CIS 580, Machine Perception, Spring 2021
Homework 7: Image Processing.
Deadline: Friday 4/23 11:59pm

1 Scale Invariant Detection [60 pts]

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. The test image we will use in this section is the sunflowers.jpg. (Template Code is available here)

1.1 Laplacian of Gaussian 5 points

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

1. **[5 points]** Complete the function log1d. Include the generated graph in your report.

1.2 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.

1. **[5 points]** Complete the function dog1d to compute the difference of gaussian filters.
2. **[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.

3. **[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 $DoG/\left(k\sigma^2 - \sigma^2\right)$ approaches the LoG when $k \to 1$.
4. **[5 points]** In practice, when building the scale space, we intentionally forget the normalizing factor $\frac{1}{(k-1)x^2}$ and just use the differences of Gaussians. Explain why.

1.3 Detecting sunflowers, 20 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.

1. **[10 points]** Complete the function **gaussian2d** that returns a discrete normalized centered 2D Gaussian, for a given standard deviation σ and size.

2. **[10 points]** In the section "*Detecting Blobs*" of the template: Create blurred versions of the image by applying 2D gaussian filters with different scales using the PyTorch framework (torch). Hint: If your are using Colab, this computation must be performed on the free GPU to speed up computation.
   Some useful functions:

   - *torch.from_numpy coverts a numpy array to array*
   - *torch.nn.Conv2d performs a 2D convolution*

3. **[5 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.

4. **[5 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.

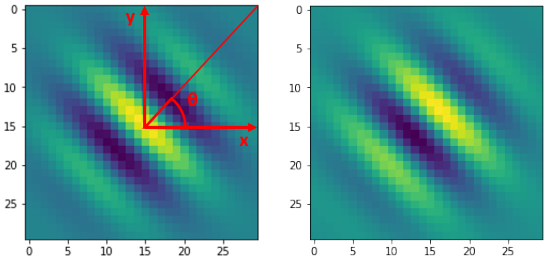
2 Finding Waldo with Gabor Filters [40 pts]

In this section, we will use Gabor filters to implement 2D detection. The goal of the Gabor Filter is to detect patterns with a specific spatial period and orientation. (Template Code is available here).

Here we will consider Gabor filters with the same standard deviation σ in both axes. The Gabor filter with period T, standard deviation σ and orientation θ (in degrees) is defined as

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \left(\cos(2\pi x_\theta / T) + i \sin(2\pi x_\theta / T) \right)$$

where i is the imaginary number and x_θ is defined as $x_\theta = \cos(\pi\theta/180)x + \sin(\pi\theta/180)y$ (and corresponds to the rotated x coordinate).



(a) Real Part (b) Imaginary Part

Figure 1: Real and imaginary part of Gabor filter with $T = 10$, $\sigma = 5\sqrt{2}$, $\theta = 45^o$

Notice that the Gabor filter has both a real and an imaginary part. Since we are interested in filtering real signals we can split the complex convolution in two convolutions. One convolution of the signal with the

real part of the filter and one with the imaginary. These two convolutions will give us the corresponding real and imaginary part of the result.
In figure 1 we can see the real and imaginary part of a gabor filter with period $T = 10$, standard deviation $\sigma = 5\sqrt{2}$ and orientation $\theta = 45^o$

2.1 Gabor Filter in 2D

First we will apply the Gabor filter in an image containing a range of frequencies and orientations shown in Figure 2 (also available here)

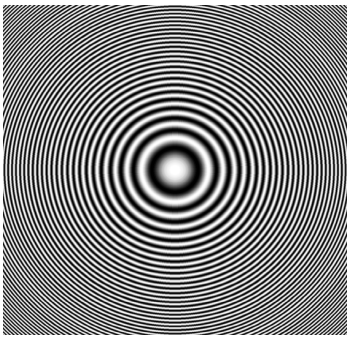


Figure 2: Image containing a range of frequencies and orientations

(a) **[10 points]** Complete the function **GaborFilter2D** that returns a pair of Gabor quadrature filters used 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.

(b) **[10 points]** Complete the section "*Check Gabor filter*" of the template. Here you should convolve the image increased_frequency.png with Gabor filters with different spatial periods and orientations. To do this, you need to compute the real part of the convolution, the imaginary part and also the magnitude of the convolution.

(c) **[4 points]** Include the plots produced by the previous script along with a small explanation about the different results.

2.2 Finding Waldo

Now we consider the classical game of finding Waldo (a character with a red-and-white stripped shirt) in the image shown in Figure 3 (also available here)

(a) **[5 points]** Complete the function **determineStripePeriod**. This function takes as input the path to an image (stripe_pattern.jpg) that contains only the stripe pattern in Waldo's shirt.
It computes the magnitude of the FFT in the frequency range corresponding to frequencies $(\omega_y, \omega_x) \in [0, \pi] \times [0, \pi]$. Using this magnitude you will need to find the frequency ω_y^* that corresponds to the peak of the FFT in this range. Then you should use ω_y^* to compute the spatial period T_y of Waldo's stripes (spatial period in pixels along the y-axis).

(b) **[3 points]** Read the script in the section "*Load Image*" of the template. Describe in a few words the formula for the creation of **im_red** image.

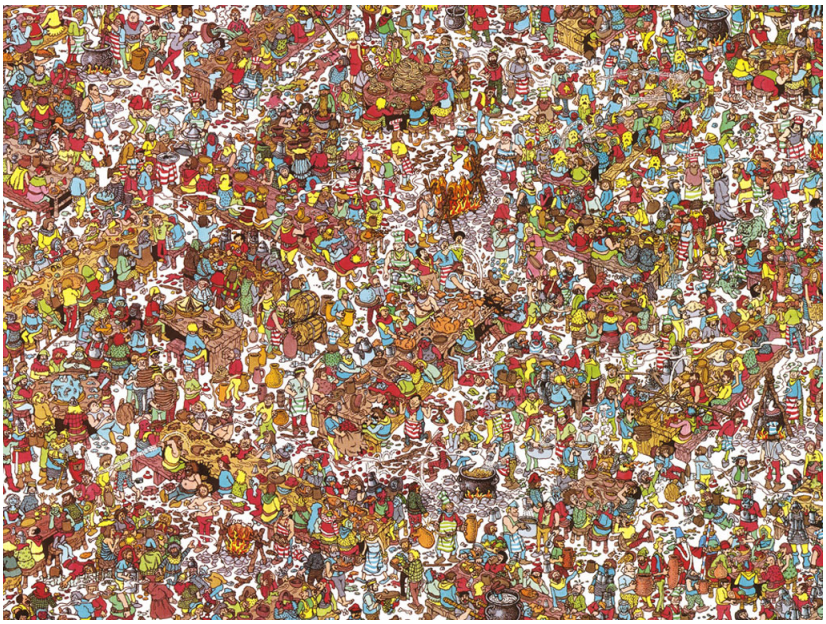


Figure 3: Can you find Waldo in this picture?

(c) **[5 points]** Complete the section "*Compute candidate areas containing the stripe pattern*" of the template. Here you should filter the image containing Waldo with a Gabor filter that is created using the spatial period T_y.

(d) **[3 points]** Include the masked image produced by the script in the final block of code. Are you able to find Waldo?