# TOP-SURF: A Visual Words Toolkit

Bart Thomee Erwin M. Bakker Michael S. Lew
LIACS Media Lab, Leiden University
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
{bthomee, erwin, mlew}@liacs.nl

## ABSTRACT

TOP-SURF is an image descriptor that combines interest points with visual words, resulting in a high performance yet compact descriptor that is designed with a wide range of content-based image retrieval applications in mind. TOP-SURF offers the flexibility to vary descriptor size and supports very fast image matching. In addition to the source code for the visual word extraction and comparisons, we also provide a high level API and very large pre-computed codebooks targeting web image content for both research and teaching purposes.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval, D.0 [**Software**]: General

## General Terms

Algorithms, Design, Documentation, Experimentation.

## Keywords

Content-based Image Retrieval, Interest Points, Visual Words, Bag-of-Words, Codebook, Open Source.

## 1. INTRODUCTION

In our world vision plays a very important role and computers are slowly catching up with the qualities of human vision. In the early days image descriptors were based on low-level features, such as colors and edges, but nowadays the descriptors are approaching image analysis from a higher level, resulting in image descriptors that are based on, for instance, salient details or image patches. Interest points are a specific kind of salient details, which describe locations in an image that are 'interesting' in a certain way. In this paper we present TOP-SURF, which is an image descriptor that combines interest points with visual words. It harnesses the high-level qualities of interest points, while significantly reducing the memory needed to represent and compare images. Our visual word dictionaries (codebooks) are created by analyzing the interest points extracted from millions of web images. The TOP-SURF descriptor is completely open source, which includes the libraries it depends on. Furthermore, the source code can be easily compiled and included in an existing project, or can be used in binary form where its functionality is available through an accessible API.

Because TOP-SURF is based on SURF [1], we will first shortly introduce this image descriptor in Section 2, before discussing our descriptor in more detail in Section 3. Along the way we illustrate the differences in descriptor size, description time and matching time between both descriptors. We also compare the performance of both descriptors using a near-duplicate detection scenario as a showcase. Finally, in Section 4 we will describe the TOP-SURF API, open source licenses, documentation and other possible scenarios in which our descriptor would be useful.

## 2. SURF

SURF is one of the best interest point detectors and descriptors currently available. It has been shown to outperform the other well-known methods based on interest points SIFT [2] and GLOH [3].

### 2.1 Representing an image

The SURF technique uses a Hessian matrix-based measure for the detection of interest points and a distribution of Haar wavelet responses within the interest point neighborhood as descriptor. An image is analyzed at several scales, so interest points can be extracted from both global ('coarse') and local ('fine') image details. Additionally, the dominant orientation of each of the interest points is determined to support rotation-invariant matching. An example image and its detected interest points are shown in Figure 1.

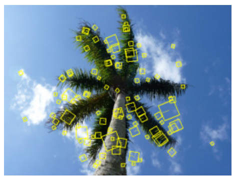

**Figure 1. Detected interest points in an image, including their orientation and scale.**

To determine the average number of extracted interest points per image, we used a collection of 100,000 images downloaded from the internet, which included logos, graphics, celebrity shots, stock photography and travel-related imagery. These images represent well what one would generally encounter when browsing on the internet, with the exception of small icons and banners that have been left out. These images have dimensions ranging between 83 and 640 pixels, with an average size of 460x400. Yet, for extracting the interest points we resized all images to 256x256.

We show the number of detected interest points for a selection of the images in Figure 2. The number of interest points found in an image ranged between 0 and 1057, and was on average 176 with a standard deviation of 85,3. It is thus certainly possible that no interest point is detected in an image at all, which can for instance occur when an image has a single color, although this does not happen frequently. Because each interest point is associated with a 64-element descriptor, the total descriptor size in our image sets thus ranges between 0KB and 270KB per image, with an average of roughly 45KB. On average 0.37s was required to extract the interest points from an image. The results were obtained using standard dual-core 2.4GHz workstation equipped with 3GB RAM. Note that more or less interest points can be detected when the images are resized to a different resolution than 256x256.

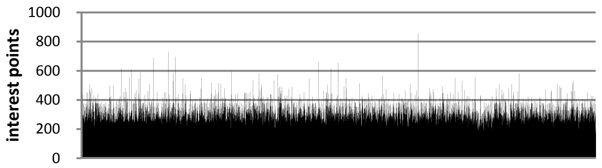

**Figure 2. Number of interest points detected in the first 25,000 images of the collection.**

## 2.2 Matching an image

When enough interest points in the first image match with those in the second image the images are likely to depict the same scene or object(s). To determine these matches the authors of SURF use the nearest neighbor ratio matching technique [2]. Each interest point in the first image is compared to all interest points in the second image by calculating the Euclidean distance between their descriptors. A match is found between a point in the first image and a point in the second image if the distance between them is closer than 0.65 times the distance when any other point in the second image is considered. We used a separate set of query images, consisting of 100 travel-related photos, and matched them with all the 100,000 images. On average 12ms was required to match one query image with all the other images. We performed the matching on a quad-core 2.4GHz workstation equipped with 8GB RAM in order to handle the large descriptor size.

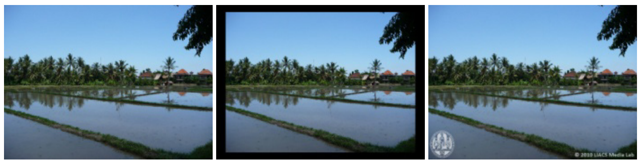

**Figure 3. Examples of near-duplicate images: original image (left), framing (middle), overlaid text and logo (right)**

To determine the accuracy of the SURF descriptor, we created several near-duplicates of each of the 100 photos, similar to how we did this in earlier work [6]. These copies were compressed, scaled, framed, colorized or overlaid with text and a logo, see for example We embedded them in the collection of 100,000 images. When matching the original images with all images, ideally the copies would be ranked ahead of all other images. We used *mean average precision* (MAP) as the evaluation measure, which is calculated by first determining the average precision over all copies for each of the queries and then averaging these average precision values. For clarity, in our results we define *precision* as the number of copies found over the total number of images looked at. Our evaluation found that the MAP for SURF was 0.31.

## 3. TOP-SURF

When considering to find matches in collections containing millions of images it is clear that using the SURF method in its default form is storage-wise infeasible. One of our reasons for developing TOP-SURF was to overcome this issue by significantly reducing the descriptor size.

## 3.1 Representing an image

Several steps need to be performed in order to calculate the TOP-SURF descriptor of an image.

### *3.1.1 Representative interest points*

We used a large set of diverse training images consisting of 1 million images downloaded from the internet, 1 million images downloaded from Flickr and 3000 land- and cityscape photos that we took ourselves. Our aim was to compose a general purpose imagery set that would be representative for the kind of images used by researchers and students in content-based image retrieval.

For each of these images we extracted their SURF interest points and randomly chose 25 points. Because some of the images did not have much detail, it occasionally occurred that less than 25 points were extracted and in those situations we used all of them. Due to limited amount of memory available we could not use all extracted points, and eventually settled on a collection containing 33.5 million interest points. The time required to collect all these points was 120 hours.

### *3.1.2 Clustering into visual words*

We devised an approach based on the bag-of-words technique of Philbin et al. [4] to group the collection of representative interest points into a number of clusters. Since each interest point can be considered a location in a 64-dimensional space, we can see this process as analyzing the locations of all 33.5 million interest points and gathering them into a certain number of groups. First, to find an initial location for each cluster we randomly and uniquely assigned it the location of one of the interest points. Then for each cluster we determined its 100 nearest neighbors, i.e. its closest interest points. If a point was close to multiple clusters we only assigned it to the cluster that it was closest to. We then updated each cluster to become the average of its current location and that of its nearest neighbors. To ensure stability of each of the clusters we performed this process 1000 times. Because discovering the exact nearest neighbors in such a high-dimensional space is very time consuming, we used an approximate nearest neighbors technique based on a forest of randomized kd-trees [9] to speed up this process.

Depending on the intended usage of our descriptor only a small number of clusters may be necessary, whereas in other instances a large number may be required. Therefore we clustered the interest points several times, choosing a different number of clusters that ranged from 10,000 to 500,000. The clustering process was done on a high-performance blade server and required 28GB RAM. In Figure 4 we show the time needed to cluster all these points into the varying numbers of clusters.

The final clusters are commonly referred to as the *visual word dictionary*. Similar to a document consisting of textual words, an image can be interpreted as consisting of visual words. Since in a collection of documents some words appear more frequently than others it is likely that in a collection of images some visual words appear more often than others as well. In our situation, the aim is to emphasize the visual words that do not occur very frequently, since they can be considered to be more special, or more descriptive, when they are found in an image. To assign the visual words weights for emphasis we incorporated a tf-idf weighting technique [5]. Our idf-weights were obtained by recalculating all the interest points of a subset of the 2 million training images and analyzing which of the visual words they would be associated with.

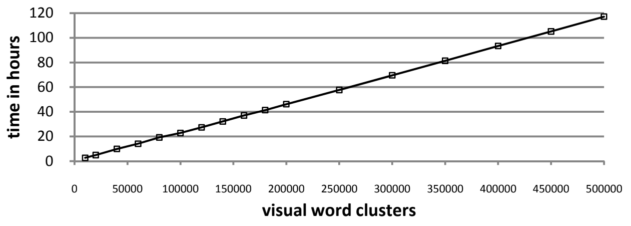

**Figure 4. Required time to cluster the collection of representative interest points into visual words.**

### *3.1.3 Selecting the most descriptive visual words*

Given a particular total number of available visual words, we can now calculate the TOP-SURF descriptor of an image. First we extract its regular SURF descriptor. We then convert the detected points into a frequency histogram of occurring visual words, by analyzing which visual word each interest point is most similar to. Next, we apply the tf-idf weighting to assign a score to all the visual words in the histogram. To form our image descriptor we finally select the highest scoring visual words. Because we only use the top $N$ visual words we thus named the descriptor TOP-SURF. An illustration is shown in Figure 5. Note that our descriptor only requires 8 bytes per selected visual word. Storing a collection of 100,000 images would roughly require 4.5GB when using the SURF descriptor, however this would only require 80MB with TOP-SURF when keeping the top 100 visual words, which is a reduction of more than 50 times.

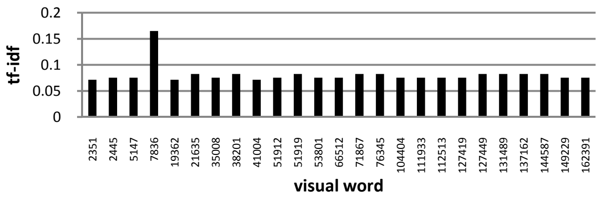

**Figure 5. The histogram of the 25 highest scoring visual words of the image shown in Figure 1 when using a dictionary of 200,000 visual words.**

Like we did with SURF in the previous section, we used the dual-core workstation – thus not the blade server – to calculate the TOP-SURF descriptors for all the 100,000 images. We did this (i) using the various dictionaries that contained 10,000-500,000 visual words and (ii) using different numbers of selected highest scoring visual words that ranged from 10-200 in steps of 10. The time that was required to extract a TOP-SURF descriptor is on average 0,44s. Since the descriptor includes the calculation of the SURF interest points, which required 0,37s as we observed before in Section 2.1, the conversion of the interest points to their visual words and the extraction of the top $N$ points thus approximately took 0,07s. Note that the time needed to determine the top 10 visual words is the same as determining the top 200, because all the visual words still have to be sorted on their scores. From the results we additionally observed that the time to extract our descriptor slightly increased from 0,42s to 0,46s as the dictionary got larger.

## 3.2 Matching an image

To compare the TOP-SURF descriptors of two images we determine the normalized cosine similarity $d_{cos}$ between their tf-idf histograms $T_A$ and $T_B$

$$d_{cos} = 1 - \frac{T_A \cdot T_B}{|T_A|\,|T_B|} \,. \tag{1}$$

A distance of 0 means the descriptors are identical and a distance of 1 means they are completely different. Note that, by definition, comparisons with an image in which zero interest points have been detected will always result in a distance of 1, which is the desired behavior. To determine the matching time between TOP-SURF descriptors, we used the same set of query images as before when we matched SURF descriptors. On average 0.2ms was needed to match one query image with all 100,000 test images. In comparison with SURF this is very fast, since only a small number of visual words need to be compared. In contrast, with SURF each interest point of a query image needs to be compared to the interest points of all other images, requiring much more time. We noticed that matching was slightly faster with descriptors that used only a small number of selected visual words. Matching was also faster as the dictionary size increased, because in this situation it is less likely for the visual words in two descriptors to exactly match, in which case these can be skipped and thus do not require further analysis.

We performed the same near-duplicate image detection experiment as with SURF and our results are shown in Figure 6 for various dictionary sizes. We can see that a larger dictionary yields a higher accuracy for small numbers of retained visual words. In this experiment, the dictionaries containing 100,000 visual words and up gave virtually the same results. As the number of retained words increases, the performance goes up for all dictionaries and levels out at around a MAP of 0.96. Note that the TOP-SURF descriptor size is dependent on the number of visual words retained and not on the dictionary size.

Overall, we can see that the TOP-SURF descriptor significantly outperforms the SURF descriptor when it comes to retrieval accuracy, descriptor size and matching time in the context of near-duplicate image detection.

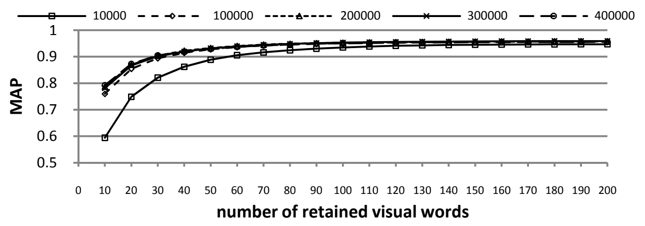

**Figure 6. Mean average precision for various dictionary sizes varying from 10,000 to 400,000.**

## 4. SOURCE CODE

The TOP-SURF descriptor is completely open source, although the libraries it depends on use different licenses. Because the original SURF descriptor is closed source, we used the open source alternative called OpenSURF [7], which is released under the GNU GPL version 3 license. OpenSURF itself is dependent on OpenCV [8] that is released under the BSD license. Furthermore, we used FLANN [9] for approximate nearest neighbor matching, which is also released under the BSD license. To represent images we used CxImage (*www.xdp.it/cximage.htm*), which is released under the zlib license. Our own source code is released under a combination of the GNU GLP version 3 license and the Creative Commons Attribution version 3 license. All these licenses are compatible with each other.

The source code of the TOP-SURF descriptor can be obtained from *press.liacs.nl/researchdownloads/topsurf*. On this page we have also posted documentation and instructions on how to include the library in your own projects. We additionally have provided binaries, samples and a graphical user interface. Because our visual word dictionaries can be quite large, they are offered as separate downloads. All deliverables are currently only offered for the Microsoft Windows platform, both for 32- and 64-bit systems. The source code is presented in a Microsoft Visual Studio 2008 C++ project, although there is no reason to believe it cannot be converted to a project from earlier or later versions of Visual Studio. The code includes all the source code of the libraries it depends on for easy compilation.

### 4.1 API

Our descriptor API offers the following functions:

- **TopSurf_Initialize**
  Initialize the library.
- **TopSurf_Terminate**
  Terminate the library.
- **TopSurf_LoadDictionary**
  Tell the library which visual words dictionary to use.
- **TopSurf_CreateDictionary**
  Create a completely new visual words dictionary.
- **TopSurf_SaveDictionary**
  Save a newly created dictionary to disk.
- **TopSurf_ExtractDescriptor**
  Extract the descriptor of an image.
- **TopSurf_VisualizeDescriptor**
  Display the locations of the detected visual words.
- **TopSurf_CompareDescriptors**
  Compare two descriptors and return the distance between them, either using cosine normalized difference or absolute difference.
- **TopSurf_LoadDescriptor**
  Load a descriptor from disk.
- **TopSurf_SaveDescriptor**
  Save a descriptor to disk.
- **TopSurf_ReleaseDescriptor**
  Release the memory used by a descriptor.

The API only has to be used when accessing the TOP-SURF descriptor through a DLL. When the source code is added to a project there is naturally more control and freedom, since functions can be called directly.

### 4.2 Benefits and uses

For convenience, we allow the user to request all detected visual words in an image and not just the top few. In addition, we extended our descriptor to also include the locations where their original interest points were detected in the image. Both these options allow our descriptor to be used in a variety of situations. For example, an application can analyze the co-occurrence of particular visual words within an image and combine visual words into *visual phrases*, opening up possibilities for improved matching of objects and people. Because of its fast matching speed and low memory requirement the TOP-SURF descriptor is especially useful for mobile and embedded applications, since the devices they will run on are generally restricted by processing power and memory.

Because our descriptor is easy to use and straightforward to integrate into projects, it is not only beneficial to researchers in the content-based image retrieval community, but also very suitable for use in student projects. Examples of student research projects at our computer science department are developing new visual phrase and visual theme search methods based on the pre-computed dictionaries, and automatic robotic navigation based on real time video input.

## 5. CONCLUSIONS

TOP-SURF is a high-performance image descriptor that can be used in a wide range of applications. It is not only very compact, i.e. using little memory, but also exhibits fast matching. Because the descriptor is completely open source, it has all the benefits that open source software provides, such as the freedom to modify and redistribute the code. In addition, we provide pre-computed visual word codebooks, making it easy to start using the descriptor.

## 6. ACKNOWLEDGMENTS

Leiden University and NWO BSIK/BRICKS supported this research under grant #642.066.603. We would like to thank Stephan Gammeter from the Computer Vision Laboratory at ETH Zürich for his critical suggestions.

## 7. REFERENCES

[1] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. 2008. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3), 346-359.

[2] Lowe, D.G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91-110.

[3] Mikolajczyk, K., and Schmid, C. 2005. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10), 1615-1630.

[4] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. 2007. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1-8.

[5] Salton, G., and McGill, M. 1983. Introduction to modern information retrieval. McGraw-Hill.

[6] Thomee, B., Huiskes, M.J., Bakker, E.M., and Lew, M.S. 2008. Large scale image copy detection evaluation. In *Proceedings of the 10th ACM International Conference on Multimedia Information Retrieval*, 59-66.

[7] Evans, C. 2009. Notes on the OpenSURF library. Technical report. University of Bristol.

[8] Bradski, G.R. 2000. The OpenCV library. *Dr. Dobbs Journal*, 25(11), 120-126.

[9] Muja, M., and Lowe, D.G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration, in *Proceedings of the 2009 International Conference on Computer Vision Theory and Applications*, 331-340.