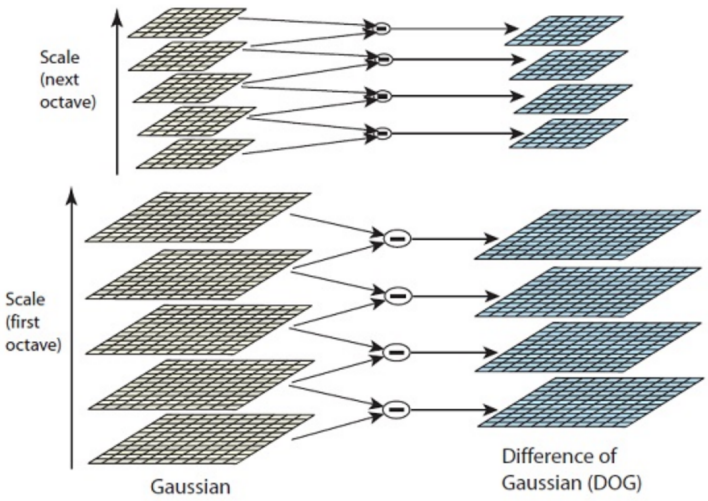
Computer Vision
Lab Exercise 2
Harris Corner Detector and SIFT Descriptor

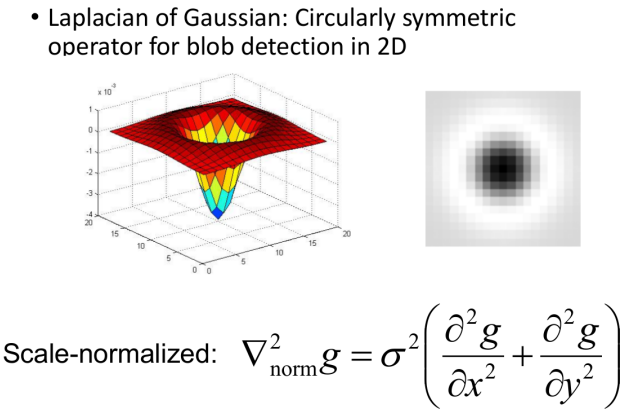
Students should work on this lab exercise in groups of two people. In this assignment, you will be implementing the scale invariant Harris Corner Detector. You will then use this detector to match images with the SIFT descriptor.

1 Scale selection using Laplacian



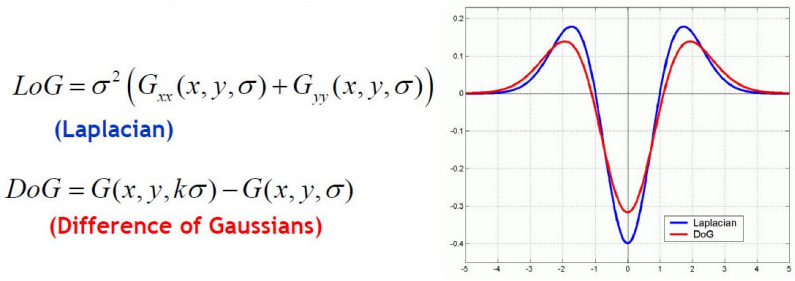
We want to find corner-points that are scale invariant. We can find these as extrema points of the Laplacian for the optimal scale, σ^*. To find the scale we need to create a scale-space as in the SIFT paper [1] by blurring the image with increasing σ values. The SIFT paper is attached on Brightspace.

Computing the Laplacian: We need to compute the Laplacian over these multiple scales and find the local extrema within a neighborhood. The Laplacian of a Gaussian can be computed using the formula below. And then a

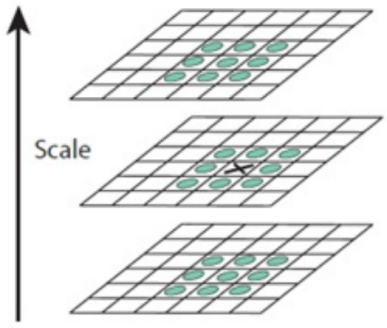


standard image convolution operation can be used for extracting the Laplacian responses.

Computing the DoG approximation of Laplacian: Alternatively, the Laplacian can be approximated with the DoG (Difference of Gaussians), directly from the scale-space. For this you would just have to subtract every two adjacent layers.



Finding local extream of Laplacian: We will search for the maximum Laplacian responses within a 3×3 neighborhood, by looking also at the layer above and the layer below the current layer in the scale-space.



We can then filter the points to remove the ones with low-responses. We then, obtain a list of interest points of the form: $[r_i, c_i, \sigma_i]$. These points can also be edges and blobs.

All the steps above when using the DoG Laplacian approximation are being performed by the provided $< DoG.m >$ file in Brightspace. This file returns a set of image locations together with their selected scale. We want to retain only the corners.

2 Harris Corner Detector

In this part, you will be implementing a scale invariant Harris Corner Detector. In the first step you need to complete the code in the Harris function that is provided on Brightspace: $< harris.m >$.

All the necessary steps are indicated in the code with the hint for the completion. You need to compute the entries of the structure tensor matrix which you will use to form your 'cornerness'(R)). (Please review your lecture slides for this task.)

The corner points are the local maxima of R. Therefore, in your function you should check for every point in the list of initial interest points, if in R its value is greater than all its neighbours (in an $[n \times n]$ window centered around this point) and if it is greater than the user-defined threshold, then

it is a corner point.

Now, the points which are returned are corners and scale invariant. Your function should return the rows of the detected corner points r, and the columns of those points c and the scale at which they were found: σ (the first corner is given by $(r(1), c(1), \sigma(1))$). Your function should also plot the original image with the corner points plotted on it (Extra: You could plot them as circles where the radius is the corresponding σ_i.)

3 Image Matching with SIFT

In this part, you will be using your corner points detected in the previous section. You will need to build a descriptor for each of these corner points in the given image-a. And try to find closest descriptor in the other given image-b using an Euclidean distance.

First start by extracting image descriptors at a patch around the detected corner points. You can find an implementation of SIFT descriptors at `http://www.vlfeat.org/index.html`. Download and setup the vlfeat package for Matlab.

```
% Find features and make descriptor of image 1
loc1                  = DoG(im1, 0.01);
[r1, c1, sigma1]      = harris(im1, loc1);
orient1               = zeros(size(sigma1));
% Pay attention to the oder of parameters [c',r'] (equal ...
    to [x,y])
[coord1, descriptor1] = sift(single(rgb2gray(im1)), ...
    'frames', [c1'; r1'; sigma1'; orient1']);

% Find features and make descriptor of image 2
loc2                  = DoG(im2,0.01);
[r2, c2, sigma2]      = harris(im2, loc2);
orient2               = zeros(size(sigma2));
[coord2, descriptor2] = sift(single(rgb2gray(im2)), ...
    'frames', [c2'; r2'; sigma2'; orient2']);
```

Then, loop over these descriptors and for each, find its closes match in the other image, based on the Euclidian distance. The descriptors must be

normalized to unit norm.

```
    desc1 = double(descriptor1(:, index1));
    desc2 = double(descriptor2(:, index2));

    % Normalize the descriptors to unit L2 norm:
    desc1 = ...;
    desc2 = ...;

    % Euclidian distance:
    dist = ...
```

You can define your own threshold for finding matches based on the Euclidian distance. Additionally the correct matches can be filtered based on the ratio of the second-best-match to the best-match, as below. Also you can reject matches that have distance ratios:

$\frac{\text{bestDist}}{\text{secondBestDist}} \geq 0.8$.

References

[1] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.