CIS 580: Machine Perception Homework 3

Gabor Function and Frequency Matching

Dan Harris

03-15-2018

Problem 1

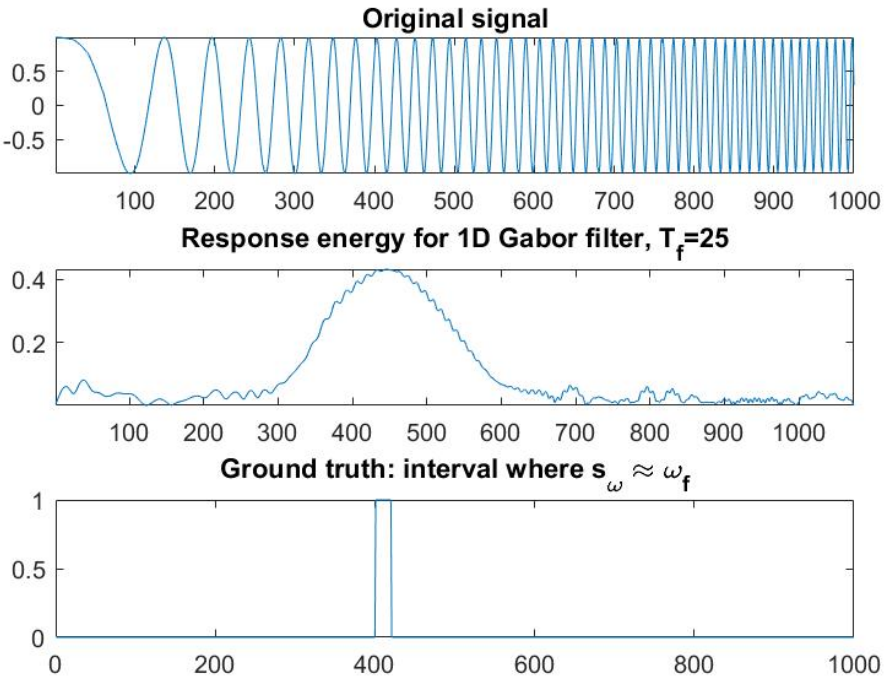


Figure 1: Energy Response for $T_f = 25$

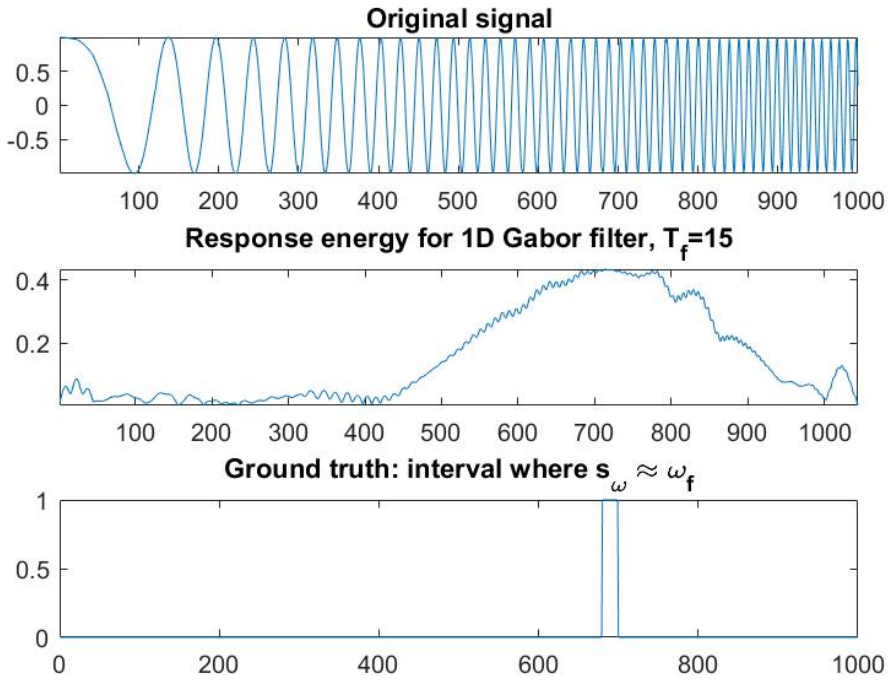


Figure 2: Energy Response for $T_f = 15$

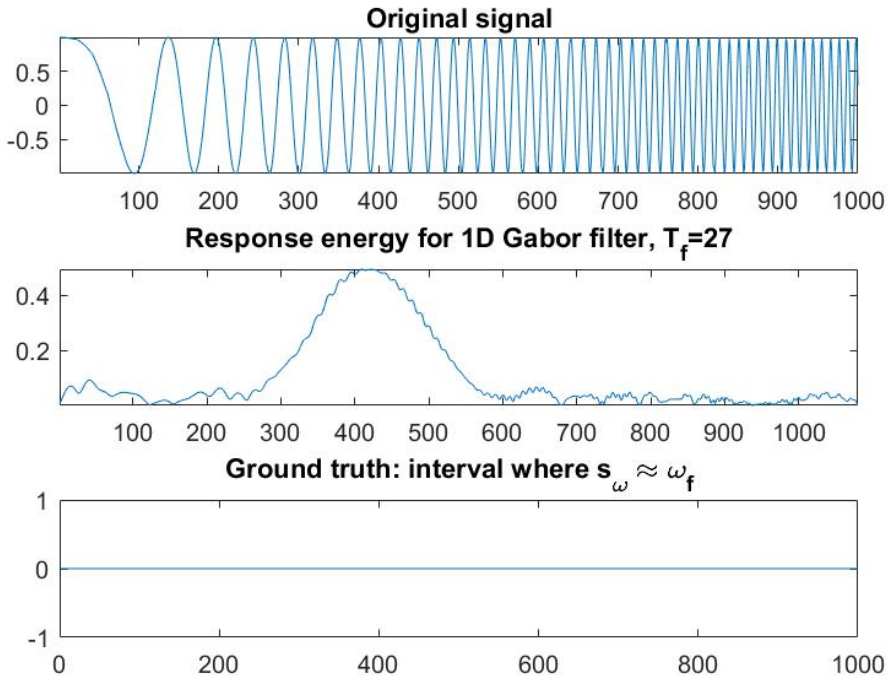


Figure 3: Energy Response for $T_f = 30$

We see that our ground truth frequency tracks the maximum energy response for a given frequency. However, when T_f falls outside the frequency range of the original signal, we see that the ground truth can no longer match the maximum energy response.

Problem 2

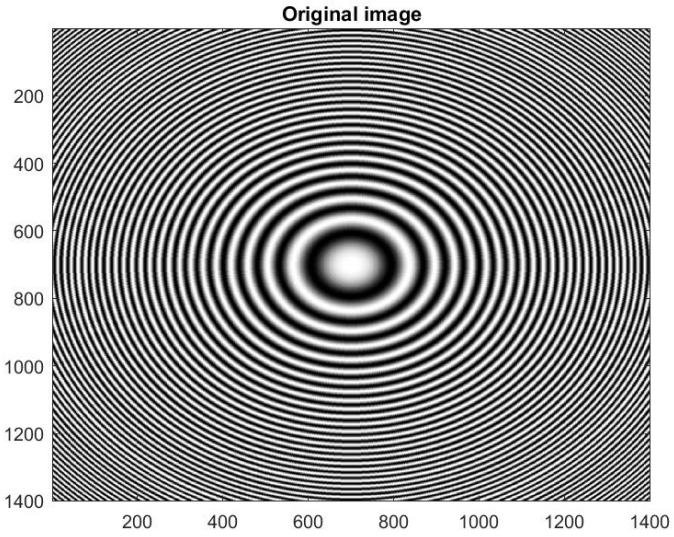
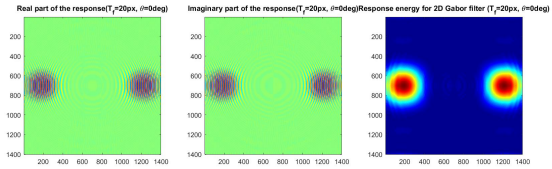
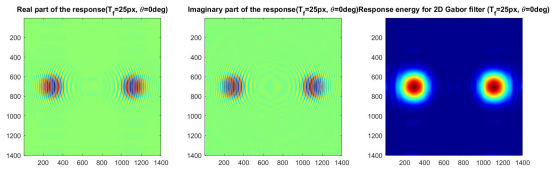


Figure 4: Original Image

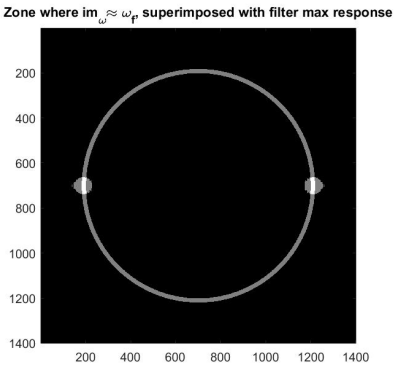


(a) Gabor Filter Response ($T_f = 20, \theta = 0$)

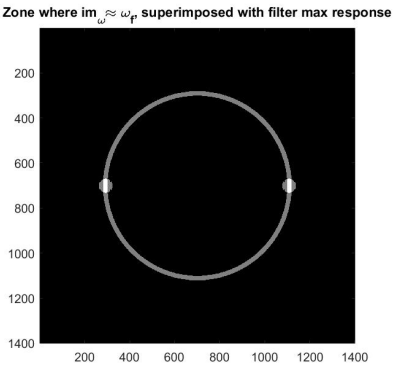


(b) Gabor Filter Response ($T_f = 25, \theta = 0$)

Figure 5: Energy response for gabor filter on original image

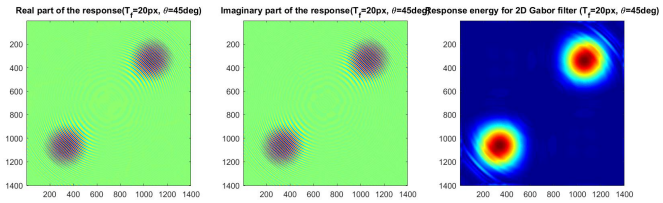


(a) Max frequency response ($T_f = 20, \theta = 0$)

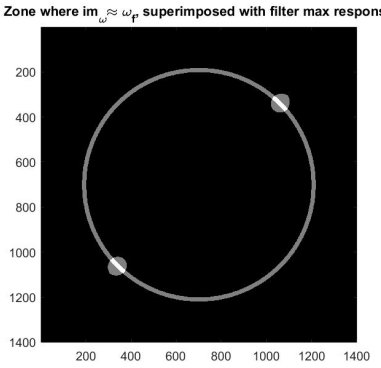


(b) Max frequency response ($T_f = 25, \theta = 0$)

Figure 6: Frequency superimposed on max filter response



(a) Gabor Filter Response ($T_f = 20, \theta = 45$)



(b) Max frequency response ($T_f = 20, \theta = 45$)

Figure 7: Gabor filter and Max Filter Response rotated by 45 degrees

As seen in Figures 6 and 7, the maximum frequency response matches the ground truth frequency even when rotated by θ.

Problem 3

(a.) This script prompts the user to select a region of a given image, in this case a Where's Waldo image. The user must select a region within the image that contains a horizontal striped pattern from which "determinestripedperiod.m" will take the Fourier transform and the user must select the maximum frequency to

determine the spatial period of the image. This spatial period is then used in the Gabor filter in order to find the regions of maximum energy (i.e high energy corresponds to regions within the image that most closely match the calculated spatial period). Lastly, this energy array is used to calculate a detection mask of binary ones and zeros where the ones represent regions of the image that are above a specified threshold. These regions are then dilated and displayed. The goal of this is to adjust the threshold and find Waldo!

(b.) "*imred*" takes the red channel and blue channels of the image and subtracts half the green channel. This is because we eventually want the function to be able to detect all red and white stripes in the image to find Waldo. Therefore, after doing so, we set all elements in the image with a pixel value less than the average value of the red channel to the mean and then subtract the mean to zero out all terms that had values less than the mean in order to accentuate the most prominent red stripes relative to other regions in the image.

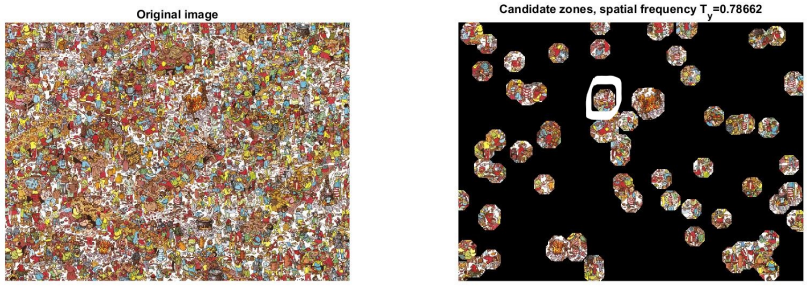


Figure 8: Original Image vs. Dilated and Thresholded Result

Waldo is identified by the white circle. The values in the energy array were normalized to be between 0 and 1. After doing so, any values below a threshold of 0.8 was used to find all regions that matched the frequency of the selected region by at least 80 percent.

Problem 4

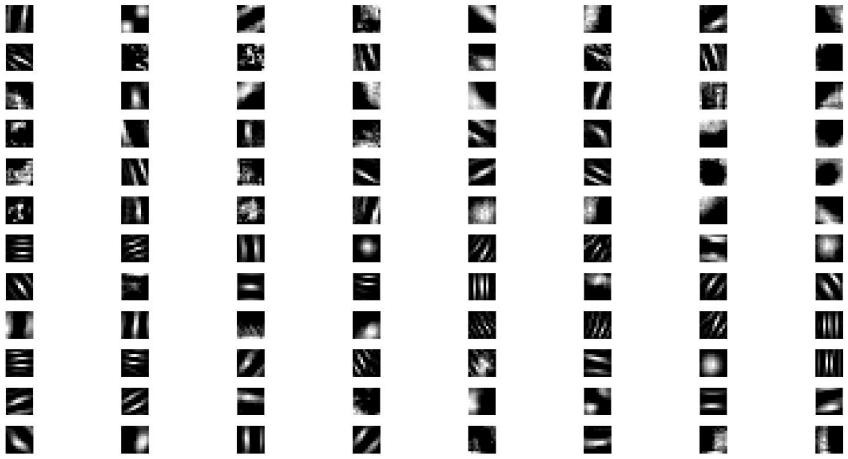


Figure 9: Alex Net Filters

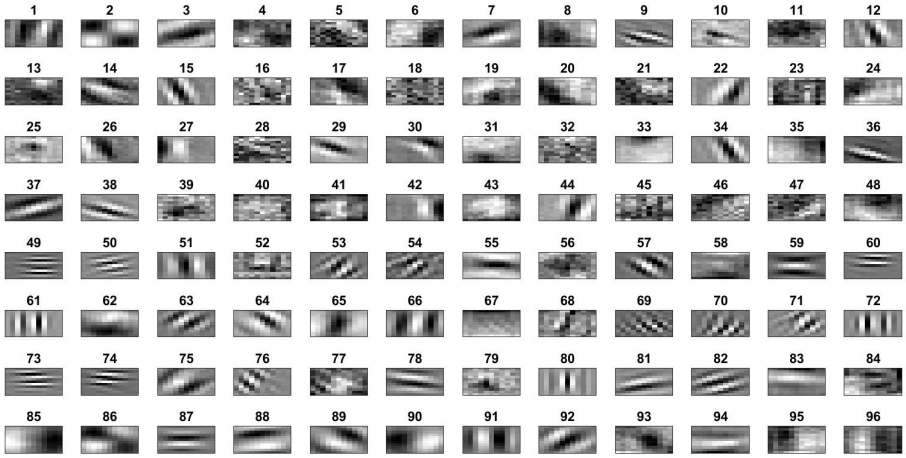
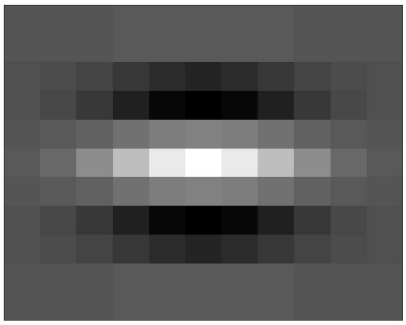


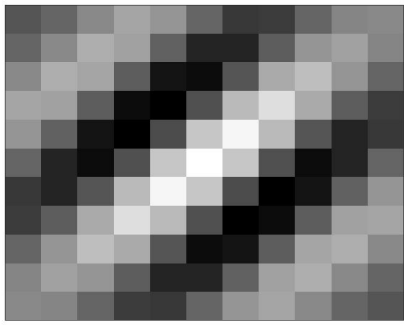
Figure 10: Alex Net Filters [8,12]

Several of the filters in AlexNet can be approximated using the 2D Gabor function. Specifically, any of the filters with periodic scaling of values (resembled by stripes using imagesc). Therefore, some examples of these can be seen in Figure 10 for images 37, 59, 63, 75, 82, and many more that match the description previously explained.

As we can see from Figure 11, the filters from AlexNet can be approximated using 2D Gabor filters. Each of these 2D Gabor filters, given a specified length, have 5 parameters than can be adjusted: σ_x, σ_y, $sigma_{xy}$,θ and T_f. Therefore, if the first layer of AlexNet is replaced by 96 Gabor filters , 480 parameters would have to be learned.



(a) Gabor Filter Approximation
($T_f = 1.25, \theta = 90$)



(b) Gabor Filter Approximation
($T_f = 4.9, \theta = -45$)

Figure 11: Gabor Filter Approximation for AlexNet

Problem 5.1

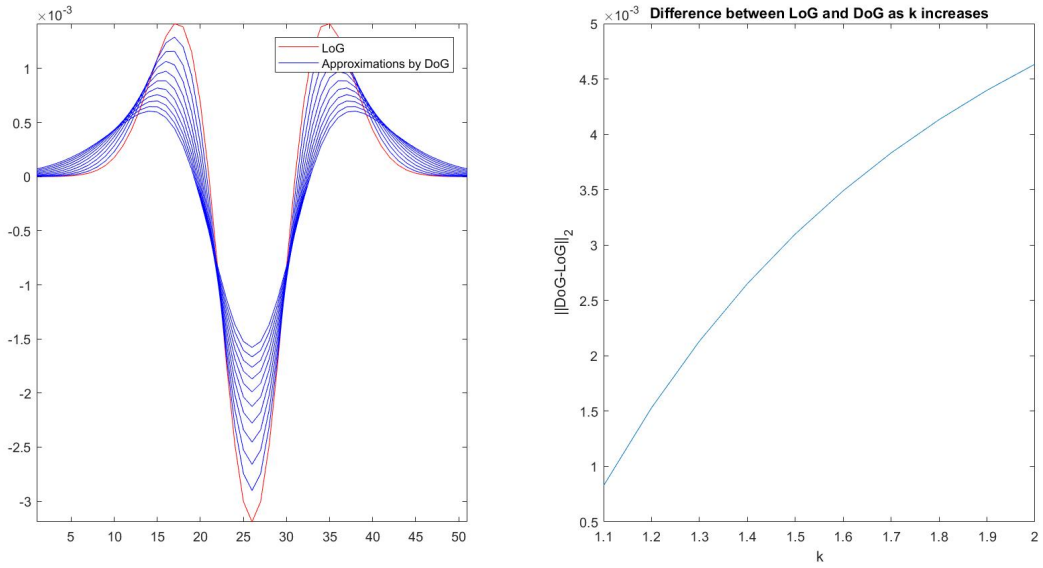


Figure 12: DoG w.r.t to k

Part A: [Figure 12]

(b.) LoG is the Lapacian of the Gaussian function which is essentially the second derivative of the Gaussian while DoG is the difference between two Gaussian functions. The term $\frac{1}{(k-1)\sigma^2}$ is the normalization factor where k acts similar to how a time step would work when taking an integral. Therefore, by increasing the number of steps (i.e as k gets smaller), the numerical approximation of the DoG gets closer and closer to the LoG. The Laplacian is the more ideal solution to be used during blob detection but it is also computationally more expensive so we approximate it using DoG.

(c.) If we are keeping k constant (in this case $k = \sqrt{(2)}$), then we can essentially leave out the normalizing factor because it would just scale our answer down. Also, given that it is constant, we would be able to factor it out from DoG and multiply later anyway.

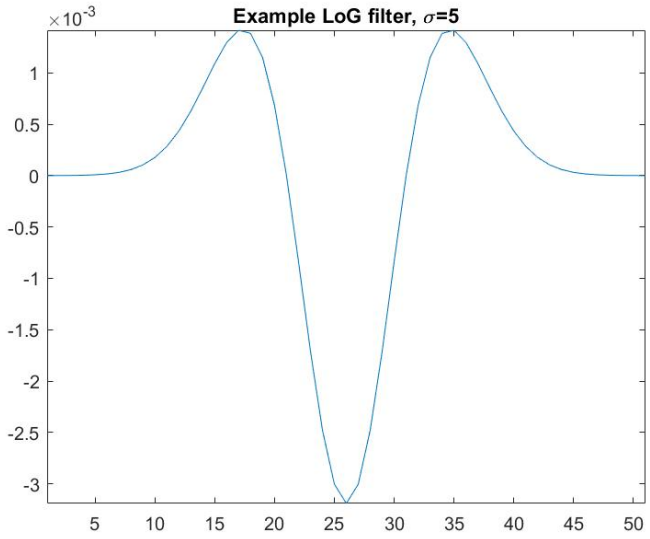
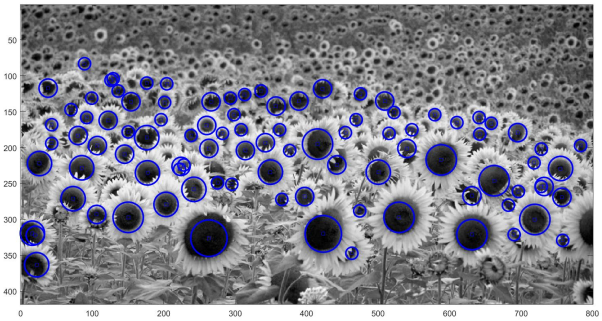


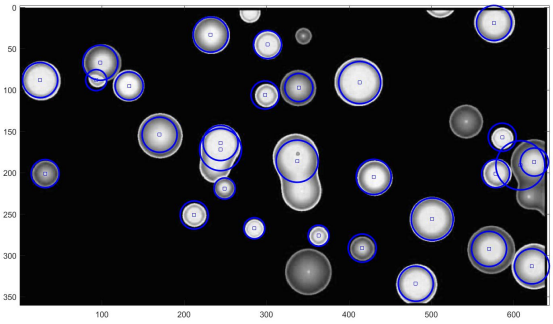
Figure 13: DoG Plot for $\sigma = 5$

Problem 5.2

The following two images illustrate blob finding for images with blobs of different scales with light backgrounds as well as dark backgrounds. The radius of each blob is scaled by the value of its detected sigma.



(a) Sunflower Blob Detection



(b) Miscellaneous Blob Detection

Figure 14: Blob detection for Light and Dark Blobs