CIS 580, Machine Perception, Spring 2020
Homework 5
Due: Monday April 8th 2019, 11:59pm

Instructions

- This is an individual homework.
- You must submit your solutions on Gradescope, the entry code is MB8ZJP.. You must submit a PDF report and a jupyter notebook file with your code. We recommend that you use LaTeX for the report, but we will accept scanned solutions as well.
- Use PyTorch (the `torch` Python package) to complete this homework. The only other python package that you may use is math.
- You will use Google Colaboratory `https://colab.research.google.com/` for this homework, which is a free online service that lets you run Python code in Jupyter notebooks and provides free GPU compute.
- Start early! If you get stuck, please post your questions on Piazza or come to office hours!

Homework

1 Setup

Please download the zip folder from It will contain an Homework6.ipynb and some images. Upload Hopework6.ipynb to Google colab and you will be able to edit it and run the code it contains.

2 Edge detection with Gaussian derivatives, 30 points

In this section, we will perform edge detection using a Gaussian derivative filter.

First, we will will define helper functions for generating discretized Gaussian filters and Gaussian derivative filters.

1. **[6 points]** Complete the function `gaussian1d` that returns discrete normalized centered Gaussian of given standard deviation σ and length. Include the generated plot in your report.
2. **[6 points]** Now complete the function `gaussian_derivative1d` that returns the corresponding derivative values of the Gaussian. Include the generated plot in your report.
3. **[6 points]** Use the `gaussian_derivative1d` function you defined to generate two 2D Gaussian derivative filters. One of the filters should be able to detect horizontal edges while the other should detect vertical edges.
4. **[6 points]** Using the torch library, complete the script by applying the kernels to the image cup.jpg to detect the edges in both x and y dimension.
5. **[6 points]** The script will plot the edges detected with several different sizes of filters. What do you notice about the different images? What effects does the size of the filter have? When might you want to use larger or smaller filters? Include the plots and your answers to these questions in your report.

3 Scale invariant detection, 70 points

In this section, we will implement a scale-invariant blob detector.We will apply a series of DoG filters to the initial image to build a 3D-matrix of responses, and we will then find local maxima in position and scale.

1. **Laplacian of a Gaussian, 5 points**

   In this section we will generate the Laplacian of a Gaussian.

   (a) **[5 points]** Complete the function `log1d`. Include the generate graph in your report.

2. **Approximating a LoG by a DoG, 25 points**

   In this section we will experiment with approximating a Laplacian of Gaussian filter by a Difference of Gaussians. To build the intuition we will reason in 1D, i.e with a section of the filter.

   (a) **[5 points]** Complete the function `dog1d` to compute the difference of gaussian filters.

   (b) **[10 points]** Complete the next block of code to plot the LoG, as well as its approximation by DoG filters. Remember the following:

   $$\mathrm{LoG}_\sigma = \frac{1}{(k-1)\sigma^2}\mathrm{DoG}_\sigma, \quad \text{where } \mathrm{DoG}_\sigma = G_{k\sigma} - G_\sigma$$

   In the Approximating a LoG by a DoG section, we will try several values of k in `k_range`. Quickly comment on the plots you obtain.

   (c) **[5 points]** Explain why you could expect the DoG to get closer to the LoG as k tends to 1. (Remark: Note that in practice, when building a scale space using DoG, we prefer using $k = \sqrt{2}$)
   So $\mathrm{DoG}/(k\sigma^2 - \sigma^2)$ approaches the LoG when $k \to 1$.

   (d) **[5 points]** In practice, when building the scale space, we intentionally forget the normalizing factor $\frac{1}{(k-1)\sigma^2}$ and just use the differences of Gaussians. Explain why.

3. **Detecting sunflowers, 40 points**

   The LoG filter in 2D is a rotationally symmetric version of the 1D filter we worked with in the previous question (the famous "mexican hat"). Because of its shape, it makes a good blob detector (a blob is a dark patch on light background). We don't know the size of the blobs to detect a priori, so we build a scale space by applying the filter with wider and wider σ. In practice we will approximate the LoG by a DoG. We will implement a simple (but inefficient) version where we do not downsample the image when blurring it.

   (a) **[6 points]** : Implement the `gaussian_2d` function which is a 2D analogue of the `gaussian_1d` function.

   (b) **[10 points]** : Create blurred versions of the image by applying 2D gaussian filters with different scales using the `torch` library. *Hint: If your are using Colab, this computation must be performed on the free GPU to speed up computation.*

   (c) **[6 points]** Complete the part of the section that filters the local maxima in the scale space according to their response. Here we want to keep only points (x, y, σ) that have a response higher than 50% of the maximum response across the whole 3D scale space.

   (d) **[6 points]** The radius of a detected blob corresponds to $\sqrt{2}$ times the detected scale: complete the formula for the radius `r` of each detection, at the end of the section, in order to plot the defections as circles of detected radius on top of the image. `r` depends on `sigma`, `k`, and the detected scale (use `smax`).

   (e) **[12 points]** Show your detection results for the image `sunflowers.jpg` and for another image of your choice containing blobs at various scales. Remark: In case you wonder how to detect white blobs on a dark background, take the local minima instead of local maxima.