Combining Harris Interest Points and the SIFT Descriptor for Fast Scale-Invariant Object Recognition

Aouad Elias Florez de la Colina Inès

Abstract

Object recognition is a wide range of techniques which consists of recognizing, identifying and locating objects within a picture with a given degree of confidence. In the past decade, local point features methods like the SIFT features, the SURF features or region-based features such as MSER have been used for the recognition and localization of objects. Even though those methods can reach very satisfying results, some of them are too slow when it comes to time-critical applications of object recognition and localization systems that depend on those features. Thus, in this project, we have decided to focus on a solution to this problem based on a combination of the Harris corner detector and the SIFT descriptor. We will see throughout this project how to combine both methods while preserving the scale-invariance. Even though the SIFT descriptor is a very robust and reliable representation for the local neighborhood of an image point, it presents some issues when it comes to computational time. The scale-space analysis required for the calculation of the SIFT feature points can be too slow. Hence, the idea presented in [1] to combine both methods.

1. Introduction/Motivation

The most common tasks of object recognition are classification, tagging, detection and segmentation. These four tasks can be grouped as followed

- Classification and Tagging: object recognition can identify what an image is composed of to a certain confidence level.
- Detection and Segmentation: once we have identified the 'objects' in a given image we would like to locate them.

In this project, we will focus on a very famous feature detection algorithm that is used in object recognition, the scale-invariant feature transform (SIFT), published by David Lowe in 1999. SIFT is a powerful algorithm that is able to find distinctive keypoints in a given image. Moreover, the found keypoints are invariant to location, scale and rotation, and robust to affine transformations. We can also add the fact that SIFT is not affected by changes in the intensity pixels of a given image, thus SIFT is invariant to illumination. Those properties are the key reasons why SIFT is widely used in object recognition. In practice, we compute the SIFT features and match those features to the features obtained in images from the training dataset (cf Figure 1). We will see throughout this report several matching algorithms to compare speed and accuracy of different methods. Finally SIFT features can be applied in many tasks like identifying matching locations between images, recognizing objects in 2D, 3D reconstruction, robot localization, image stitching...

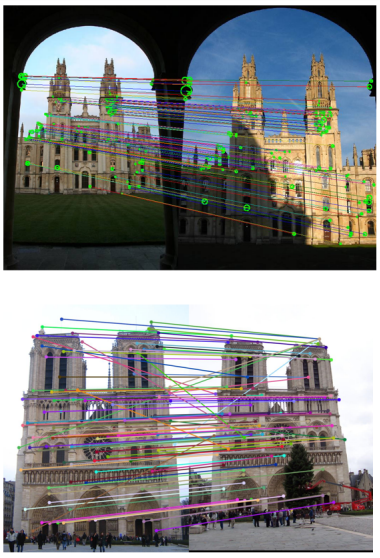


Figure 1: Matching between two images from the same landmarks but not from the same perspectives

SIFT has become a widely used method when it comes to recognizing or locating objects. This algorithm is composed of four major steps: Scale-Space Extrema Detection, Keypoint Localization, Orientation Assignment and Local

Descriptor Creation. Some applications of SIFT, like time-critical applications of object recognition and localization, demand a very low computation time. Unfortunately, the first step to compute SIFT, the scale-space analysis is too slow for visual servoing applications. Hence, we will implement the method described in [1], which consists of replacing this step by using a Harris corner detector. This method has proven to be less time-consuming. The challenge here will be to preserve the scale-invariance property. We will detail the methodology in this report.

The project has proven to be very interesting as the SIFT descriptors are very robust and reliable representations for the local neighborhood of an image point and it is a shame that computation time becomes a barrier for many recognition and localization methods. We will tackle this problem by merging the SIFT and Harris corner detector algorithms.

2. Problem Definition

As mentioned in the introduction, the scale-space analysis of the algorithm is too slow. Thus, to make SIFT faster, we will optimize this step by implementing the Harris corner detector. SIFT presents many important properties (described in the previous section) and among those we have scale-invariance. This is achieved by analyzing and processing images at different scales. Indeed, we take the input image and apply gaussian smoothing s times, to obtain a set of s images in the first group, group which is given the name octave. The variance of the gaussian kernel is chosen such that the blur of the last image of the set is twice the blur of the first image. Then, we take the last image of the first octave, resize it, by dividing its size by two. Once again, we apply the gaussian smoothing s times and repeat the process till we obtain the number of octaves wanted. This property is given by the scale-space analysis. Hence, the constraint will be to maintain this property while using the Harris detector.

3. Related Work

In the past decade, The SIFT algorithm and the Harris corner detector algorithm have usually been compared to see which one performed best in a given situation. [3] presents a comparison of both algorithms in feature matching tasks. It compares which features would match better, the Harris features or the SIFT features. According to the results of the conducted experiments in [3], Harris features are better than SIFT in only one aspect. They are very little time-consuming. Apart from that SIFT features are more robust and the matching accuracy has proven to be much higher. [4] compares image stitching using both algorithms. Both algorithms are widely used in computer vision, and both have many applications. The approach developed in [1] comes as an innovation as it doesn't compare both algorithms, this time it combines them together to reach a lower computational-time. Its objective is to take the best of both methods, low computation time for Harris, invariance properties for SIFT, and build a more effective algorithm.

4. Methodology

4.1. Overall Methodology

To solve the problem described in the previous sections we have decided to follow the method described in [1] and implement it. Here are the main steps we will follow:

- As previously mentioned, the most time-consuming part of the SIFT algorithm is the scale-space analysis performed for calculating the keypoints positions. This corresponds to the first two steps of the SIFT algorithm: Scale-Space Extrema Detection and Keypoint Localization. Hence, we will modify those steps by including the Harris corner detector while preserving scale-invariance.

- In the original SIFT algorithm, the one described in [2], we use octaves to ensure the scale-invariance. Here, by using the Harris detector, we will preserve this property by computing the corners of the all gaussian images of each octave. Finally we will compare the performances of the SIFT implemented with Harris and the general SIFT by comparing the matchings obtained with both methods.

- To compare matches, we will implement a matching algorithm based on Lowe's second nearest neighbour test and a geometric transformation using RANSAC.

Thus, to properly carry out this project, we first have to understand well how the SIFT features and keypoints are computed, so we are able to modify the algorithm to include the Harris corner detector.

4.2. SIFT Implementation

SIFT, which stands for Scale Invariant Feature Transform, is a method widely used in computer vision for extracting feature vectors that describe local patches of an image. As previously mentioned, these features have very important properties. To implement SIFT, we have followed the pipeline described in the original paper [2].

4.2.1 Step 1: Scale-Space Extrema Detection

The use of a scale space in this algorithm has the objective of replicating a common phenomenon that we observe in nature. Sometimes in an image, you need to zoom in to be able to see some details. Thus, in this first step of the algorithm, we build a scale space which is defined as a function

$L(x, y, \sigma)$. This space is generated from the convolution of a gaussian kernel with the input image. As explained in a previous section, we build octaves with a given number of images s, such that the blur of the last image of the octave is twice the blur of the first image of the octave. To achieve this, we can choose a variance of $\sigma = 2^{(1/s)}$ for the gaussian filter. Then this last image is resized and becomes the first image of the next octave. The octaves created are used to generate another set of images: the Difference of Gaussians (DoG) (cf Figure 2). DoG is used as an approximation of the LoG filter, which is expensive to compute. On Figure 3 are shown the first images of the first six DoG octaves of the all_souls image from Oxford. We understand by looking at this figure that by resizing and blurring the image at each octave, we can spot potential keypoints that might not have been visible otherwise.

These octaves are key for determining the keypoints of the given image. They are defined as extremas. To detect them, we scan each scale-space DoG octave and include the center of each 3x3x3 neighborhood. Those centers become keypoints if and only if their intensity is the maximum or the minimum value of the neighborhood. Hence we get candidate keypoints. The following section will present methods to improve this first selection.

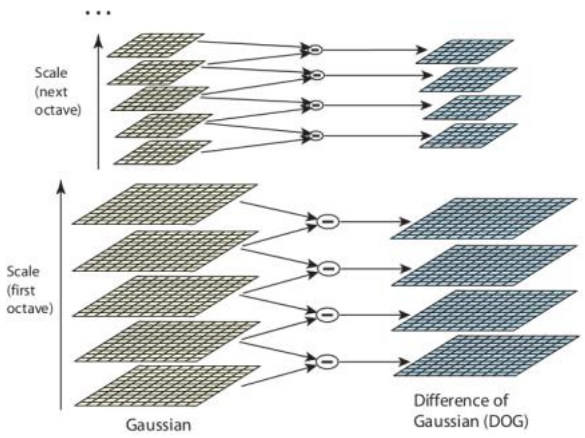


Figure 2: Gaussian and DoG pyramids (source [2])

4.2.2 Step 2: Keypoint Localization

In the above step, we have computed many keypoints. Not all of them are interesting as features so we will get rid of the points of low contrast and the points that are on an edge. To do so, we will perform the three following steps:

- Compute the subpixel location of each point.
- Throw out that keypoint if its scale-space value at the subpixel is under a certain threshold.

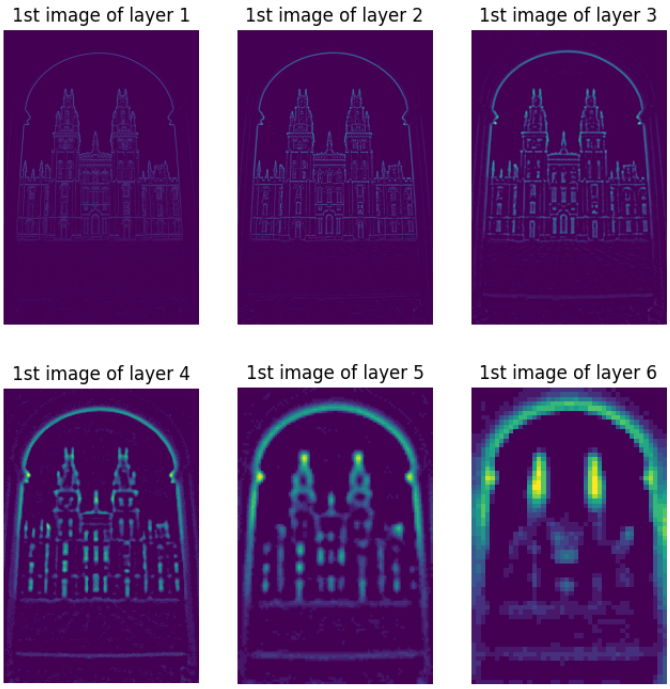


Figure 3: First images of the first 6 DoG octaves

- Eliminate keypoints on edges using the Hessian around each subpixel.

First of all, by denoting D a given octave, we can improve our keypoint selection by using a method developped by Brown based on the Taylor expansion of the scale space function $D(x, y, \sigma)$. We have the following expression:

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \quad (1)$$

where D and its derivatives are evaluated at the sample point and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from this point. This Taylor series expansion allows us to obtain a more accurate location of the extrema, given by $\hat{\mathbf{x}}$. By computing the derivative of (1) we get:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}} \quad (2)$$

If the intensity of the computed extrema is such that $|D(\hat{\mathbf{x}})| < threshold$ (low contrast), we just get rid of this keypoint. [2] gives the following value: $threshold = 0.03$. To compute the Hessian and Jacobian of D we have followed the method suggested by Brown, which consist of approximating them by using differences of neighboring sample points.

Second of all, we need to eliminate edge responses. To do so, we use the Hessian matrix already computed to eliminate low-contrast keypoints and we use a very similar technique than the one used for the Harris detector. If

$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$ with the threshold $r = 10$, then we keep the keypoint, otherwise we eliminate it.

4.2.3 Step 3: Orientation Assignment

Until now, we have obtained keypoints which verify the following properties: translational invariance given by the convolution with the gaussian filter, and scale invariance given by the computation of the DoG octaves. This step is here to ensure rotational invariance. To achieve this, we take a patch around each keypoint and assign an orientation to it. This orientation will be given by its dominant gradient direction which will be found by building a histogram of gradients. We will assign the orientation of the bin with the highest weight in the histogram.

4.2.4 Step 4: Local Descriptor Creation

Finally, this step handles the computation of the SIFT descriptors. Now that we have our keypoints that satisfy scale invariance, translational invariance and rotational invariance, we can find those descriptors. To create a feature vector, we also use a histogram of gradients. We use a 16x16 patch around each keypoint. This patch is divided into 16 4x4 subregions. For each block we create an 8-bin orientation histogram. . Finally, all of these histograms are concatenated into a 4x4x8=128 element long feature vector. This final vector is then normalized, thresholded, and renormalized to ensure invariance to minor changes.

4.3. Combining Harris detector and SIFT

4.3.1 Using Harris corners instead of extremums

To combine both the Harris corner detector and the original SIFT algorithm, we had to slightly modify step 2 (keypoint localization). The original SIFT scanned through each DoG octave and included the center of each 3X3X3 neighborhood if that center was an extrema of the considered patch. Now, with the Harris detector, we just initialize the keypoints by applying Harris to all gaussian images of every octave. By keeping the DoG scales, we ensure that the scale-invariance property is maintained. After having initialized the keypoints with the Harris corner detector, we proceed as before by getting rid of the extremas with low contrast and the ones on edges. We will see, in the next section how this method decreases the computational time of our algorithm.

4.3.2 Bilateral filtering before combining Harris and SIFT

In this process, we are actually using the same pipeline as described in step 1, hence combining Harris corners and SIFT keypoints. However, before starting the process of finding keypoints, we smooth each image with a bilateral filtering with the following parameters : filter size 9X9, $\sigma_{\text{color}} = 75$ and $\sigma_{\text{space}} = 75$. This has been performed to help the Harris detector to find the most appropriate corners, since the bilateral filter smooths only flat regions of an image.

4.3.3 Detecting corners only on first image of each octave

The idea here is that the corners are not so well detected after several Gaussian smoothing operations. Hence, only the first one or two Gaussian images are eligible for detecting real corners, however after the third blurring of an octave, we can't detect real corners. This is why, in order to reduce much more the computation time, we computed keypoints relying only on the first Gaussian image of each octave. We report matching accuracies in the evaluation section. For easier notations, we will call this method the 'First Image' method.

4.4. Matching

Now that we have computed the SIFT features, we can plot the keypoints and match the computed keypoints according to their descriptors. We have decided to compare 3 types of matching:

- Nearest neighbours matching
- Lowe's second nearest neighbour test
- Improved matching by geometric verification

4.4.1 Nearest Neighbours matching

Nearest Neighbors matching represents a simple matching approach based on finding, for each keypoint, the nearest descriptor in terms of the Euclidean distance. We will see in the evaluation of the results that this matching can be highly improved by using one of the two following methods.

4.4.2 Lowe's second nearest neighbour test

This method comes as an improvement to the one described above. Lowe introduced a second nearest neighbour test to identify, and hence remove, ambiguous matches. In this method, we consider the two nearest neighbours of a given keypoint. We compute the NN_{ratio} defined by:

$$NN_{ratio} = \frac{1^{st}NNdistance}{2^{nd}NNdistance} \tag{3}$$

If this ratio is larger than 0.8, then we remove the match. Finally, we can also add a constraint on our matches given by the method below.

4.4.3 Improved matching by geometric verification

In addition to the 2nd NN test, we can also require some consistency between the matches. To do so, we consider a geometric transformation between the two input images. For two given keypoints, $(x_1, y_1, s_1, \theta_1)$ in image 1 and $(x_2, y_2, s_2, \theta_2)$ in image 2, we can find parameters (t_x, t_y, s, θ) such that:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = sR(\theta) \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{4}$$

In this case, we have (x_1, y_1), (x_2, y_2) the coordinates of the keypoints, (s_1, s_2) the diameters of the keypoints' neighborhoods and (θ_1, θ_2) the orientation of each keypoint. We have:

$$\begin{aligned} \theta &= \theta_1 - \theta_2 \\ s &= \frac{s_2}{s_1} \\ t_x &= x_2 - s(x_1 cos(\theta) - y_1 sin(\theta)) \\ t_y &= y_2 - s(x_1 sin(\theta) + y_1 cos(\theta)) \end{aligned} \tag{5}$$

Finally, we can find the consistent matches by applying the the RANSAC algorithm described by the following steps:

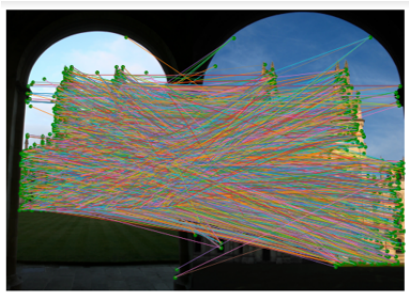
- Compute the similarity transformation define by equation (4)
- Map all the SIFT detections in one image to the other using this transformation
- Count the number of inliers
- Repeat the above steps for a given number of iterations and keep the transformation with the highest number of inliers.

Given this methodology, we will present in the next section our results and how we compared matching performances between the original SIFT algorithm and the algorithm combining both the Harris corner detector and the SIFT.

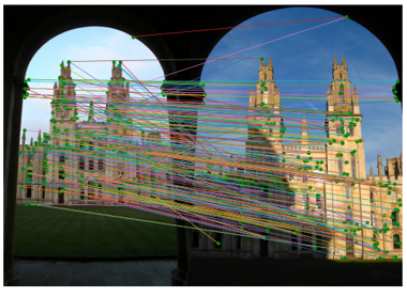
5. Evaluation/Results

5.1. Matching

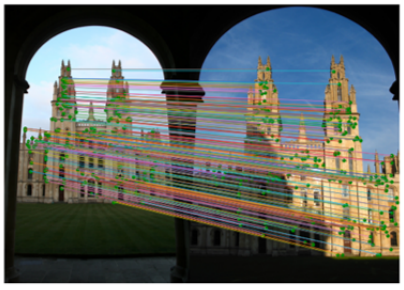
Matching is very important for the SIFT descriptors because it is a way to visualize the accuracy of the descriptors. Before judging the accuracy of our results, we have compared different matching techniques using the OpenCV's SIFT descriptors. We have considered OpenCV to be our ground truth. This will be very useful later to compare the performances of our SIFT implementation with the combination of SIFT and Harris. As presented in the previous section, we have compared three methods: Nearest Neighbours matching, Lowe's second nearest neighbour test and finally the geometric matching. We can compare matchings on Figure 4. We can immediately see that the best method is, without a doubt, the geometric matching. Nearest Neighbours takes into account all keypoints, thus we cannot really distinguish the matchings. Nevertheless, if you look closely, you can spot many mismatches. For Lowe's matching, the results are clearer, but we can see many mismatches as well. Finally, geometric matching works very well and gives very satisfying results. This method is based on the RANSAC algorithm and comes as an improvement of the two previous methods. Hence, according to the results based on OpenCV's SIFT features, we will only use the geometric matching in the following.



(a)



(b)



(c)

Figure 4: (a): Nearest Neighbours matching, (b): Lowe's second nearest neighbour test, (c): Geometric matching

5.2. Original SIFT algorithm

To implement this algorithm we have understood, used, commented and modified the imple-

mentation found following this link: https://github.com/rmislam/PythonSIFT.git. It follows described in the methodology section. Here, we want to check the positions of the computed keypoints by comparing the found keypoints with OpenCV's implementation, as well as checking the accuracy of the matching. Furthermore, we needed to implement SIFT from scratch to be able to include Harris in the algorithm. We will compare in a further section, the computational times it takes to run both algorithms.

We can see on Figure 5 that overall, we obtain very similar keypoints. The density of keypoints obtain on the center monument is a little higher for the implemented version of SIFT. Nevertheless, we obtain with this method less 'error' keypoints (for example, the keypoints in the middle of the sky or on the grass). We can conclude from these observations, that the keypoints obatined by the implemented version are very satisfying. Let's now analyse the accuracy of the descriptors by looking at the geometric matching.

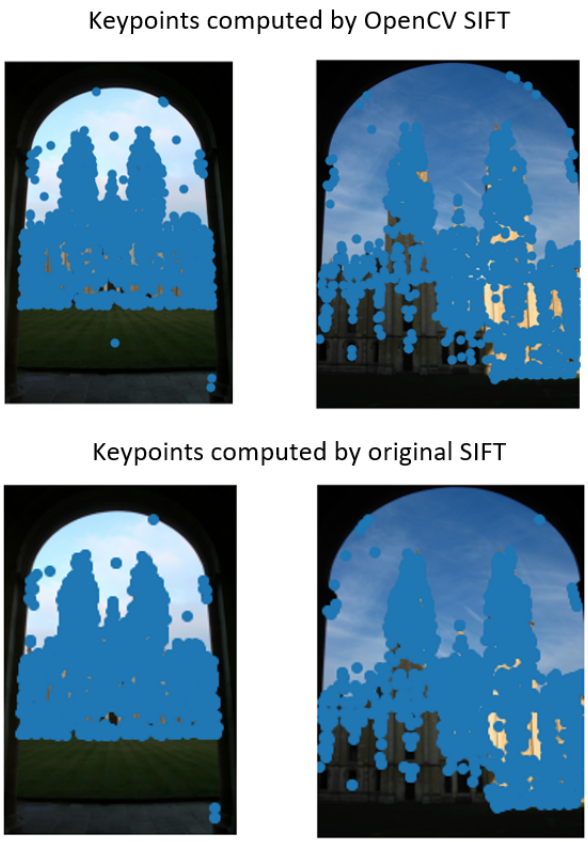


Figure 5: Comparison between keypoints computed by OpenCV's SIFT and the implemented SIFT

We can see on Figure 6 that the matching of the implemented version is also very satisfying. Compared to the OpenCV version, there seems to be less inliers (for the RANSAC algorithm) as it looks like there are fewer keypoints. Furthermore, there less matches on the towers of the the monument. Nevertheless, the results of the implementation from scratch are very good. Now that we have a good implementation of the SIFT algorithm, we can modify it to obtain an algorithm that combines SIFT and the Harris detector.

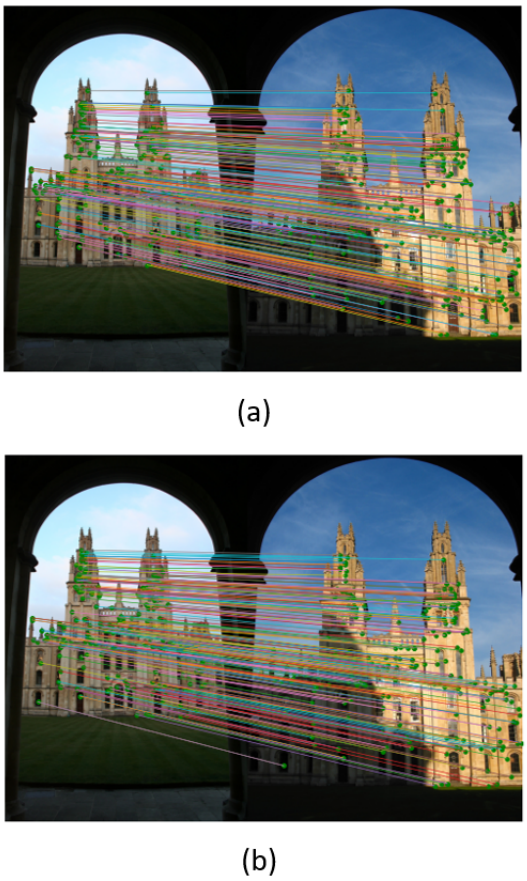


Figure 6: Comparison between the matching obtained with OpenCV's SIFT (a) and the implemented SIFT (b)

5.3. Combining SIFT and Harris

As explained in the previous section, we have modified step 2 of the original SIFT implementation to include Harris. We will compare the keypoints and the descriptors found with the combined algorithm and the original one.

On Figure 7, we can see by comparing with Figures 5 and 6 that we obtain similar keypoints but less than with the original SIFT implementation. For the descriptors, the matching is accurate. We observe for the Harris version

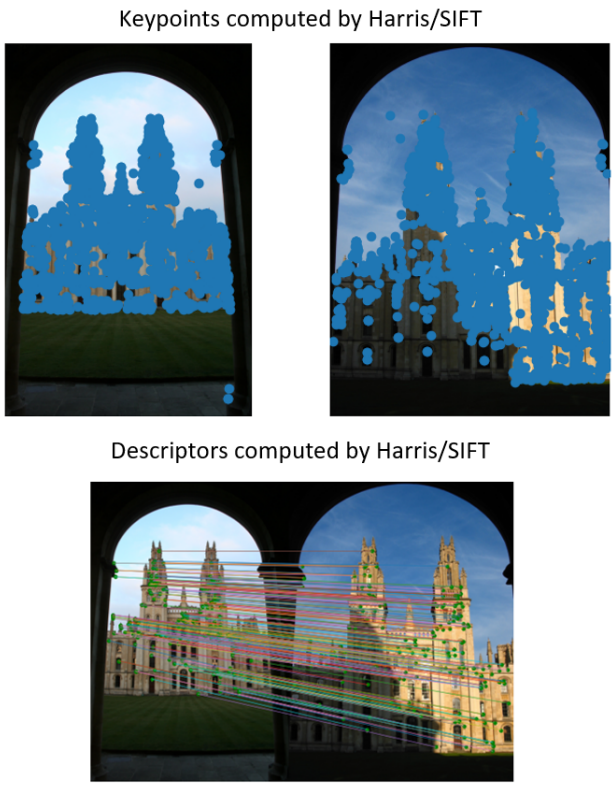


Figure 7: Keypoints and Descriptors found by combining SIFT with Harris

some matches of keypoints on the edges of the ark that were not present in the original implementation. Overall, the results are satisfying.

5.4. Bilateral filtering before combining Harris and SIFT

As we can see on figure 8, the matching occurs on a smaller set of points than before. Indeed, as underlined before, smoothing prevents from computing more corners than there are. The matching is well performed, however it is of course less confident than using all SIFT keypoints. However, it is way faster than the previous methods. We report computational times in the last subsection.

5.5. Detecting corners only on first image of each octave

In comparision with what was explained in previous paragraphs, here we only compute Harris corners on the first image of each octave, since above the first image the Gaussian blur is just too heavy for the image to detect corners. We can see the result of the matching in the following figure 9. Once again, the number of matched keypoints is cut drastically, however we still get a nice and proper match. However, the match is way less confident than when we had

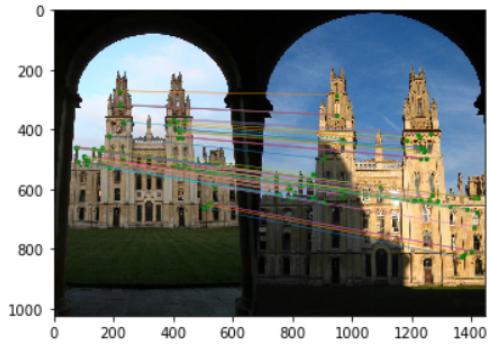


Figure 8: Matching found by using SIFT with Harris, but using only first image of each octave for computing keypoints

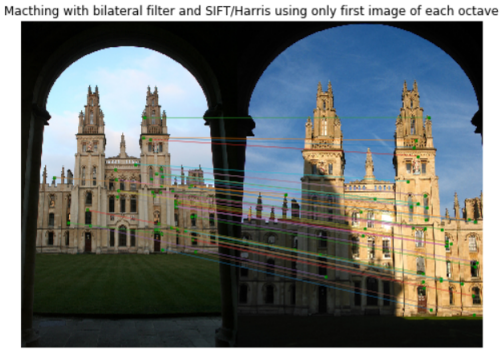


Figure 9: Matching found by smoothing with bilateral filtering before combining SIFT with Harris

numerous SIFT keypoints. In terms of time however, it is way faster than any of the precedent methods, as explained in the next subsection.

5.6. Computational time

We observe a decrease in the computational time at each method. The results are shown on Table 1.

Table 1: Computational times of all methods

Method	Computational time (sec)
Original SIFT	485
Harris/SIFT	431
Filtering+Harris/SIFT	203
First Image	173

6. Conclusion

In conclusion, we have seen in this project many ways to compute effecient keypoint features with reduced computation time. However, it goes without saying that each

method reduces the number of eligible keypoints, hence reducing the number of matched points between two similar images. Nonetheless, the matching remains sufficient enough to grasp the similarity between images.
The SIFT method still outperforms the other methods in terms of number of matched keypoints. But in terms of computational time, using a Harris corner detector on the first image of each octave and computing descriptors afterwards still preserves the scale invariance, and matches images in record time : 173 seconds on our machine, comparing to 485 seconds for the original SIFT algorithm.
Even though we might still prefer to use SIFT for its high accuracy, it is still nice to know there are other methods that might output a little bit less matched points, but that are way less time consuming.

References

[1] Pedram Azad, Tamim Asfour, Rudiger Dillmann *Combining Harris Interest Points and the SIFT Descriptor for Fast Scale-Invariant Object Recognition*. Institute for Anthropomatics, University of Karlsruhe, Germany, 2009 1, 2

[2] David G. Lowe *Distinctive Image Features from Scale-Invariant Keypoints*. Computer Science Department, University of British Columbia, Vancouver, B.C., Canada, 2004 2, 3

[3] My-Ha Le, Byung-Seok Woo, Kang-Hyun Jo *A Comparison of SIFT and Harris Conner Features for Correspondence Points Matching* http://www.cs.tau.ac.il/~turkel/imagepapers/comparison_sift-harris-corner.pdf 2

[4] Miss. Pornima V. Patil , Prof. Dr. M. S. Chavan *A Comparative Analysis of Image Stitching Algorithms Using Harris Corner Detection And SIFT Algorithm* International Journal of Engineering Research and Technology, 2017 https://www.ripublication.com/irph/ijert_spl17/ijertv10n1spl_92.pdf 2