d \in T_j$.

Finally, we calculate the mark $m(d)$ for each descriptor $d \in I_{i,j}$:

$$m(d) = \sum_{j' \neq j} m_{j'}(d) \tag{3}$$

We then create a filtered training set T' and a filtered training database $DB_{T'}$ from T in the following way:

$$\begin{gathered} T' = \left\{ d \in T \middle| m(d) > \phi \right\} \text{ for some } \phi \\ DB_{T'} = \left\{ (d, l(d)) \middle| d \in T' \right\} \end{gathered} \tag{4}$$

In order to test the performance of the database we used another image from each object. We extracted descriptors from each test image using SIFT and then matched each descriptor extracted to the database.

Let I_i^t be a test image and $D\left(I_i^t\right)$ be the descriptors extracted from I_i^t:

1. For each $d \in D\left(I_i^t\right)$ we calculated $l(T', d)$ by searching $DB_{T'}$.
2. For each $d \in D\left(I_i^t\right)$ we calculated the distinctiveness ratio suggested in [15]:

$$r(d) = \frac{\text{distance to the nearest descriptor in } T' \text{ labeled } l(T', d)}{\text{distance to the nearest descriptor in } T' \text{ not labeled } l(T', d)} \tag{5}$$

3. We calculated the label of I_i^t

$$l(I_i^t) = majority\left\{ l(T', d) \middle| d \in D\left(I_i^t\right) \text{ and } r(d) > 1.5 \right\} \tag{6}$$

For each descriptor d in the test set, if $r(d) > 1.5$ we say that d was matched. If d was matched and $l(T', d) = l(d)$ we say that d was matched correctly.

4 Results

First, let's look at the performance of the original application without reducing the database size. Table 1 shows general statistics of the databases for 100, 200, and 500 objects. We can see that the precentage of matched keypoints is 16%-19% and the precent of correctly matched keypoints is 13%-18%. This shows that a large precentage of keypoints that were matched had a correct match. The distinctiveness ratio described in Eq. 5 is responsible for this high accuracy. We can also see that at most 20% of the keypoints were matched, showing the potential of using filtering in order to improve the performance of the application.

When looking at the result of the object recognition application described in Sec. 3 for 500 objects, we can see in Fig. 4 that in random filtering, in average, performs rather well by itself, losing only 5% accuracy when filtering 70% of the database showing the potential for reducing the database size. We can also

Table 1. General Statistics of Test Databases. Columns 1-3: quantities of objects, images and descriptors. Column 4-6: percent of matched discriptors, percent of correctly matched descriptors, and percent of mtched images using distinctiveness ratio of 1.5.

# of objects	# of images	# of descriptors	% descriptors matched	% descriptors matched correctly	% of images matched correctly
100	400	71214	19.7	17.6	92
200	800	122252	21.2	18.6	68.5
500	2000	278571	16.3	13.6	61.4

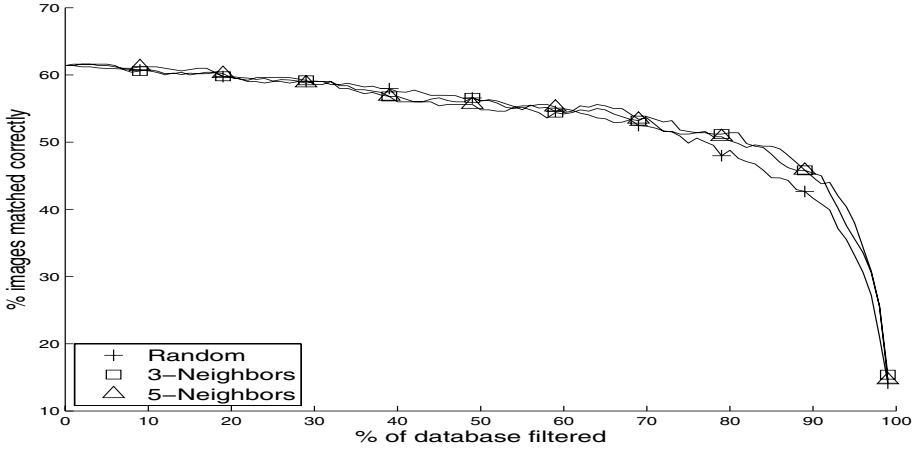

Fig. 4. Percent of matched images vs. filtering of the database for 500 objects, 3 and 5 nearest neighbors compared to averaged random filtering

see that when filtering up to 70% of the database our approach gives similiar results in a predictable way. In Fig. 5 we can see that when filtering 70%-95% percent of the database using our approach we achieve the same accuracy of the average random filtering with 2/3 the database size. For example, in order to match 45% of the images random filtering can filter 84% of the database leaving 16% of the descriptors, while with our approach we can filter 91% leaving only 9%.

Next, let's look at how filtering has affected the number of correctly matched descriptors. In order to assess our results we used an average of 20 random filtering of descriptors as reference. Figure 6 shows the result of filtering for 500 objects in relative percentage over the random reference. The main result these figures show is that when filtering 70%-95% of the database our filtering technique give an increase of 5%-20% percent correctlly matched descriptors relative to random filtering.

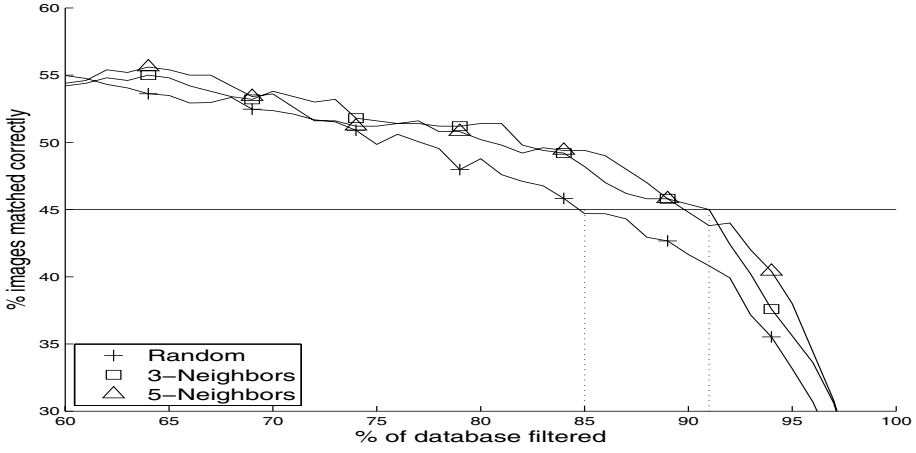

Fig. 5. Percent of matched images vs. filtering of the database for 500 objects, 3 and 5 nearest neighbors compared to averaged random filtering, when filtering 70%-90% of the database, showing we achieve the same accuracy of the average random filtering with less than 2/3 the database size

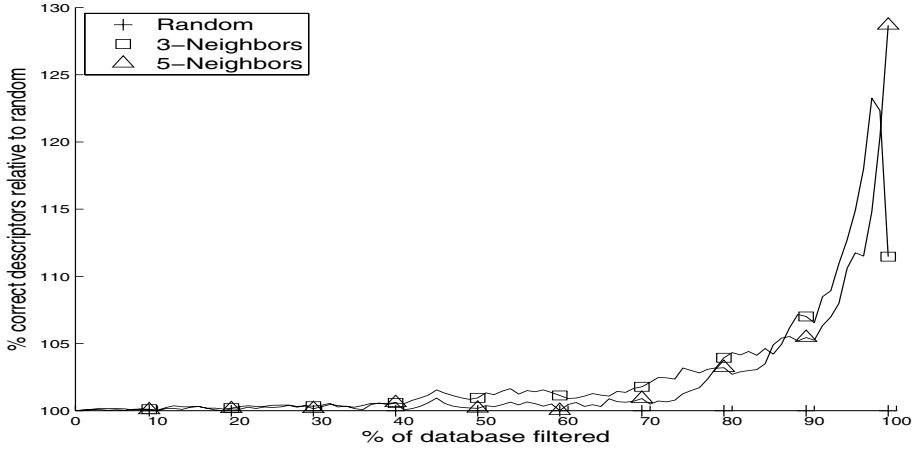

Fig. 6. Percent of matched descriptors relative to the random reference vs. filtering of the database for 500 objects, 3 and 5 nearest neighbors

Finally, when looking at the effect of our method on databases of different sizes, Fig. 7 shows that while for database of 100 and 200 objects our approach did not produce much improvement over random filtering, when database size was increased to 500 objects our approach produced much better results, emphasizing the benefit of our approach for large databases.

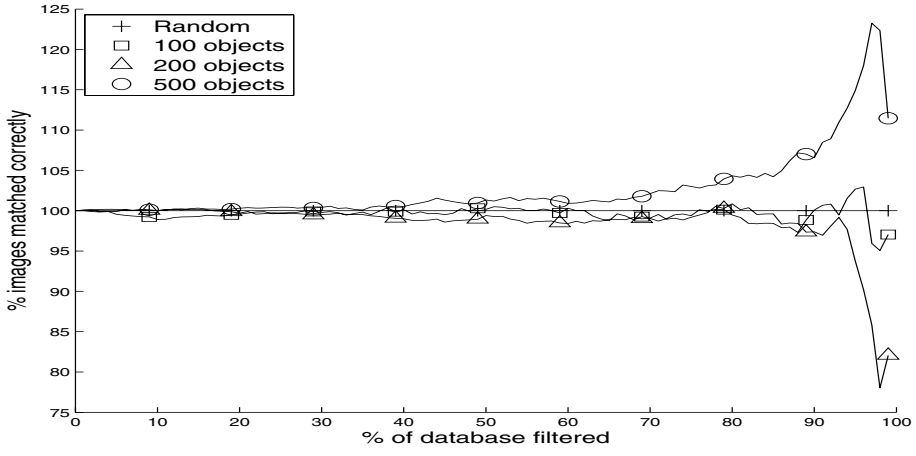

Fig. 7. Percent of matched descriptors relative to the random reference vs. filtering of the database for 100, 200, and 500 objects, 3 nearest neighbors

5 Conclusions and Future Work

In this paper we have proposed a new approach for filtering large databases of keypoints according to the needs of a specific application. We demonstrated this approach on an object recognition application. Using leave one out K nearest neighbor regression we learned the characteristics of useful keypoints and used this knowledge to filter the database. Our experiments have shown that when filtering using our approach we can achieve the same performance results with 2/3 the database size compared to random filtering.

Future research will concentrate on adapting our approach to other applications in computer vision. For example, in computer visions tasks that use the Bag of Keypoints approach (such as [3]) a keypoint filtering stage can be introduced before creating the Bag of Keypoints, filtering out less relevant keypoints and thus increasing accuracy. Tracking tasks can also benefit from our approach by deploying a keypoint filter that will learn to filter out keypoints that are less suitable for tracking. Other directions for future research are trying to improve the training stage by using different machine learning techniques such as SVM or RANSAC, and testing our approach on keypoint databases of even larger scale.

Acknowledgments. I would like to thank Ariel Tankus and Tania Barski-Kopilov for their valuable suggestions and support. In addition, I would like to thank Eddie Aronovich and the CS System Team for installing and supporting the Condor System [24] which made all the computations needed for this research possible.

References

1. Lowe, D.G.: Object recognition from local scale-invariant features. In: ICCV, pp. 1150–1157 (1999)
2. Obdrzálek, S., Matas, J.: Object recognition using local affine frames on distinguished regions. In: Rosin, P.L., Marshall, A.D. (eds.) BMVC, British Machine Vision Association (2002)
3. Dance, C., Willamowski, J., Fan, L., Bray, C., Csurka, G.: Visual categorization with bags of keypoints. In: ECCV International Workshop on Statistical Learning in Computer Vision (2004)
4. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: CVPR, pp. 264–271. IEEE Computer Society, Los Alamitos (2003)
5. Tissainayagam, P., Suter, D.: Object tracking in image sequences using point features. In: APRS Workshop on Digital Image Computing Online Proceedings, pp. 1197–1203 (2003)
6. Pritchett, P., Zisserman, A.: Wide baseline stereo matching. In: ICCV, pp. 754–760 (1998)
7. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: Rosin, P.L., Marshall, D. (eds.) Proceedings of the British Machine Vision Conference, London, UK, September 2002, vol. 1, pp. 384–393. BMVA (2002)
8. Harris, C., Stephans, M.: A combined corner and edge detector. In: Proc. 4th Alvey Vision Conf., Manchester, August 1988, pp. 189–192 (1988)
9. Reisfeld, D., Wolfson, H., Yeshurun, Y.: Detection of interest points using symmetry. In: ICCV, pp. 62–65 (1990)
10. Loy, G., Zelinsky, A.: Fast radial symmetry for detecting points of interest. IEEE Trans. Pattern Anal. Mach. Intell. 25(8), 959–973 (2003)
11. Tankus, A., Yeshurun, Y., Intrator, N.: Face detection by direct convexity estimation. Pattern Recognition Letters 18(9), 913–922 (1997)
12. Tankus, A., Yeshurun, Y.: Convexity-based visual camouflage breaking. Computer Vision and Image Understanding 82(3), 208–237 (2001)
13. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 128–142. Springer, Heidelberg (2002)
14. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. International Journal of Computer Vision 60(1), 63–86 (2004)
15. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
16. Bay, H., Tuytelaars, T., Gool, L.J.V.: SURF: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
17. Winder, S.A.J., Brown, M.: Learning local image descriptors. In: CVPR. IEEE Computer Society, Los Alamitos (2007)
18. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis & Machine Intelligence 27(10), 1615–1630 (2005)
19. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Gool, L.J.V.: A comparison of affine region detectors. International Journal of Computer Vision 65(1-2), 43–72 (2005)

20. Ramisa, A., de Mántaras, R.L., Aldavert, D., Toledo, R.: Comparing combinations of feature regions for panoramic VSLAM. In: Zaytoon, J., Ferrier, J.-L., Andrade-Cetto, J., Filipe, J. (eds.) ICINCO-RA (2), pp. 292–297. INSTICC Press (2007)
21. Bosch, A., Zisserman, A., Muñoz, X.: Scene classification via pLSA. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 517–530. Springer, Heidelberg (2006)
22. Bosch, A., Zisserman, A., Muñoz, X.: Scene classification using a hybrid generative/discriminative approach. IEEE Trans. Pattern Anal. Mach. Intell. 30(4), 712–727 (2008)
23. Geusebroek, J.M., Burghouts, G.J., Smeulders, A.W.M.: The Amsterdam library of object images. Int. J. Comput. Vis. 61(1), 103–112 (2005)
24. Litzkow, M., Livny, M., Mutka, M.: Condor - a hunter of idle workstations. In: Proceedings of the 8th International Conference of Distributed Computing Systems (June 1988)