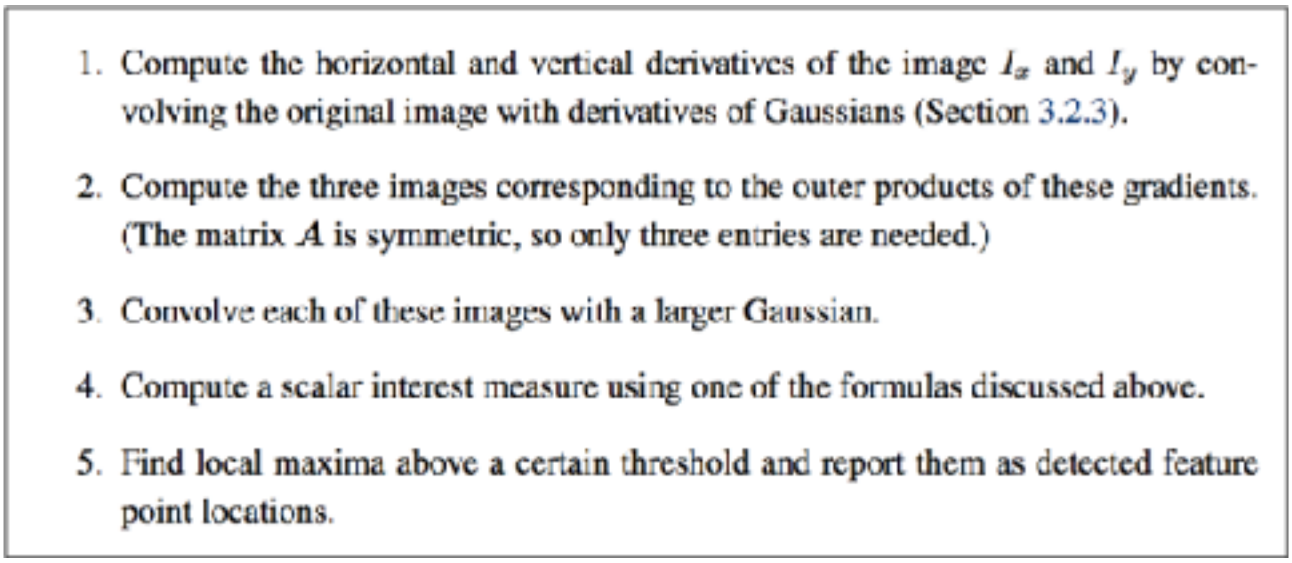
Uniwersytet Wrocławski | Instytut Informatyki

Computer Vision and Photogrammetry
Summer 2017-18

LAB 4: FEATURE DETECTION

Task 1 (30) Implement Harris corner detector

Implement the Harris corner detector as presented in class. If you use other sources of information, please state them in comments or in your Jupyter Notebook. You can use ndimage.filters for computing Gaussians and derivatives of Gaussians.



1. Compute the horizontal and vertical derivatives of the image I_x and I_y by convolving the original image with derivatives of Gaussians (Section 3.2.3).
2. Compute the three images corresponding to the outer products of these gradients. (The matrix A is symmetric, so only three entries are needed.)
3. Convolve each of these images with a larger Gaussian.
4. Compute a scalar interest measure using one of the formulas discussed above.
5. Find local maxima above a certain threshold and report them as detected feature point locations.

Szeliski's Book, Capter 4, page 214.

You can find more details in the Lecture Slides, in Szeliski's book (http://szeliski.org/Book/) or in the original paper, [C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.]

Task 2 (30) Adaptive Non-maximal Suppression.

While most feature detectors simply look for local maxima in the interest function, this can lead to an uneven distribution of feature points across the image, e.g., points will be denser in regions of higher contrast. To mitigate this problem, Adaptive Non-maximal suppression (ANMS) only detects features that are both local maxima and whose response value is significantly (10%) greater than that of all of its neighbors within a radius **r**. They devise an efficient way to associate suppression radii with all local maxima by first sorting them by their response strength and then creating a second list sorted by decreasing suppression radius. The output is the selection of the top n features.

Task 3 (40) Scale Invariant Keypoint Detector.

Implement a scale invariant keypoint detector. Main two options are Difference of Gaussians (DoG) which is typically employed within SIFT, or the Harris-Laplace.

Harris-Laplace
Paramters for the Harris-Laplace, you can use scipy.ndimage.filters.gaussian_laplace.

```
    # SCALE PARAMETERS
    sigma_begin = 1.5
    sigma_step  = 1.2
    sigma_nb    = 13
#sigmas for all different scales
    sigma_I = (sigma_step.^range(sigma_nb-1))*sigma_begin
#sigmas for derivatives
    sigma_D = 0.7 * sigma_I
```

Difference of Gaussians (more complicated)

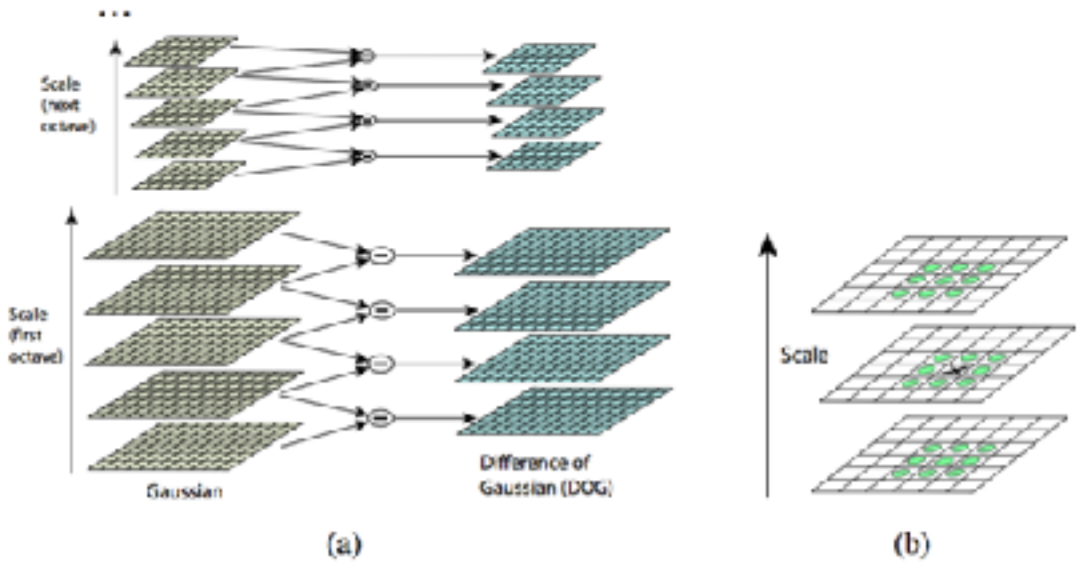


Figure 4.11 Scale-space feature detection using a sub-octave Difference of Gaussian pyramid (Lowe 2004) © 2004 Springer: (a) Adjacent levels of a sub-octave Gaussian pyramid are subtracted to produce Difference of Gaussian images; (b) extrema (maxima and minima) in the resulting 3D volume are detected by comparing a pixel to its 26 neighbors.

Use four octaves. The sigmas between scales are k*sigma where k = 2^(1/s) with s being the number of scales. According to Szeliski, s = 3 is optimal. Initial sigma 1.6. After selecting the maxima and minima between scales, use the hessian to remove edge responses. Similar to the Harris detector, but with different fomula. Please, check the references for details.

Some extra information can be found in Szeliski (216 and 217), the original paper (http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf) and the opencv tutorial. I think they got some of the parameters in the description wrong. http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html

Extra:

There are several detectors implemented in OpenCV and other python libraries. Test, compare, and read about them.