NPGR036 Homework 2

Mgr. Matúš Goliaš, Doc. RNDr. Elena Šikudová PhD.

April 18, 2023

General information

There are three tasks totalling 100 points, similar to the first homework. The first two are about implementing algorithms, and the third is about proving that a particular equality holds. We provide an assignment script `hw2.py` where you should put your implementations and an evaluation script `evaluator.py` which shows your results. The third task is a pen & paper exercise, and you should complete it in formula-friendly writing software, e.g. latex, or on paper. You should submit two files: the source file `hw2.py` with the implementation of the first two tasks and a PDF with the proof of the third task. If you need to submit more files, put all of your files into a zip archive and upload the archive.

We give you two images for experimentation and four debug images generated by the evaluator. The debug images are called *disk*, *rectangle*, *stars* and *checkerboard*. You can select a debug image by writing its name into the `--image` argument of the evaluation script. On top of that, we give you reference results for the debug images, which are compiled in the folder *ref*. The evaluation script allows you to specify a path to the reference file in the argument `--reference` to display the expected results next to your solution.

1 Harris corner detector

Implement the Harris corner detector, which computes the response function from the matrix of multiplied derivatives of a pixel area. You have to compute the response function of all four methods mentioned in the lecture. Those were:

1. *Harris and Stephens:* $\lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$.
2. *Shi and Tomasi:* λ_{min}.
3. *Triggs:* $\lambda_{min} - \alpha\lambda_{max}$
4. *Brown, Szelisky and Winder:* $\frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2}$.

Where λ denotes eigenvalues of the Harris pixel derivative matrix as described in the lecture, the matrix is 2x2, which means that it has only two eigenvalues and its decomposition is quite simple - it doesn't need general eigenvalue decomposition algorithms. Figure 1 shows the correct results produced by the evaluation script.

Your implementation should return all four response functions for the given image. These responses will have the same shape as the original image, and they should be returned in a single dictionary. The source file contains additional information about implementation requirements.

In addition, you should investigate the effect of image rotation on the algorithm performance. The evaluator lets you rotate the image by specifying the `--angle` argument. Write your findings and possible explanations in a comment section inside the function.

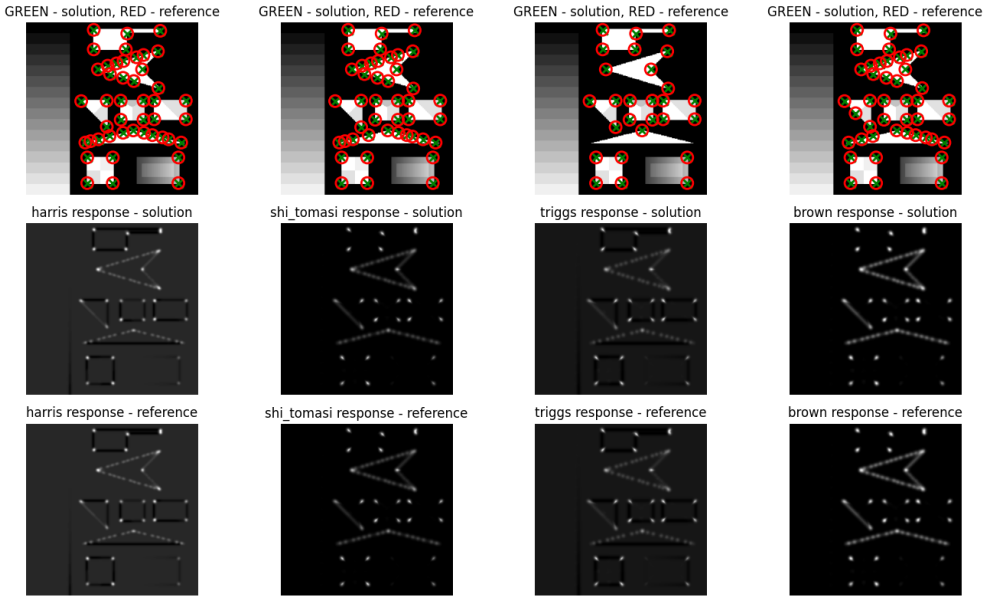

Figure 1: Example of the evaluator results for the Harris algorithm. It uses reference file, which is not given as part of the assignment.

Using any functions for eigenvalue decomposition is forbidden; thus, you are supposed to compute the eigenvalues manually, which is very simple in the 2x2 case. It is possible to find functions in NumPy, OpenCV, skimage or other libraries, which would solve the problem, but you are not allowed to use them.

2 SUSAN edge/corner detector

Implement the SUSAN corner detector, which was introduced in the lecture. The algorithm wasn't described in much detail in the lecture. Therefore you should look at the original paper, which proposed the algorithm or the rundown of the paper given in the references at the bottom of this PDF [1] [2]. You are supposed to implement USAN computation using both hard and soft function definitions of $c(\vec{r}, \vec{r_0})$. Figure 2 shows the correct results produced by the evaluation script.

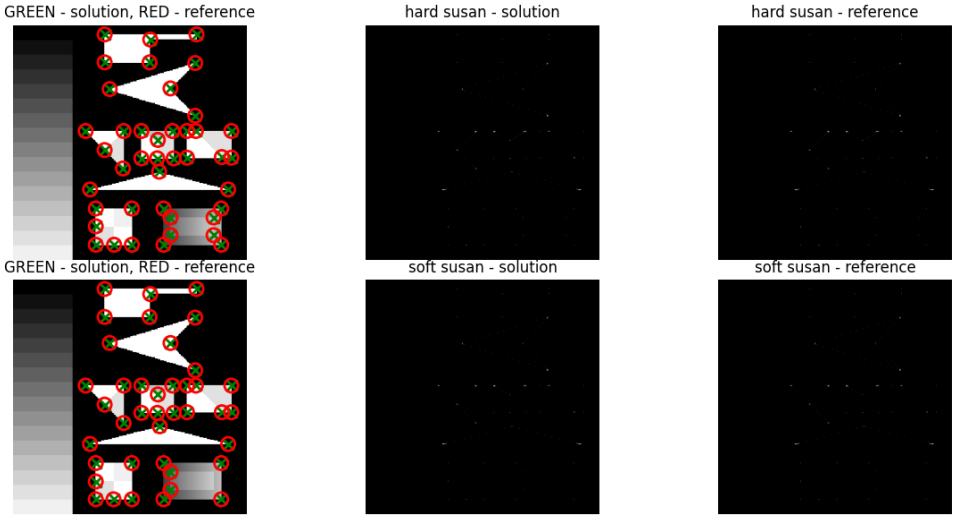

Figure 2: Example of the evaluator results for the SUSAN algorithm. It uses reference file, which is not given as part of the assignment.

The algorithm computes the number of pixels with a similar intensity to the currently processed pixel in a surrounding neighbourhood (USAN value/neighbourhood). We will consider a

disc-shaped neighbourhood with a variable integer radius given as input to your function. After computing the USAN values, the algorithm minimises them by subtracting them from a geometric constant g, which is a fraction of the largest possible USAN value. This fraction is specified as an input parameter to your function `gMultiplier`. The hard assignment of pixels into USAN is done by thresholding intensity differences, see Equation 1. On the other hand, soft assignment is done by giving a weight to all pixels in the neighbourhood, see Equation 2.

$$c(\vec{r}, \vec{r_0}) = \begin{cases} 1, & \text{if } |I(\vec{r}) - I(\vec{r_0})| \leq t \\ 0, & \text{if } |I(\vec{r}) - I(\vec{r_0})| > t \end{cases} \tag{1}$$

$$c(\vec{r}, \vec{r_0}) = e^{-(\frac{I(\vec{r}) - I(\vec{r_0})}{t})^6} \tag{2}$$

The original article and the referenced website mention that for corner detection, the algorithm computes the centroid of the neighbourhood and verifies that pixels on the line from the centre through the centroid belong to USAN neighbourhood. You do not have to do this since we will detect corners only by setting `gMultiplier` to a lower value (the default value in the evaluator is already appropriate for corner detection). Make sure to look at the references for additional details, and if something needs to be clarified, contact us, and we can discuss details about the algorithm.

In addition, you should investigate the effect of image rotation on the algorithm performance. The evaluator lets you rotate the image by specifying the `--angle` argument. Write your findings and possible explanations in a comment section inside the function.

3 Proof of Gaussian derivative equality

In lecture 7 we assumed that $\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma}$. Prove this statement. Consider the Gaussian function $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$.

Show all steps of the derivation.

1. Simplify $\sigma\nabla^2 G$.
2. Simplify $\frac{\partial G}{\partial \sigma}$.
3. Show that 1. and 2. are equal.

Write your proof on paper or in formula-friendly writing software and submit either the photo of your proof or a PDF.

References

[1] S. M. Smith and J. M. Brady, "Susan—a new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, May 1997. [Online]. Available: https://doi.org/10.1023/A:1007963824710

[2] S. Smith. The susan edge detector in detail. [Online]. Available: https://users.fmrib.ox.ac.uk/~steve/susan/susan/node6.html