CIS 580, Machine Perception, Spring 2019
Homework 5
Due: Monday April 15th 2019, 5pm

Instructions

- This is an individual homework and worth 100 points
- You must submit your solutions on Gradescope, the entry code is 96JGNN. We recommend that you use LaTeX, but we will accept scanned solutions as well.
- You must submit code to the autograder through turnin on a CETS run server.
- Start early! If you get stuck, please post your questions on Piazza or come to office hours!
- The following files must be submitted via turnin: `gaussian1d.m`, `gaussian2d.m`, `gaborFilter1D.m`, `gaborFilter2D.m`, `dog1d.m`. We provide a few tests; use `run test/run_tests` to run them. Please make sure that the given tests are passing before submitting your work.

Homework

1 Gabor filters

In this section, we will implement 2D detection using a Gabor filter. We will first play with two toy examples in 1D and 2D to understand the fundamentals, then we will attempt to have your computer play the game of "Where's Waldo".

1. **Gabor filter in 1D**

   In this question we will consider the script `gabor1d_script.m` in the homework kit: it generates a 1D signal of increasing frequency `s` as well as a vector `s_w` containing ground-truth frequencies (see figure 1).

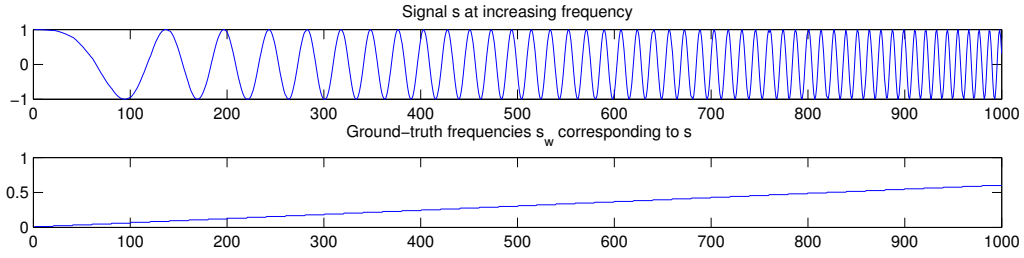

Figure 1: Signal at increasing frequencies and corresponding ground-truth frequencies.

(a) **[2 points]** Complete the function `gaussian1d.m` that returns discrete normalized centered Gaussian of given standard deviation σ and length.

(b) **[4 points]** Complete the function `gaborFilter1D.m` that returns a pair of Gabor quadrature filters at given spatial period `T_f` in pixels, Gaussian envelope σ and length. The two filters correspond respectively to the real and imaginary part of the Gabor filter we defined in class.

(c) **[6 points]** Complete the script `gabor1d_script.m` to compute the filter response: `r1`, `r2` are the imaginary and real parts of the response, and `energy` is the magnitude of the response (complex norm).

(d) **[4 points]** Using the script, plot figures for two distinct values of T_f and check that the maximum response matches the ground truth.

2. **Gabor filter in 2D**

We now consider the same problem in 2D: we want to detect patterns with a specific spatial period *and* orientation in the image represented in figure 2.

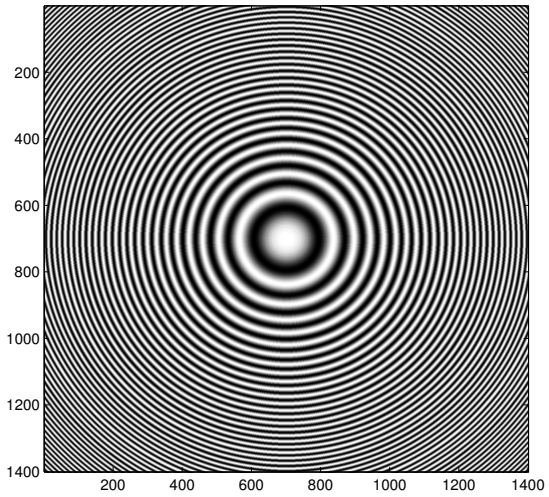

Figure 2: Image containing a range of frequencies and orientations.

(a) **[4 points]** Complete the function `gaussian2d.m` that returns discrete normalized centered 2D Gaussian of given covariance matrix Σ and size.

(b) **[8 points]** Complete the function `gaborFilter2D.m` that returns a pair of Gabor quadrature filters to detect a given spatial period `T_f` in pixels **and** orientation `theta` in degrees. The two filters correspond respectively to the real and imaginary part of the Gabor filter we defined in class.

(c) **[6 points]** Complete the script `gabor2d_script.m` to compute the filter response: `r1`, `r2` are the imaginary and real parts of the response, and `energy` is the magnitude of the response (complex norm).

(d) **[4 points]** Using the script, plot figures for two distinct values of T_f and check that the maximum response matches the ground truth.

2 Scale invariant detection

In this section, we will implement a scale-invariant blob detector that produces results as in figure 3: we will recursively apply a DoG filter to the initial image to build a 3D-matrix of responses, and we will then find local maxima in position and scale.

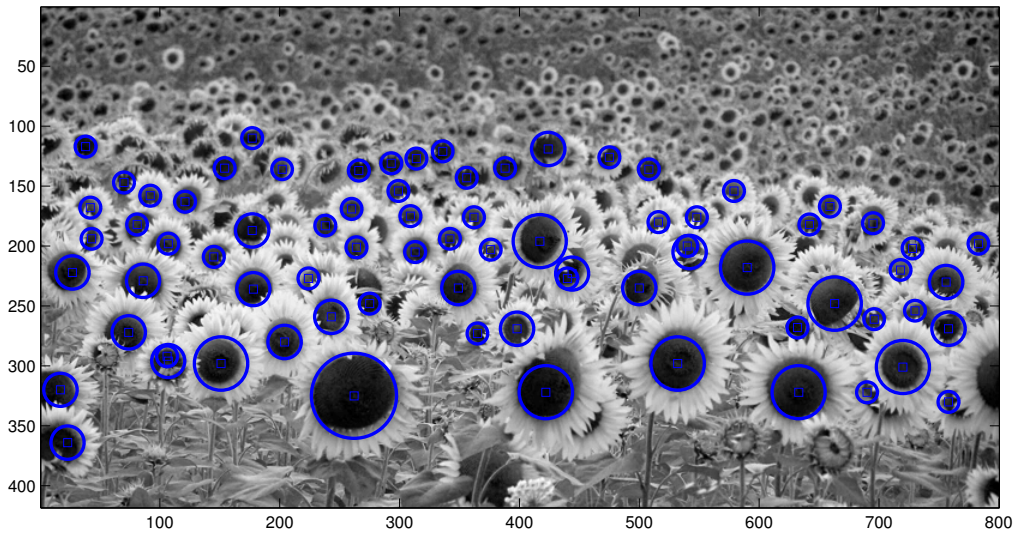

Figure 3: Scale-invariant blob detection example.

1. **Approximating a LoG by a DoG**

   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) **[14 points]** We want to compute and plot a LoG filter, as well as its approximation by DoG filters. Remember the following:

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

   In `DoG_script.m`, we will try several values of k in `k_range`. Complete `dog1d.m` and `DoG_script.m`. Quickly comment on the plots you obtain.

   (b) **[4 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 $k \to 1$.

   (c) **[4 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.

2. **Detecting sunflowers**

   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) **Preliminaries**

In this section the only file you need to complete and run is `blob_script.m`. Make sure you:

- change the two paths at the beginning of the file: one is for the image, the other for a function `minmaxfilt` used to find local maxima/minima in an n-dimensional array.
- run `minmaxfilter_install.m` in `MinMaxFilterFolder`, to compile the mex files for your platform. (The homework kit already includes compiled mex files for 64-bit Linux, so you can skip this step if you're using that platform)

(b) **[12 points]** Complete the part of `blob_script.m` that builds the 3D matrix `scales`. Each slice `scales(:,:,i)` contains the original image, blurred by a simple Gaussian filter of parameter `sigma_i = sigma * k^(i-1)`.

Important: we exploit the fact that the 2D Gaussian filter is separable. Instead of convolving the image with a 2D Gaussian, build a 1D Gaussian filter `g_i` (using `gaussian1d.m`), convolve the image with `g_i` and then with `g_i'` (transpose of `g_i`): this has the same effect as convolving with the 2D filter but is more efficient.

(c) **[12 points]** Complete the part of `blob_script.m` 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 accross 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 `blob_script.m`, in order to plot the detections 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.