Final project

Digital Image Processing

Project subject: SIFT - Scale Invariant Feature Transform	
Project members: 1. Eryk Urbański 2. Zofia Karasińska	**First version date:** 18.01.2024
	Last edit: 20.01.2024

Table of contents

1. Introduction

1.1. What is SIFT?

The Scale-Invariant Feature Transform is a computer vision algorithm composed of detection, extraction, description and finally matching of intricate features in images. It was proposed by David Lowe in 1999 [1][2] and is still being introduced in many different implementations.

1.2. What is it used for and what are its advantages?

The main use case that benefits from SIFT's advantages is recognition of objects with complex appearances for which simple approaches like edge and corner detection are insufficient. Moreover, in cases where a more sophisticated method like template matching faces problems, SIFT can help overcome them. An example of this is when an object to be found in an image is rotated and magnified differently and so it does not show up in its original, template form. SIFT enables us to deal with this kind of situation, leveraging one of its main characteristics - scale invariance. Another positive aspect of SIFT is that the detected regions - interest points, or keypoints (introduced and described later in the documentation) - are scattered throughout the whole reference image, which makes it possible to find objects occluded by other objects. This is achievable because features extracted using SIFT are highly descriptive, whereas template matching would require a lot of different partial templates in many configurations, making SIFT computationally much more efficient. The SIFT algorithm helps with some more creative and complex tasks that output an actual, visually plausible product - for instance automatic panorama stitching or image collage creation.

1.3. Limitations

The Scale-Invariant Feature Transform is a very capable and robust algorithm, but it also has its limitations. It has the capability to successfully handle flat-characterised objects, however when applied to the recognition of three-dimensional objects, SIFT encounters challenges due to the changing local appearance of features based on the viewpoint. In practical terms, using SIFT on images of the same three-dimensional object from different perspectives results in a large number of features but a decreasing number of correct matches as the viewpoint difference increases. Another problem is connected to resource requirements and scalability. The computational complexity of the SIFT algorithm can be higher, especially when processing large images. The process of extracting SIFT features and matching them can be time-consuming, which is especially important for applications that require fast response times or real-time operation. In situations where the number of features is significant, processing a large number of features can be performance-intensive, while too few features may result in inaccurate matching.

2. Algorithm description

2.1. Step explanations

1. The first step to take in the SIFT algorithm pipeline is to compute the Gaussian scale-space, which ultimately is a stack or a volume of images created by filtering a reference image with Gaussians of different sigma (σ).
2. By subtracting each pair of consecutive images in the stack the Difference of Gaussians (DoG) is obtained.
3. The next step is to find extrema in the image volume. Those extrema are the candidates for interest points. Processing images with complex appearances relies on the analysis of structures more suitable and abstract than edges or corners. In the SIFT method these structures are regions around interest points with some kind of rich content.
4. Filter out weak extrema, suppressing the effects of noise using threshold. The result are the detected keypoints. Visualisation includes drawing circles in the original image around each keypoint with radius proportional to the sigma value. That's while a stack of filtered images was created - to assure scale invariance.
5. Overlapping a region with a square window gives the possibility to calculate the orientation of the gradient at each pixel. With that, a histogram is created, and the maximum value in it corresponds to a gradient direction which is set as the principal orientation of the keypoint. The gradient magnitude is ignored since it is affected by factors such as lighting and camera gain.
6. A signature or a descriptor needs to be computed in order to match a feature with another. Computations are based on a scale and rotation normalised window of a SIFT feature pixels. The window is divided into four quadrants and a histogram of each of those quadrants is computed. Then, the four histograms are concatenated into one and this is referred to as the SIFT descriptor.

2.2. Algorithm summary

1. Compute the Gaussian scale-space
   input: image
   output: scale-space
2. Compute the Difference of Gaussian (DoG)
   input: scale-space
   output: DoG

3. Find keypoint candidates (extrema)
   input: **w** DoG
   output: {($\mathbf{x_d}$, $\mathbf{y_d}$, $\boldsymbol{\sigma_d}$)} list of discrete extrema (position and scale)
4. Exclude weak extrema obtaining keypoints
   input: {($\mathbf{x_d}$, $\mathbf{y_d}$, $\boldsymbol{\sigma_d}$)}
   output: {(**x**, **y**, $\boldsymbol{\sigma}$)} list of keypoints
5. Assign a reference orientation to each keypoint
   input: ($\partial_m\mathbf{v}$, $\partial_n\mathbf{v}$) scale-space gradient and {(**x**, **y**, $\boldsymbol{\sigma}$)}
   output: {(**x**, **y**, $\boldsymbol{\sigma}$, $\boldsymbol{\theta}$)} list of oriented keypoints
6. Build the keypoints descriptor
   input: ($\partial_m\mathbf{v}$, $\partial_n\mathbf{v}$) scale-space gradient and {(**x**, **y**, $\boldsymbol{\sigma}$, $\boldsymbol{\theta}$)}
   output: {(**x**, **y**, $\boldsymbol{\sigma}$, $\boldsymbol{\theta}$, **f**)} list of described keypoints

3. Results

3.1. Detecting interest points - invariance to orientation and scale

Below are visualisations of keypoint areas calculated using different thresholds for extrema filtering and for different test images, the training image remaining the same. Figure 1 shows the result of applying a low threshold contributing to a high number of interest points. As it can be seen, many keypoints seem to be in similar physical locations within images. This can be more easily perceivable looking at Figure 2, where a higher threshold results in fewer keypoints and the test image is just slightly rotated in reference to the training one. Figure 3 features an image both significantly rotated and differently magnified, to be precise, from a much more distant viewpoint. In this last one the seemingly matching locations can also be noticed.

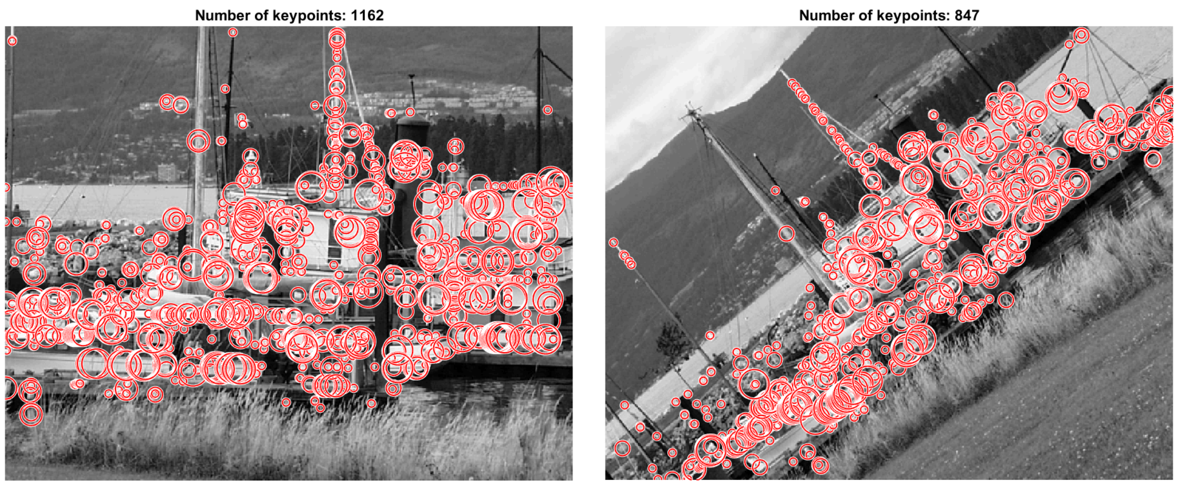

Figure 1: Detection comparison - low threshold

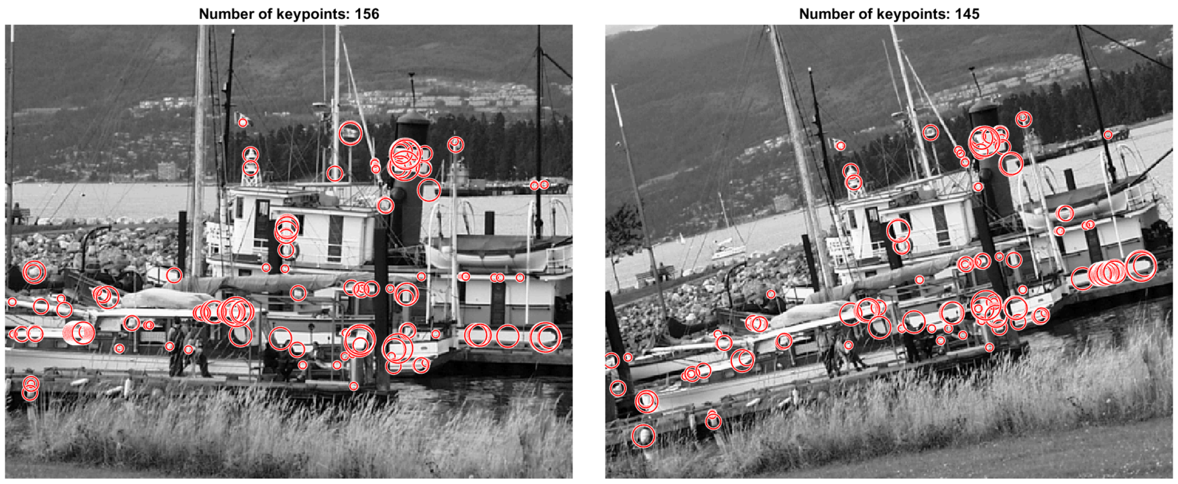

Figure 2: Detection comparison - higher threshold

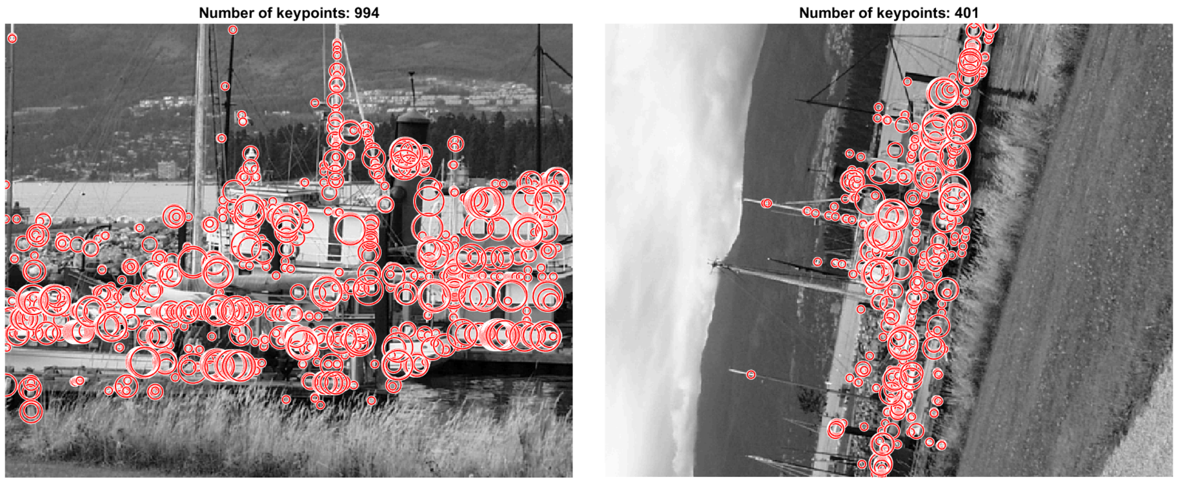

Figure 3: Detection comparison - rotation and magnification

3.2. Detecting interest points - invariance to lighting, brightness

Figure 4 shows one of the other advantages of SIFT. The change of illumination in the image on the right did reduce the number of found keypoints, but not to an extent which would invalidate later calculations.

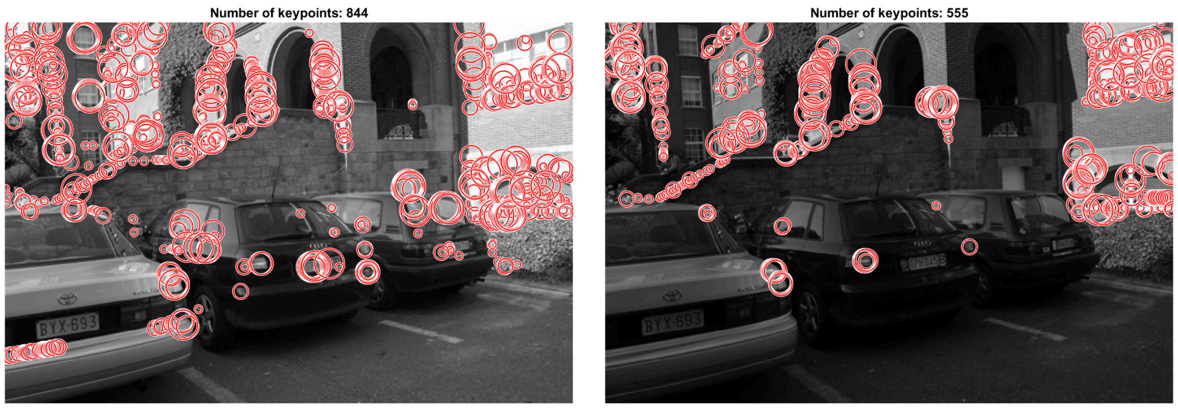

Figure 4: Detection comparison - different lighting conditions

4. Summary

SIFT characteristics, such as invariance to various factors, ranging from basic geometric manipulations like rotation to challenging environmental conditions such as variations in lighting and subtle viewpoint changes, make this algorithm an interesting and powerful tool. Being able to detect objects in different positions or with a different coverage ratio showcases the algorithm's adaptability and robustness. Moreover, its resilience to scale variations and partial occlusions enhances its utility in diverse real-world scenarios. Unfortunately we didn't manage to implement all elements of SIFT. The parts up to detection seem to work somewhat well, but to confirm this, correct descriptor extraction should be achieved, so actual descriptor matching between two images would have been possible. We definitely look forward to diving deeper into SIFT and enhancing our understanding of this algorithm's complexities. The attached code samples are both working phases of the algorithm, as well as the work-in-progress code snippets, which show our engagement. In conclusion, despite the challenges encountered in the implementation process, we explored the theory behind the Scale-Invariant Feature Transform and attempted to implement as many features as possible at the time.

5. Bibliography

[1] D. Lowe, *Object recognition from local scale-invariant features*, Proceedings of the International Conference on Computer Vision, 1999
[2] D. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision 2004
[3] S. K. Nayar, *SIFT Detector*, First Principles of Computer Vision, 2022

[4] I. Rey-Otero, M. Delbracio, *Anatomy of the SIFT Method*, Image Processing On Line, 2014