Practical Book 1

Object Recognition

University of Barcelona

May 30, 2016

Authors:
Selva Castelló, Javier
Duato Catalán, Daniel
Camps, Julià

Contents

1 Introduction **3**

2 Discussion on the main SIFT parameters **3**

3 Discussion on the main SURF parameters **4**

4 Performance comparison of SIFT and SURF **4**

5 Analysis of RANSAC for object recognition **6**

6 Conclusion **6**

7 Appendix **8**
7.1 IMAGES USED 8
7.2 RESULTS FROM THE SURF CONCLUDING EXPERIMENTS 9

9 References **10**

1 Introduction

In this practical exercise we will discuss the main parameters of two well known feature detectors and descriptors:

- Scale Invariant Feature Transform (SIFT)
- Speeded Up Robust Features (SURF)

In addition, we will compare their performances on the same testing data. For this experiments we are using the FIND-OBJECT software provided in *https://github.com/introlab/find-object*, which implements the aforementioned detectors.

Finally, we will comment on the influence of RANSAC for object recognition.

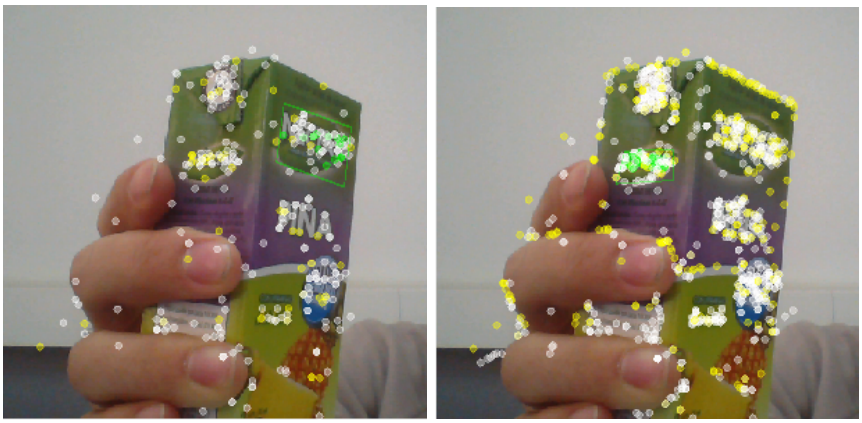
For the experiments, as we already mentioned on the section 1, we are using the FIND-OBJECT program. Since this program is able to work with web-cam, we will use this feature in order to detect objects on scenes where the object 'snapshot' (view) has been altered (displaced, rotated, flashed with a camera, partially occluded, etc.).

The images used for the experiments can be seen in the appendix 7.1.

2 Discussion on the main SIFT parameters

In this section we will discuss the results of tuning the main parameters of the SIFT algorithm.

- **Octave layers:** In our experiments, increasing the number of layers per octave has shown to raise the number of descriptors found on the scene, since the algorithm is searching in more scales. However, we are not improving the performance in any case, since the algorithm did not change its number of hits or misses in the experiment performed.



(a) Descriptors found with Octave layers = 3

(b) Descriptors found with Octave layers = 13

Figure 1: Here we can appreciate the two examples of applying significant values of the Octave layers parameter.

In figure 1 we can observe that the instances of the object found on the scene are different. This is due to the algorithm searching in more scales when increasing the octave layers, so we are able to find the smaller appearance of the object, when with the initial value it was not possible.

- **Contrast threshold:** The effect of the contrast threshold is proportionally inverse to the number of features kept. Even though the default contrast threshold on the program is 0.04, since Lowe [3] used 0.03 on his experiments we decided to start from 0.01 and cover the range of discrete values in the interval $[0.01, 0.07]$. The algorithm was able to handle correctly all the range of values tried, with some "difficulties" (usually fails when trying to find the object) on the extreme configurations, 0.01 and 0.07. The value which showed to perform better was 0.02.
- **Edge threshold:** In this case, the edge threshold is directly proportional to the number of features kept. The default edge threshold on the program is 10 which actually matches the value used by Lowe [3]. We have observed that the effect on the performance of this parameter was highly related to which image was being tested. For example: for the image 2c the algorithm did always find the instance for values of the threshold above 1.65, while for the image 2f it started finding the object only for values higher than 7.00.
- **Sigma value:** The default value of sigma in the program is 1.6, which matches with the value used by Lowe [3]. We have tried a range of values going from 0.1 to 10. This experiment has shown that for values smaller than 0.33 it was not able to find enough features to detect the object. Also, for sigma values greater than 5.5 it fails to recognize the object because the Gaussian filters are too big. The maximum number of features are found using sigma values around 0.4, and from there on it decreases smoothly until it is not able to detect the object.

3 Discussion on the main SURF parameters

In our experiments the SURF detector performed significantly worse than SIFT, correctly matching only around one fifth of the images.

- **Octave layers:** Increasing the number of octave layers increases the number of detected features, since features are searched at more scales. However, no significant changes in performance were observed.
- **Octaves:** Changing the number of octaves also changes the number of features detected, but not as much as the number of octave layers. Less than 4 octave (the default parameter) slightly reduces the number of detected features (presumably removing the larger ones), while increasing this number above 4 doesn't produce any changes. This might be due to the fact that the image isn't big enough to justify the big filters that would be produced at such octaves.
- **Hessian threshold:** As expected, reducing the threshold permits more features to be detected, while increasing it permits less features to be detected.
- **Upright tag:** This tag is used to make faster the feature detection at the cost of sacrificing rotational invariance in the features. In our experiments, given that the only correct matches didn't suffer any rotations, checking this tag didn't make any difference.

4 Performance comparison of SIFT and SURF

In this section we will compare how the different combinations of parameters affected both algorithms' performances.

To simplify the interpretation of the experiments we have decided to count the number of well detected objects on the images, rather than specifying at which image the hits or misses occur.

Configurations:

1 **Default:** This configuration maintains all the default parameters given by the software used for the experiments 1 (which have been shown to be very similar, if not the same, with the values chosen in the references from the bibliography [3] and [1]).

2 **Default SIFT:** This configuration maintains all the default parameters for the "detector" algorithm SIFT (in the software used for the experiments 1), then we copy the common parameters to SURF to match this first ones.

3 **Default SURF:** This configuration maintains all the default parameters for the "detector" algorithm SURF (in the software used for the experiments 1), then we copy the common parameters to SIFT to match this first ones.

4 **High parameters values:** All parameters have been multiplied by 2. From this experiment we want to compare the stability (if they are very tuning dependent) of the algorithms in relation to their configurations.

		Common	SIFT Params			SURF Params			SIFT Results		SURF Results	
		#OL	CT	ET	σ	HT	# OC	UR	Time	Hits	Time	Hits
Configurations	1	3 - 2	0.04	10	1.6	600	4	NO	100	7	115	2
	2	3	0.04	10	1.6	600	4	NO	100	7	130	3
	3	2	0.04	10	1.6	600	4	NO	120	9	115	2
	4	6	0.08	20	3.2	1200	8	NO	130	4	125	2

Table 1: Performance table on different configurations for both discussed algorithms.

LEGEND:
COMMON PARAMETERS: (Two values in the same field indicate that the first one has been used for SIFT and the second one for SURF) **#OL:** Number of octave layers for each octave.
SIFT PARAMETERS: **CT:** Contrast threshold; **ET:** Edge threshold; σ**:** Sigma of the Gaussian.
SURF PARAMETERS: **EX:** Extended; **HT:** Hessian threshold; **# OC:** Number of octaves; **UR:** Upright approach of the SURF algorithm;
SIFT RESULTS: **Time:** Average time that has taken the algorithm when using the SIFT algorithm; **Hits:** Number of images (from the ones presented on the 7) where the object has been correctly found using the parameters on this line of the table for the SIFT algorithm;
SURF RESULTS: **Time:** Average time that has taken the algorithm when using the SURF algorithm; **Hits:** Number of images (from the ones presented on the 7) where the object has been correctly found using the parameters on this line of the table for the SURF algorithm.

In table 1 we can observe and compare the performance of the SIFT and SURF algorithms, given the same object recognition problems, when tuning some of their parameters.

It can be observed that SIFT is finding, in most cases, almost all the instances of the object on the images used for testing, on the other side, the SURF algorithm is giving very

bad results, since has been only able to find at most 2 out of 10 appearances of the object searched.

SURF seems to be more robust, since its performance is not changing when increasing all its parameters. However, since SURF performance is not significant to be compared with SIFT performance, we can not conclude which algorithm is more robust from these experiments.

5 Analysis of RANSAC for object recognition

Once the keypoints have been detected and described by either SIFT or SURF, the next step in the task of object recognition is to correctly match those points with a feature database. In order to do so, a nearest neighbour algorithm is commonly used. However, this task is not as easy as it may seem, as we are going to need multiple feature matches in order to correctly detect and recognise an object, and we need them to agree in the object they are representing. One direct approach to solve this problem would simply be to select the best matches and pick the object represented by their majority. However, this will not always work, since the best matches aren't always correct.

RANSAC [2] solves this problems by working only with a small subset of the keypoints to try and find the object they are describing. RANSAC (RANdom SAmple Consensus) is an algorithm for fitting mathematical models assuming the existence of outliers in the data. As we were saying, there are going to be keypoints that match different objects, and so, they will point out to different directions. RANSAC may be used to avoid this problems and find the best matches. The algorithm works iteratively by trying to match an object only with a randomly selected subset of the keypoints. When an object is matched (model), it tries to fit the rest of the keypoints to that. The model that allows a bigger amount of keypoints to "get into" the model, or the model with an amount of matching points in it above a threshold, is the one selected.

6 Conclusion

We have achieved very good results with SIFT. However our results with SURF have been very disappointing, since it failed to make correct matches even with some simple transformations. Despite what their authors claim, we found SURF to be less robust than SIFT. This could be due to having used images with low texture content. For this reason we tried searching for an object with more texture. The new experiments with the SURF algorithm, showed to confirm our hypothesis: *When searching for objects with more texture, SURF was performing much better than with the object selected initially for this practical work, actually it found the object in all the tested scenes.* This experiments can be seen in the Appendix section 3.

Moreover, no significant difference in performance speed between both detectors was noticed, when actually it was expected from SURF to outperform SIFT in computation speed terms. Finally, RANSAC helps these algorithms by smoothing the problems generated by the outliers when trying to match an object to a set of keypoints.

7 Appendix

7.1 IMAGES USED

(a) Object to be find

(b) Easiest scene

(c) Lower scaled

(d) Rotated 90Â°

(e) Rotated 180Â°

(f) Rotated on Y axis

(g) Lightly occluded

(h) Very occluded

(i) Perspective view

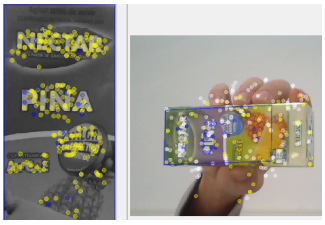
(j) Variations in brightness and contrast

7.2 RESULTS FROM THE SURF CONCLUDING EXPERIMENTS

(a) Frontal view

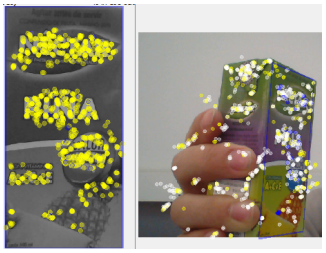
(b) Perspective view

(c) Variations in brightness and contrast



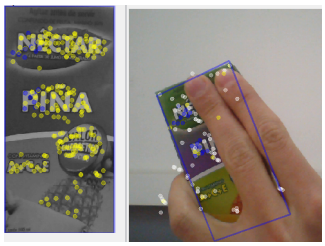
(d) Rotated 90Â°

(e) Rotated 180Â°



(f) Rotated on Y axis, for this image we had to set on the "Affine" parameter to 2, in order to be able to find the instance on the scene.

(g) Lightly occluded



(h) Very occluded

Figure 3: Here we can appreciate the successfully experiments when using the second object tried for the SURF algorithm. The parameters for this experiments where set to default.

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.09.014. URL `http://dx.doi.org/10.1016/j.cviu.2007.09.014`.

[2] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `http://doi.acm.org/10.1145/358669.358692`.

[3] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110. ISSN 1573-1405. doi: 10.1023/B:VISI.0000029664.99615.94. URL `http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94`.